Investigation on Image Quality Degradation due to Movement in Automotive Cameras

Bachelor Thesis

by

Julian Kremming & Mike Kupezki

submitted to obtain the degree of

Bachelor of Science (B.Sc.)

at

TH Köln University of Applied Sciences
Institute of Media and Imaging Technology

Course of Studies

Media Technology

First supervisor: Prof. Dr. Dirk Poggemann
TH Köln University of Applied Sciences

Second supervisor: B.Eng. Max Gäde
Image Engineering GmbH & Co. KG

Cologne, July 2024

Contact details: Julian KREMMING
Moltkestr. 108
50859 Köln
juliankremming@gmail.com

Mike KUPEZKI
Tenktererstr. 4
50679 Köln
mikekup@gmx.com

Prof. Dr. Dirk Poggemann
Cologne University of Applied Sciences
Institute of Media and Imaging Technology
Betzdorfer Str. 2
50679 Köln
dirk.poggemann@th-koeln.de

B.Eng. Max Gäde
Image Engineering GmbH & Co. KG
Im Gleisdreieck 5
50169 Kerpen
max.gaede@image-engineering.de

Abstract

Camera systems in vehicles are playing an increasingly important role in the development of autonomous systems. The movement of a vehicle can have an impact on image quality. In this thesis, the influence of movement on the effective resolution is analysed, taking various camera systems into account. The result is that a sensor with more but smaller pixel sizes has only minor advantages over a sensor with a fewer large pixels in the event of movement.

Key words: autonomous driving, modulation transfer function, spatial frequency response, slanted edge chart, ISO 12233, motion tracking, split pixel, effective resolution

Acknowledgement

We would like to thank Prof. Dr. Dirk Poggemann for making this untypical arrangement of a Bachelor thesis possible. In terms of guidance on research, feedback, background knowledge and support, we would like to express our thanks to Dr. Marzena Franek and Ulrich Seger from Robert Bosch GmbH. Thanks to Solectrix GmbH for providing hardware and support. We would like to thank Muhammad Atif (Sony Electronics) for his friendly and competent support with the Sony sensors. Special thanks to Max Gäde and the team of the Image Quality Lab & Innovation as well as Uwe Artmann as the CTO of Image Engineering GmbH & Co. KG for the emotional and technical support throughout the course of this thesis.

Contents

Abstract **I**

List of Tables **V**

List of Figures **VII**

1 Introduction **1**

2 Basics **2**
2.1 Camera Systems . . . 2
2.1.1 Image Signal Processing . . . 2
2.2 Motiontracking . . . 5
2.2.1 IMU . . . 5
2.2.2 Arduino . . . 6
2.2.3 Datalogger . . . 7
2.3 Development Board . . . 7
2.3.1 General . . . 7
2.3.2 SoftISP . . . 8
2.4 Charts . . . 8
2.4.1 MTF/SFR . . . 9
2.5 Hexapod . . . 10
2.6 Software . . . 11

3 State of the Art **12**

4 Measurements **14**
4.1 Preliminary Investigation . . . 14
4.2 Setup . . . 18
4.2.1 Motion Tracking . . . 18
4.2.2 IMU Programming . . . 19
4.2.3 Laboratory Environment . . . 22
4.2.4 Camera Systems . . . 23
4.2.5 Hexapod . . . 26
4.3 Data Collection Procedure . . . 27
4.3.1 Motion Data . . . 27
4.3.2 Image Data . . . 28
4.4 Data Analysis Procedure . . . 29
4.4.1 Template Finder . . . 29
4.4.2 Analyzer Settings . . . 32
4.4.3 Analyzer CMD . . . 33

5 Results **35**
5.1 IMX728 - Examination of the Lens . . . 35
5.2 SFR Analysis Comparison . . . 36

5.2.1 Stillstand Comparison of all Cameras 36
5.2.2 Comparison of Pitch Movement with all Cameras 37
5.2.3 Specification of the Worst Frames Based on SFR10 Values 39
5.2.4 Pitch Comparison - Zero Crossing Images (Sony IMX490) 39
5.3 SFR and Movement Correlation 42
5.3.1 Frequency Components of Movement Data 43
5.4 Lost Cargo 46

6 Conclusion **49**
6.1 Discussion 49
6.2 Suggestions for Future Research 50

Appendix **55**

Declaration of Authorship **56**

List of Tables

1 Camera systems [3][4][5] 2
2 Camera Register Descriptions - IMX490 23
3 Camera Register Descriptions - IMX728 26
4 Pivot Point for Different Sensors 27
5 Lost Cargo Calculation (ID 1, Stillstand) 47
6 Lost Cargo Calculation (ID 73, Full Braking from 7km/h) 47
7 List of Equipment 55

List of Figures

1 Visualisation of input and output levels (read out from Sony IMX623 camera registers) . . . 3
2 Bayer Filter and its typical responses [2, page 62] . . . 4
3 Subpixel Structure SP1 and SP2 [9] . . . 4
4 IMX490 Subpixel Channel [9] . . . 5
5 DFRobot I2C BMI160 6-Axis Inertial Motion Sensor [12] . . . 6
6 DFRduino UNO R3 [14] . . . 6
7 Data Logger Module [15] . . . 7
8 SXIVE Bundle LT by Solectrix with one Camera [16] . . . 8
9 Slanted Edge Chartlayout C_M 60 % . . . 9
10 Projection of data along the edge . . . 10
11 SFR Curve (iQ-Analyzer 6 Screenshot) . . . 10
12 Hexapod Features [23] . . . 11
13 onsemi Automotive Image Sensor Trends [24] . . . 12
14 IMU Specification . . . 14
15 Frequencycomponents Gyro-Z-Signal . . . 15
16 Gyroskop Z over Time . . . 16
17 Integrated Sinewave . . . 16
18 Connection Diagram (BMI160 - Arduino UNO) [27] . . . 18
19 IMU Installation in Test Car . . . 18
20 Close-up IMU Alignment . . . 18
21 Surface Conditions . . . 19
22 I2C Connection . . . 19
23 Device Manager, Bits/second . . . 19
24 IMU Calibration . . . 20
25 RTC and SD Card Initialisation . . . 20
26 Data Access in Main Loop . . . 21
27 Data Output on SD Card . . . 21
28 Termination Condition for Main Loop . . . 22
29 Laboratory Overview . . . 22
30 3D-printed camera housing (build by Bosch) . . . 23
31 Setup with camera facing charts . . . 23
32 Sony IMX728 Exposure Time Check with LED Panel . . . 24
33 Register Value Configuration . . . 25
34 Shell Skript Exposure Time - IMX728 . . . 26
35 Motiondataformat [28] . . . 27
36 Extract from formatted motion data . . . 28
37 Shell Script Rosbag to Raw Extraction . . . 28
38 Image Workflow . . . 29
39 Example Check Image with marked ROI - IMX623 . . . 30
40 OpenCV Slanted Edge Templates . . . 31
41 Example Output ROI File / iQ-Analyzer 6 Input ROI File for Free Edge Tool 32

42 Chat Layout Selection 32
43 ROI Selection 33
44 Comparison IMX728_a versus IMX728_b 36
45 Comparison Stillstand 36
46 Comparison Stillstand versus 5 Hz Pitch - IMX728_b 37
47 Comparison Stillstand versus 5 Hz Pitch - IMX623 38
48 Comparison Stillstand versus 5 Hz Pitch - IMX490 38
49 Comparison Performance Loss - 5 Hz Pitch 39
50 Sine 5 Hz 1,5 deg - Position of Worst Frames (Symbolic image) 39
51 Comparison Stillstand versus 1 Hz Pitch - IMX728_b 40
52 Comparison Stillstand versus 3 Hz Pitch - IMX490 41
53 Comparison Stillstand versus 5 Hz Pitch - IMX490 41
54 Comparison Performance Loss, Pitch - IMX490 42
55 Comparison Stillstand versus 5 Hz Yaw - IMX490 42
56 Correlation SFR10 and Gyroscope Data 43
57 Frequency Components Movement ID 73 44
58 Frequency Components Movement ID 78 44
59 Frequency Components Movement ID 83 45
60 Frequency Components Movement Stillstand 45
61 LostCargo SFR FOV 46
62 LostCargo Stillstand 48
63 LostCargo 73 48
64 LostCargo Stillstand vs 73 percent 48
65 Potential Charts for Future Research 51

1 Introduction

The increasing development of autonomous vehicles and their driver assistance systems requires ever higher quality and reliability in the camera technologies that are used. Camera systems must undergo further development in order to meet the increasing demands placed upon them. The capacity to identify distant objects on the road early enables autonomous systems or drivers to react to potential dangers with a greater degree of time and distance. This enables them to brake or swerve in a timely manner and avoiding a collision. As the number of pixels in camera systems increases with each generation, it is hypothesised that this will also improve the detection of objects, as the system could then be able to recognise them more quickly. The goal of this thesis is to investigate the influence of various movements on the effective resolution of camera systems and to develop a methodology for analysing these influences. To measure the influence of movement on the effective resolution, test charts were used and the cameras were moved during the recording of video sequences. Individual frames were extracted from these sequences and then analysed using a defined image processing procedure. For this investigation, three different camera systems were utilised. These systems were designed for automotive usage and differ by pixel count and size. Firstly, preliminary investigations were carried out to see what movements a vehicle actually makes and to what extent the movement of an object is reflected in the image. A measurement setup was then created in which the camera systems are moved in a controlled manner. The resulting images are analysed. As a movable platform, a hexapod was utilised in a laboratory environment. The main performance indicator for the effective resolution is based on the spatial frequency response which is derived from slanted edge test charts. In the long term, these studies could be used as a basis for developing better camera systems to make road traffic safer by preventing more accidents. This thesis was written in partnership with Robert Bosch GmbH and with technical support by Solectrix GmbH.

2 Basics

The following sections will provide a detailed explanation of the terminology and equipment employed in the measurements presented in this thesis.

2.1 Camera Systems

To assess the impact of movement on image quality, three camera systems with *Sony* CMOS sensors were utilised. The cameras find application in the automotive industry and are used for *Advanced Driver Assistance Systems* (ADAS)[1]. Unlike the older *Charge Coupled Device* (CCD) sensor, *Complementary Metal Oxide Semiconductor* (CMOS) type camera sensors use integrated amplifiers, noise-reduction and digitisation circuits on the sensor [2, page 144]. This lead to advantages for CMOS sensors, especially for automotive use, such as a lower power consumption while offering higher processing speeds at lower production costs [2, page 145]. Each of the systems under test has comparable horizontal field of views (HFOV), ranging from 55° on the Sony IMX728 to 65° on the Sony IMX490. What differentiates the systems is the pixel size and active pixel resolution. The Sony IMX728 has the highest resolution and smallest pixel size, while the Sony IMX623 offers the lowest resolution with the largest pixel size. Both, the Sony IMX490 and IMX623 provide a pixel size of 3 µm. This results in three cameras with three different resolutions (Tab. 1).

	LI-IMX728-9295-070H*	LI-IMX490-GMSL2-065H	LI-IMX623-GMSL2-060H
Image Size (Pixel)	3857 x 2177	2897 x 1877	1937 x 1553
Frame Rate	30 FPS	25 FPS	60 FPS
Pixel Size	2.1 µm	3 µm	3 µm
Optical Format	1/1.72"	1/1.55"	1/2.42"
FOV (horizontal)	55°(calculated)	65°	60°

*modified due to exceptionally poor performing lens (5.1, IMX728 - Examination of the Lens)

Table 1 Camera systems [3][4][5]

2.1.1 Image Signal Processing

Sensor Data Handling The concepts of *decompanding*, *pedestal* and *offset* are part of the processing of raw images from a camera system.

Decompanding refers to the operation of reversing the process of the companding function, which is applied during the capture of the image. This is done to handle a high dynamic range of image information in a compressed bit depth. Sensors can therefore produce an image with a higher dynamic range than originally possible with their bit rate. Decompanding restores the linear relationship between the brightness of the scene and the captured image data. The signal is divided into many linear segments. The relationship between the input and output levels is defined by the specification of the coordinates of knee points, where the x-coordinate is the input signal level and y-coordinate the output

signal level (Fig. 1). As the linearisation is realised by these segments, it is called *piecewise linear* (PWL). The PWL function uses the piecewise linear curves to compress the sensor's output levels [6].

The *pedestal* is an additional value added to the image signal that makes sure, the darkest parts of the image are represented with values above zero. This is done to prevent unwanted loss of image information.

The image's *offset* refers to the position of the data stream from which the visual information start. Before that, additional data is transferred, for example information about picture height and colour settings, also known as the header. This guarantees that the software responsible for displaying the image is aware of the starting point at which it should begin interpreting the data stream as individual pixels.

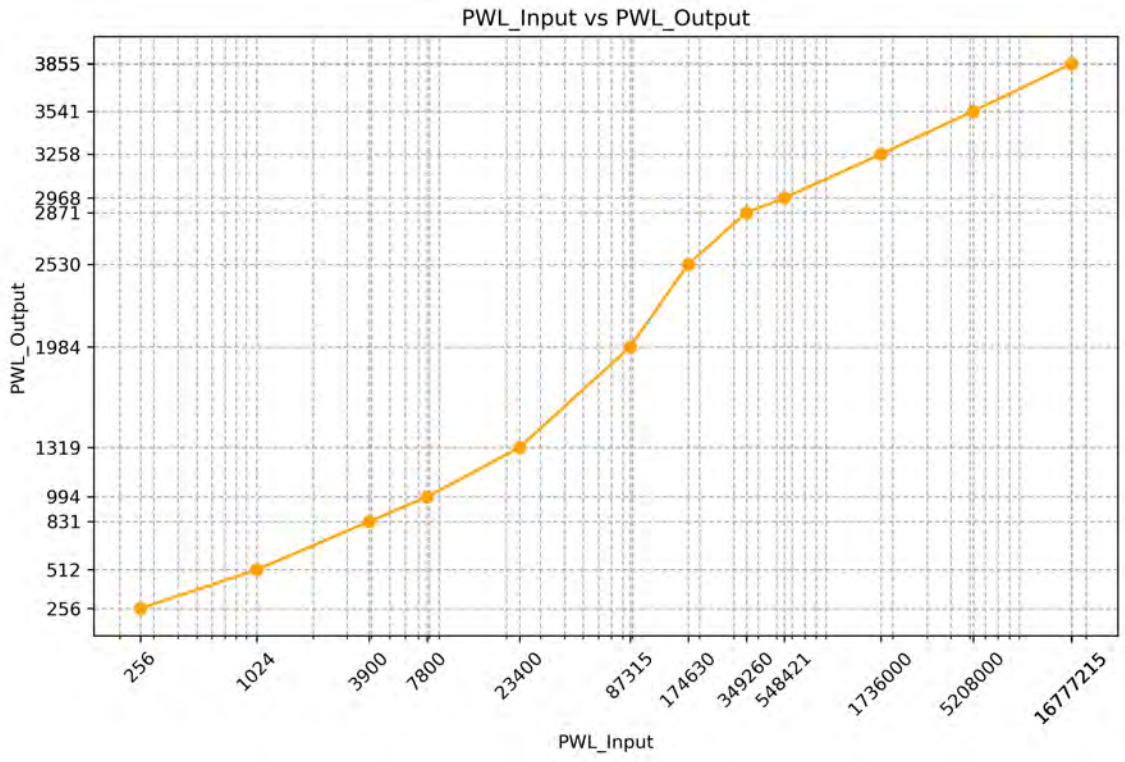


Figure 1 Visualisation of input and output levels (read out from Sony IMX623 camera registers)

Bayer Pattern Because an image sensor itself is "basically a monochrome sensor responding to light energies that are within its sensitive wavelength range" [2, page 62], a way to differentiate between colours is needed. A Bayer pattern is the most common arrangement of a colour filter array above the monochrome sensor. It is used in modern single chip digital image sensors to create colour images. It was invented in 1976 by Bryce E. Bayer and consists of a grid of filters over the sensor's photo diodes [7]. The marked *two by two grid* of filters consists of two green, one blue and one red filter (Fig. 2). These filters only permit light from one colour, blocking all the others. Due to the human eye's greater sensitivity to green light, covering 50 % of the grid with green filters helps to achieve a perceived higher resolution. In order to obtain a colour image, the colour information is interpolated from surrounding pixels in a process called *demosaicing* [2, page 62 ff].

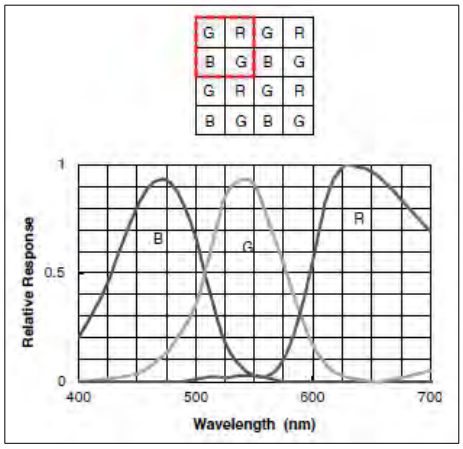


Figure 2 Bayer Filter and its typical responses [2, page 62]

Split Pixel Split pixel technology is a specific pixel design used in camera sensors to achieve a larger brightness range in the images. This allows a high dynamic range to be captured so that both bright and dark areas in an image can be reproduced in detail. Each pixel of the sensor is divided into two subpixels (Fig. 3) that have different sizes and thus different sensitivities [8]. All cameras used for the practical experiments of this thesis utilise the aforementioned split pixel technology.

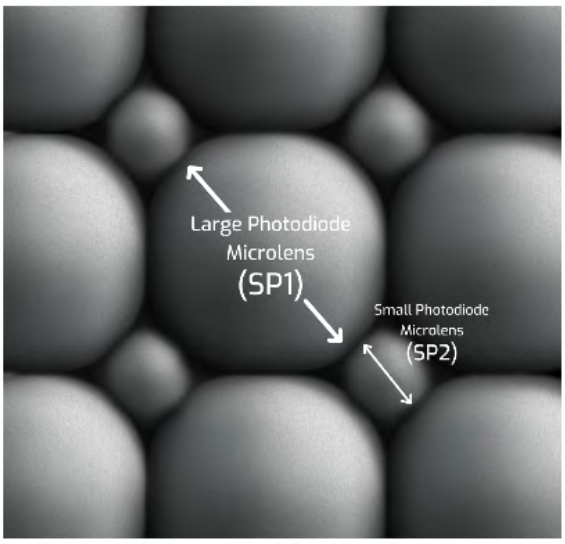


Figure 3 Subpixel Structure SP1 and SP2 [9]

Traditional HDR methods use a successive exposure of the pixels. This leads to motion artefacts after the images are merged. Split pixel sensor designs offer a solution to this, as they allow capturing HDR images within one exposure (Fig. 4). The images are merged using the respective manufacturer's algorithm of the device. Part of the sensors are two subpixels, each of which can switch between two gain levels. This technology is called *dual conversion gain* (DCG) and can switch between *low conversion gain* (LCG) and *high conversion gain* (HCG) [10].

- SP1_HCG (subpixel 1, high conversion gain, channel 0)
- SP1_LCG (subpixel 1, low conversion gain, channel 1)

- SP2_HCG (subpixel 2, high conversion gain, channel 2)
- SP2_LCG (subpixel 2, low conversion gain, channel 3)

High conversion gain implies that the subpixel enables a high sensitivity for incoming light, whereas low conversion gain has a low sensitivity. This allows details to be captured in dark areas as well as in bright areas, providing an effective solution for improving image quality [8].

Figure 4 IMX490 Subpixel Channel [9]

2.2 Motiontracking

Motion tracking is a technique for capturing the movement of objects in real time. To ensure a successful simulation within a laboratory, it is crucial to capture accurate and reliable motion data.

2.2.1 IMU

Motion tracking can be accomplished by using *inertial measurement units* (IMU). An IMU is an electronic component that is used in technical systems to measure movement data. This device is used in many different areas such as smartphones, drones, robots and also in vehicle technology. An IMU essentially has two different sensors: an accelerometer and a gyroscope.

The accelerometer measures the linear acceleration in the unit m/s^2 along the three spatial axes (x,y,z) of the object. The measurements provide information on how fast the object is moving in the respective direction.

The gyroscope indicates the rotational speed, which is measured in *degrees per second* (°/s). This determines the rotation of the object relative to the reference axis, which includes the movements *pitch*, *yaw* and *roll*.

One IMU that offers the six-axes movement recording described above, is the BMI160 from Bosch. The IMU is also offered in conjunction with DFRobot hardware (Fig. 5). In general, this reduces the development effort by providing a pre-configured interface alongside a base code. It is also compatible with an Arduino microcontroller (Fig. 6) and can be addressed via the *inter-integrated circuit interface* (I2C), which is an efficient

communication path [11, page 3]. In addition, for motion simulation in the laboratory, it is important to capture the data precisely during recording, which is made possible by the data recording rate of the IMU.

Figure 5 DFRobot I2C BMI160 6-Axis Inertial Motion Sensor [12]

2.2.2 Arduino

Arduino is a free platform for building systems that combine hardware and software. The platform includes hardware in the form of microcontrollers, as well as the *Arduino Integrated Development Environment* (Arduino IDE) software, which enables developers to create prototypes and projects. The software can be used to program and control the various sensors that can be connected to the microcontroller (Fig. 6). The Arduino IDE includes its own programming language, which is based on C/C++. The development environment supports a variety of extension packages and numerous libraries that can be downloaded. These include, for example, the *FastIMU* library, which provides the basis for precise and fast motion analysis of data with IMUs [13]. The connection to the PC is usually made via USB, which transfers the code to the microcontroller. The analysis and measurement results can also be displayed in real time in the output window as required. Depending on the requirements of the project, a specific Arduino board has to be defined within the development environment via the selected USB port.

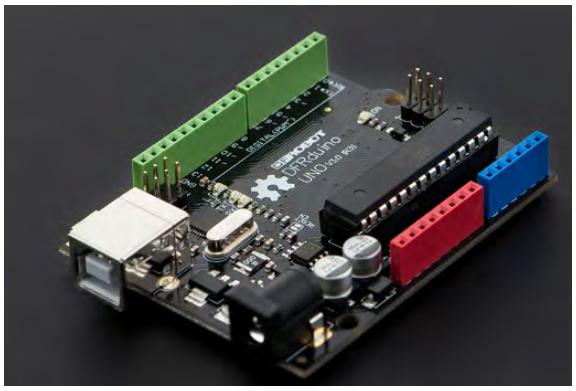


Figure 6 DFRduino UNO R3 [14]

2.2.3 Datalogger

To avoid limitations due to the transmission speed of a USB cable, a data logger (Fig. 7) can be integrated into the motion measurement system. Data loggers are modules used to record and store data over a predefined period. They connect directly to the microcontroller and are wired according to the respective instruction. This configuration ensures a shorter and more stable data transfer path, without being affected by cable length or potential interference between the controller and the PC. These systems also include a real-time clock [15], which continuously measures time and can be exported along with the recorded acceleration and rotational speed information.

Figure 7 Data Logger Module [15]

2.3 Development Board

The development board is an image processing system that consists of "the actual image processing software, a frame grabber board, hardware accelerators and a large number of apps and plugins" [16].

It therefore offers a way to capture raw images and video streams from multiple sensors/cameras to view and record them in real time using a graphical user interface.

2.3.1 General

The *SXIVE Bundle LT* (Fig. 8), produced by *Solectrix*, provides a flexible and fast development solution for automotive imaging systems. The system is based on the *NVIDIA Jetson AGX Orin* and the *proFRAME* frame grabber module [16]. The hardware combination offers a pre-configured, instantly operational system that enables the simultaneous use of two cameras. A significant advantage is the compact size of the system, which is ideal for in-car applications. The cameras are connected to the frame grabber using *Gigabit Multimedia Serial Link 2* (GMSL2) and can then be controlled within the Linux development environment using the integrated graphical user interface *SoftISP*.

Figure 8 SXIVE Bundle LT by Solectrix with one Camera [16]

2.3.2 SoftISP

The *SoftISP* software component is used to facilitate the processing and display of raw image data derived from image sensors. A range of adjustment options are offered to the user in the *SoftISP* settings, including gamma, noise reduction and colour correction. For a standard configuration within the application in a car, the settings are not adjusted. The exposure time and other parameters that are not set in *SoftISP* are configured via the camera registers. These are listed in the application notes of the respective manufacturer, which are not freely available. Each parameter is assigned a hexadecimal identifier, which can then be addressed in the console of the *SXIVE LT bundle*.

2.4 Charts

Slanted edge charts are a fundamental tool in the evaluation of digital imaging systems for assessing the image quality and spatial resolution. This type of chart is designed to measure the *Spatial Frequency Response* (SFR) of the imaging system and quantifies how the system reproduces detail at different spatial frequencies. The edges are slanted (Fig. 9) to minimise the influence of pixel-aliasing. As the edge crosses the pixels, it cannot be parallel to the pixel grid and is measured in different phases to photoelements on the image sensor [17]. The chosen chart layouts consist of a 10 % and a 60 % *Michelson Contrast*. The Michelson Contrast C_M is defined as:

$$C_M = \frac{\beta_{\max} - \beta_{\min}}{\beta_{\max} + \beta_{\min}} \tag{1}$$

with β as the reflection factor of the chart [18, page 184].

The chart's size A1066 is 1245 mm on the long side and 835 mm on the short side [19]. This was the largest locally producible size and provided large enough surface for the test environments in this thesis.

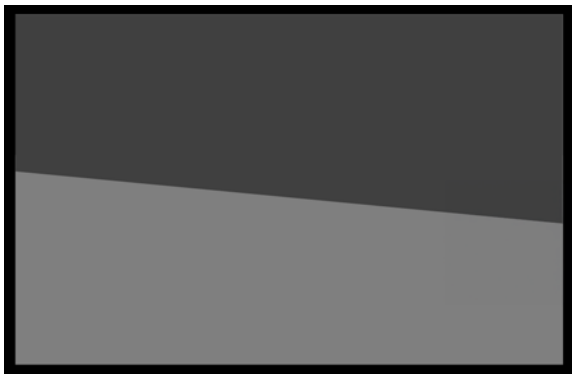


Figure 9 Slanted Edge Chartlayout C_M 60 %

2.4.1 MTF/SFR

The *modulation transfer function* (MTF) is a criteria for the evaluation of camera systems. It describes the loss of contrast in an image dependent on the spatial frequency. In an ideal world, the perfect system would have an MTF of 0.5 line pairs per pixel (lp/px). That would be the case, if the system transfers every detail without losing contrast. In reality, factors like pixel size, optical aberrations from the lens and movements impact the MTF. While the number of pixels on a sensor is often referred as the resolution of the camera system, these factors come into play and show the performance of the whole system. When using the *slanted edge charts*, the correct term to describe the resolution measurements would be the SFR, standing for *Spatial Frequency Response* [20]. With regard to an SFR10, this contrast is lost at a level of 10 %, or 50 % for an SFR50, respectively. Deriving an SFR from slanted edge charts for digital cameras has been specified in ISO 12233 [21]. A region of the chart is located and the data of each line of pixels is projected on the edge (Fig. 10). If an image with the marked region is analysed, a graph (Fig. 11) is obtained from which the SFR10 value can be read (in this case, approximately 0.15 lp/px).

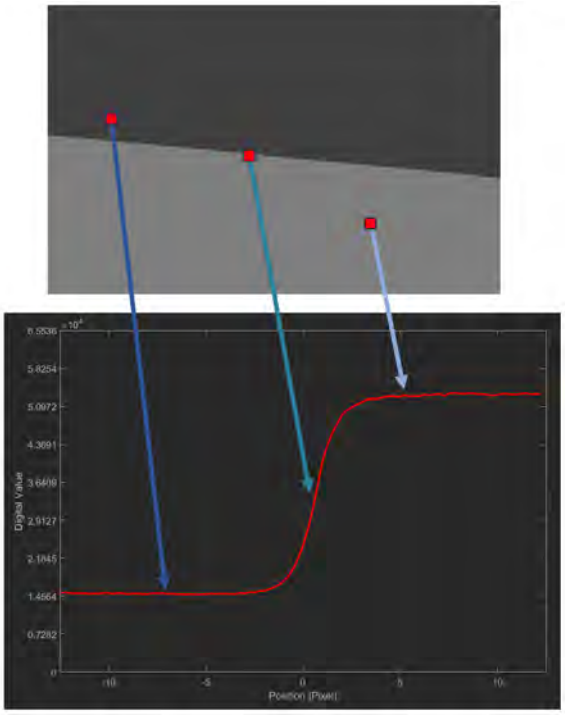


Figure 10 Projection of data along the edge

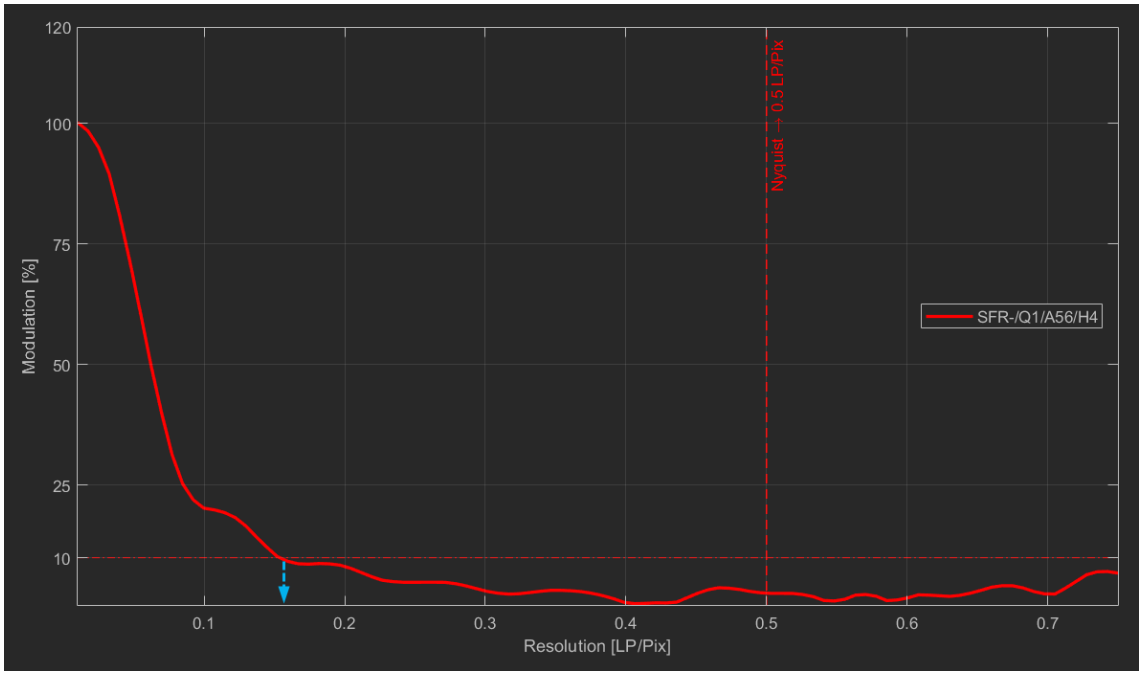


Figure 11 SFR Curve (iQ-Analyzer 6 Screenshot)

2.5 Hexapod

As a system, a hexapod can move its mounting plate and attached objects in space on six axes. Three axes are linear (x, y, z) and three axes are rotational (roll, pitch, yaw). These axes can be combined to generate movements. The hexapod *STEVE-6D* from *Image Engineering* is suitable for precisely simulating the recorded IMU movements within a laboratory. The control software allows specially formatted motion profiles to be loaded

into the system and to play them back. With its six degrees of freedom, it was designed for tests of optic image stabilisation of digital cameras [22]. It can move its roll and pitch axis by ±15° and its yaw axis by ±30° with a velocity of 600 mrad/s (Fig. 12).

Active axes	X, Y, Z, θ_X (roll), θ_Y (pitch), θ_Z (yaw)	
Travel range*	X, Y: Z: θ_X, θ_Y: θ_Z:	±50 mm ±25 mm ±15° ±30°
Single-actuator design resolution	0.5 µm	
Min. incremental motion	X, Y: Z: θ_X, θ_Y, θ_Z:	3 µm 1 µm 5 µrad
Backlash	X, Y: Z: θ_X, θ_Y: θ_Z:	3 µm 0.2 µm 20 µrad 30 µrad
Repeatability	X, Y: Z: θ_X, θ_Y: θ_Z:	±0.5 µm ±0.4 µm ±7 µrad ±12 µrad
Max. velocity	X, Y, Z: θ_X, θ_Y, θ_Z:	50 mm/s 600 mrad/s
Typ. velocity	X, Y, Z: θ_X, θ_Y, θ_Z:	30 mm/s 300 mrad/s

Figure 12 Hexapod Features [23]

2.6 Software

Different software tools were used to help with collecting data and analysing images. The following ones are the most essential: *Arduino IDE*, *Python*, *iQ-Analyzer 6* and *ImageJ*. The *Arduino IDE* was used to program and readout data of the BMI 160 IMU. Programmed code directly runs on the *Arduino UNO* and has access to the connected sensors and boards. In this thesis, *Python* is used to plot graphs, manage data and calculate measurements. *Python* is a high-level programming language known for readability and simplicity. To analyse the recorded frames with the slanted edge charts and get SFR measurements, the *iQ-Analyzer 6* by *Image Engineering* was used. In most cases, the *iQ-Analyzer* API was used via the command line interface. It is based on *MatLab*. To convert raw images to TIFF files, *ImageJ* was used. On the *Solectrix SXIVE Bundles*, the *SoftISP* along with the *sxpf* library was utilised.

3 State of the Art

Technological advancements in the automotive industry are evolving rapidly, especially in the field of advanced driver assistance systems. Camera technology is decisive for improving the driver experience and guaranteeing safety for all passengers. A significant trend in this area is the increasing use of high-resolution front-facing cameras. These cameras are fundamental for precisely identifying and interpreting data from the environment around the car, which enhances safety features like automated traffic sign or object/creature recognition. A number of key companies in the automotive industry, including *onsemi*, *Sony* and *OmniVision*, are contributing to this trend with the introduction of their latest camera sensors to the market.

The development of *onsemi* cameras (Fig. 13) is based on the fact that objects can be detected better thanks to the high resolution. This indicates that safety-critical decisions can be made more effectively. Alongside the increase in resolution, the pixel size is reduced to 2.1 µm. High dynamic range optimisation (Hyperlux) is used to compensate for the pixel size based on the light sensitivity [24].

Figure 13 onsemi Automotive Image Sensor Trends [24]

It can be seen that sensors below three megapixel with *LED flicker mitigation* (LFM) were still used in the years 2018 to 2021. These will then be completely discontinued after 2022 and replaced by 3 megapixel and 8.3 megapixel sensors.

A similar development can be seen in the latest sensor from *Sony*. At 17.42 megapixels, the *IMX735* has the highest pixel count in the industry. It also has a pixel size of 2.1 micrometers. The dynamic range priority mode provides a dynamic range of 130 dB, making it suitable for difficult illumination conditions like those at tunnel entrances or exits [25]. Due to limited availability, the high resolution sensor examined in this thesis, is the *Sony IMX728*, which provides 8.39 megapixels [1].

Omnivision's newest 2.1-micron sensor *OX08D10* delivers a high dynamic range feature known as *TheiaCel* technology. The eight megapixel sensor will enter mass production in the second half of 2024 and is therefore on the same stage as the other relevant companies within the automotive industry [26].

In summary, it can be observed that the newest automotive cameras have pixels that measure 2.1 micrometers in size and will have around eight megapixels of resolution. Compared to previous years, this is a large increase in resolution. The deficit of the pixel size

reduction is supposedly compensated by high dynamic range technology, which is individually configured by each manufacturer.

4 Measurements

In order to establish a measuring concept, it was necessary to undertake a preliminary investigation. This involved recording motions, setting up the laboratory and planing a process for analysing the images.

4.1 Preliminary Investigation

In the preliminary study of this thesis, investigations were made about which movements would affect the camera systems in a car with a camera mounted on the windscreen facing outwards. The main movements are acceleration and deceleration of the car, resulting in pitch movements. Turning in corners or avoiding obstacles causes the car to yaw but also to roll. Other influences come from the surface the vehicle is driving on, ranging from unpaved road sections over cobblestones to asphalt tiles (Fig. 21). An IMU (2.2.1) was needed to capture the vehicle's motion data, so an appropriate frequency for the sampling rate needs to be established. To gather information at a sufficiently high rate without making the setup too complex, bulky and expensive, a simple solution from DFRobot was recommended, claiming to provide a data output from 0,1 Hz to 200 Hz (Fig. 14). After setting up the Arduino code, the Datagrabber and SD card module, initial measurements were made to familiarise with the hardware and software. The IMU was mounted on the membrane of a speaker, playing sinus sounds at different frequencies. These frequencies should be recognisable in the recorded data. It turned out that the given IMU was only able to put out data at a rate of 10 to 15 Hz, which was confirmed by the supplier, claiming a mistake in the published datasheet. The *FastIMU* library emerged in the course of the troubleshooting process. Its documentation contained a list of compatible fast IMUs, on the basis of which a new IMU (BMI160) was ordered [13]. This resulted in the achievement of a sampling rate of approximately 100 Hz.

Figure 14 IMU Specification

To accurately estimate the size requirements for the test charts, calculations were made to determine what distances would result in what pixel sizes on sensors capturing a one metre object (eq. 2). For example, this object in 10 metres distance would appear as 425 pixels on the Sony IMX728. This was used to categorise the required size of the charts.

$$\text{Imagesize [mm]} = \frac{\text{Focallength [mm]} \times \text{Subjectsize [mm]}}{\text{Distance [mm]}} \tag{2}$$

The effect of motion blur depends on the exposure time used. To represent the worst case of the exposure time used in practice, 15 ms are used. For example, an image in 10

metres distance travelling orthogonally at speed of 10 km/h would approximately move 18 pixels on the sensor of the Sony IMX728, during one frame (eq. 3) (eq. 4).

$$\text{Distance on Sensor [mm]} = \frac{(\text{Velocity [m/s]} \times \text{Exposure Time [s]}) \times \text{Focallength [mm]}}{\text{Distance to Object [m]}} \tag{3}$$

$$\text{Distance on Sensor [Pixel]} = \frac{\text{Sensor Width [Pixel]} \times \text{Distance on Sensor [mm]}}{\text{Sensor Width [mm]}} \tag{4}$$

The process of applying a motion to the camera sensor had to be validated. Different frequencies and amplitudes were set on the STEVE-6D hexapod and the recorded IMU data was analysed. Applying a motion of 5 Hz and 2 degrees on the pitch axis on the hexapod were recorded by the IMU and needed to be validated.

Prior to calculating the signal, the sampling frequency needs to be calculated by averaging the time difference between two recordings (eq. 6). The z-axis data is split up into its frequency components (eq. 7) (eq. 8) , showing a clear peak at 5 Hz (Fig. 15). Plotting the z-axis data over time, an amplitude of approximately 31,6 degrees/second is measured (Fig. 16). Combining these two measurements and integrating a sine wave of 5 Hz with an amplitude of 31,6 degrees/second results in an amplitude of about 2 degrees (Fig. 17), matching the amplitude set in the STEVE software, thus validating the measurement system.

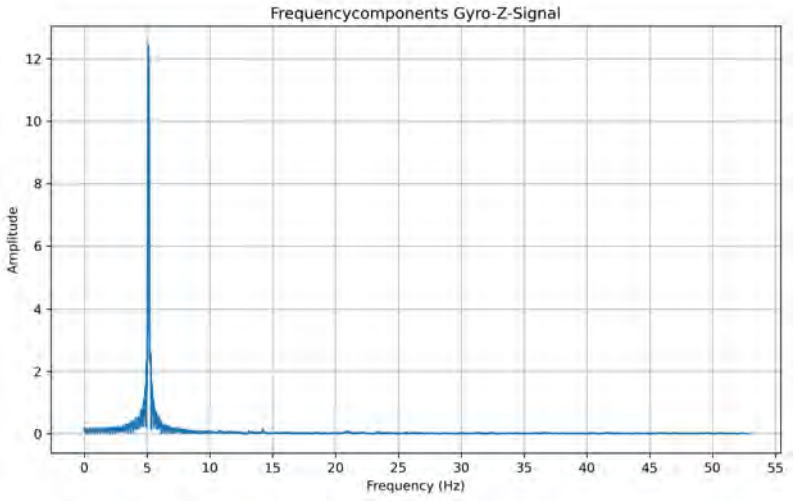


Figure 15 Frequencycomponents Gyro-Z-Signal

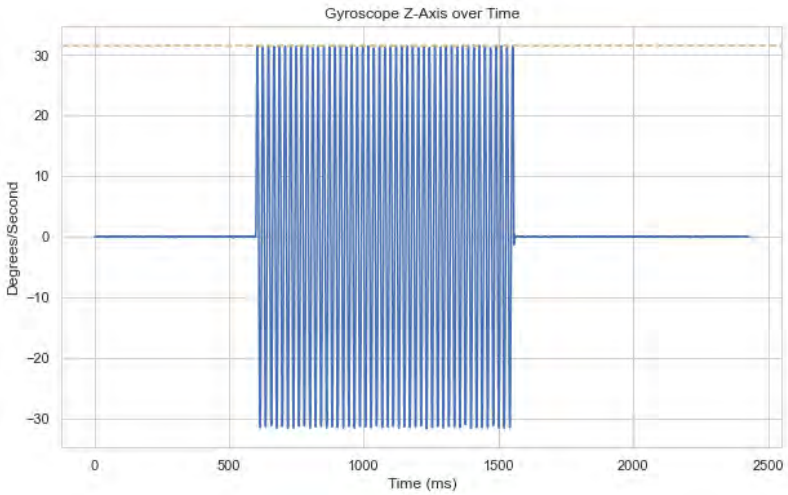


Figure 16 Gyroskop Z over Time

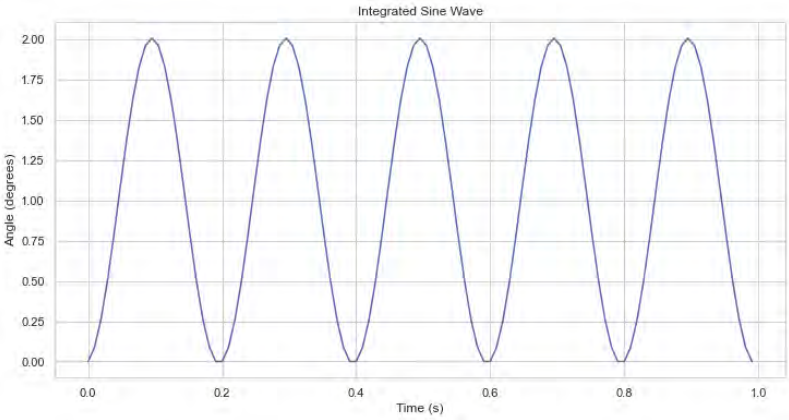


Figure 17 Integrated Sinewave

The following formulas where part of the validation process:

Time difference between successive measurements

$$\Delta t_i = t_{i+1} - t_i \tag{5}$$

where t_i is the time of the i-th measurement

Calculation of the average time difference:

$$\bar{\Delta t} = \frac{1}{N-1} \sum_{i=1}^{N-1} \Delta t_i \tag{6}$$

Fourier-Transformation

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k n / N} \tag{7}$$

Calculation of the frequency components for the Fourier Transformation:

$$f_k = \frac{k}{NT} \tag{8}$$

where T is the sampling period and N is the number of samples.

Parameters

$$\begin{aligned} A &= 31.6 \quad \text{(Amplitude)} \\ f &= 5\,\text{Hz} \quad \text{(Frequency)} \\ f_s &= 105\,\text{Hz} \quad \text{(Sampling Rate)} \\ T &= 1\,\text{s} \quad \text{(Duration)} \end{aligned}$$

Time Vector

$$t_i = \frac{i}{f_s}, \quad i = 0, 1, \ldots, f_s \cdot T - 1 \tag{9}$$

Sine Wave

$$y(t) = A \cdot \sin(2\pi f t) \tag{10}$$

Integrated Sine Wave (Angle over Time)

$$\theta(t) = \frac{1}{f_s} \sum_{i=0}^{n} y(t_i) \tag{11}$$

4.2 Setup

4.2.1 Motion Tracking

In preparation for the test drive, it was necessary to install the *BMI160* motion sensor in the car using the Arduino. The wiring between the Arduino and the sensor was connected according to the recommended I2C connection provided by *DFRobot* (Fig. 18).

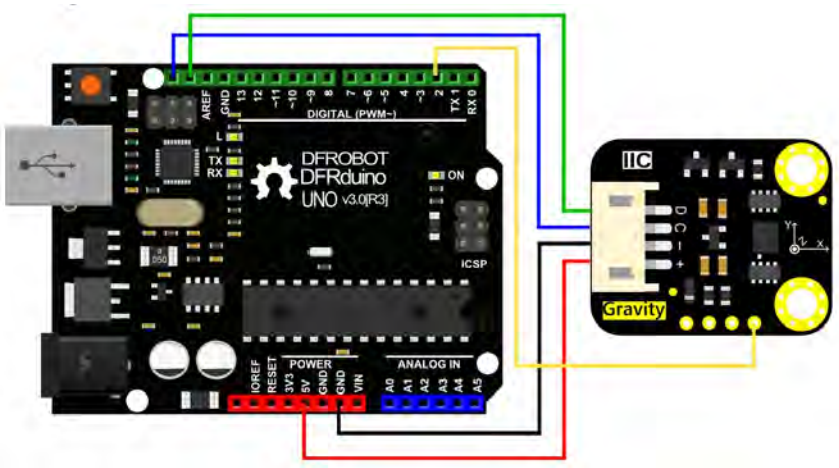


Figure 18 Connection Diagram (BMI160 - Arduino UNO) [27]

The camera mount, in which the sensor was firmly integrated, was mounted onto the dashboard of the car using glue pads (Figs. 19 and 20). It should be noted that the y- and z-axes are reversed due to the attachment of the IMU to the camera mount. This is taken into account when converting the data into a *STEVE*-formatted file (Appendix E.).

Figure 19 IMU Installation in Test Car

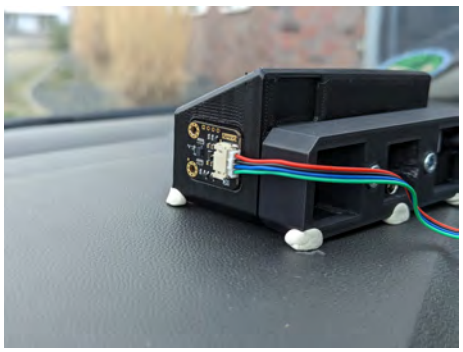


Figure 20 Close-up IMU Alignment

The test car was an unmodified *Renault Scenic 4. Generation ENERGY TCe 115*. Two people were needed to participate in the motion data recording. One person controlled the car while the second person monitored the Arduino software and the recording of the motion data. The test drive was carried out on a public road in Horrem. Containing motion data from 15 drives, the movement ID checklist (Appendix A.) includes predefined and documented surface conditions (Fig. 21), speed, and driving manoeuvres.

(a) Mixed Surface

(b) Cobblestone

(c) Asphalt

Figure 21 Surface Conditions

4.2.2 IMU Programming

The program (Appendix B.) is used to record and save the motion data of the test drive. The code runs inside the *Arduino IDE*, while the hardware components of the motion capture system consist of the *IMU*, the *datalogger* and the built-in *real-time clock* (RTC). Based on the *FastIMU* library, the code reads acceleration and gyroscope data as well as temperature from the built-in sensors and prints them to the console [13]. The influence of *temperature* is negligible for the measurements discussed in this work, it was only checked at the start of the test to ensure that everything was in operational conditions. The data is then stored on an SD card, which is integrated into the movement recording system via the data logger. In the following, the most important sections and functions of the program are explained.

The code is essentially divided into two parts: the setup of the sensors and the main execution loop. After setting the global variables, the setup begins, in which the communication with the IMU, as well as with a connected status LED, is established via an I2C protocol (Fig. 22). The data exchange frequency of 115200 bits per second (bps) is also defined, which must be synchronised with the controlling device, in this case a laptop. This is done via the device manager (Fig. 23).

```
Wire.begin();
pinMode(2, INPUT_PULLUP);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, INPUT);
Wire.setClock(400000); //400 kHz I2C-Takt
Serial.begin(115200);
```

Figure 22 I2C Connection

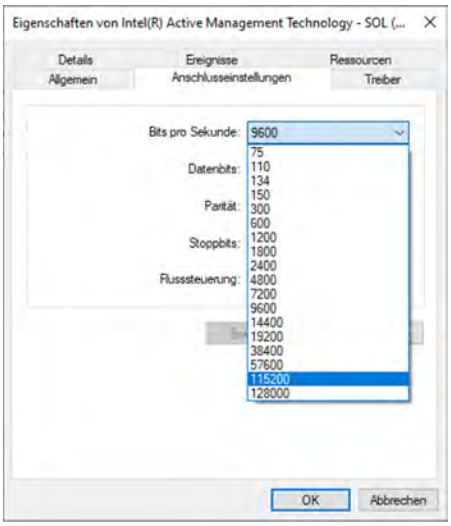


Figure 23 Device Manager, Bits/second

As soon as the *PERFORM_CALIBRATION* is defined, the sensors of the IMU are calibrated (Fig. 24). This includes the magnetometer (not fitted in the BMI160), gyroscope and accelerometer. If the calibration was successful, it is displayed in the Arduino console.

A small delay has also been implemented so that the user can follow the status of the calibration on the console (Fig. 24).

```
#ifdef PERFORM_CALIBRATION
  Serial.println("FastIMU calibration & data example");
  if (IMU.hasMagnetometer()) {
    delay(10);
    Serial.println("Move IMU in figure 8 pattern until done.");
    delay(30);
    IMU.calibrateMag(&calib);
    Serial.println("Magnetic calibration done!");
  }
  else {
    delay(50);
  }
```

Figure 24 IMU Calibration

Once the IMU has been successfully calibrated, the RTC and SD card can be initialised (Fig. 25). If the RTC or SD card are not recognised by the system, messages are displayed on the console. In the preliminary study (4.1), it proved advantageous to integrate a status LED into the system in addition to the text message. It is connected to digital port four (Fig. 25). As soon as data is no longer being exported to the SD card, the signal is set to *LOW* and the LED turns off.

```
if (!rtc.begin()) {
  Serial.println("Konnte RTC nicht finden. Überprüfe die Verbindung!");
  while (1);
}
if (!SD.begin(10, SPI_FULL_SPEED)) {
    Serial.println("Initialisierung der SD-Karte fehlgeschlagen!");
    return;
    digitalWrite(4, LOW);
}
```

Figure 25 RTC and SD Card Initialisation

The main program cycle is executed continuously in the loop. This takes place at a speed of approximately 106 Hz, which was determined during the preliminary study. At the beginning of the loop, the current time stamp of the RTC is read (Fig. 26). Acceleration and gyroscope data from the IMU are then accessed.

```
DateTime now = rtc.now();
IMU.update();
IMU.getAccel(&accelData);
IMU.getGyro(&gyroData);
```

Figure 26 Data Access in Main Loop

In the next part of the code, all data is written to the SD card (dataFile). Unfortunately the RTC is unable to output data with greater precision than seconds. For the speed and gyroscope data, the x-, y- and z-axes are saved into a buffer (Fig. 27). The data is separated by a tabulator and written to a CSV-file.

```
dataFile.print(now.hour(), DEC);
dataFile.print(":");
dataFile.print(now.minute(), DEC);
dataFile.print(":");
dataFile.print(now.second(), DEC);
dataFile.print("\t");
dataFile.print(accelData.accelX);
dataFile.print("\t");
dataFile.print(accelData.accelY);
dataFile.print("\t");
dataFile.print(accelData.accelZ - 1);
dataFile.print("\t");
dataFile.print(gyroData.gyroX);
dataFile.print("\t");
dataFile.print(gyroData.gyroY);
dataFile.print("\t");
dataFile.print(gyroData.gyroZ);
dataFile.println();
```

Figure 27 Data Output on SD Card

The termination condition to end the loop and close the data export was also integrated into the loop (Fig. 28). For this purpose, a cable was plugged into digital port 2, which is not connected on the other side. It is constantly checked whether the signal from port two is *LOW* or *HIGH*. As soon as the free end of the cable touches the metal frame of the data logger, the data transmission is terminated as the signal is grounded. This was practical and easy to implement as the cable was permanently within reach for the person monitoring the *Arduino software* to stop the program. In addition to the *SD card initialisation LED*, this is visually indicated by a second status LED on the *Arduino UNO*. Even if the query is called repeatedly in the loop, the output frequency was the same with and without the

condition. After the loop is terminated, the data is then saved and the SD card can be removed.

```
if (digitalRead(2) == LOW) {
    Serial.println("Datei geschlossen. Programm wird beendet.");
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    dataFile.close();
    Serial.println("Datei ist gespeichert");
    while (1);
}
```

Figure 28 Termination Condition for Main Loop

4.2.3 Laboratory Environment

Because the goal of the experiments in this thesis is to get as close as possible to real world conditions, a large space in the laboratory had to be found to accommodate the measurement setup. The dimensions needed were about 15 metres by 3 metres. The setup (Fig. 29) for recording the videos consisted of the two test charts, one mounted in an *iQ-Chartmount* and one turned 90 degrees standing on the floor (Fig. 31). These charts were illuminated with two *Aladdin Lights AMS-200BTD*, set to produce 300 lux on chart with 90% uniformity. In 10 metres distance from the chart, the hexapod (*STEVE-6D*) was positioned, with the cameras mounted on top of it, facing the chart.

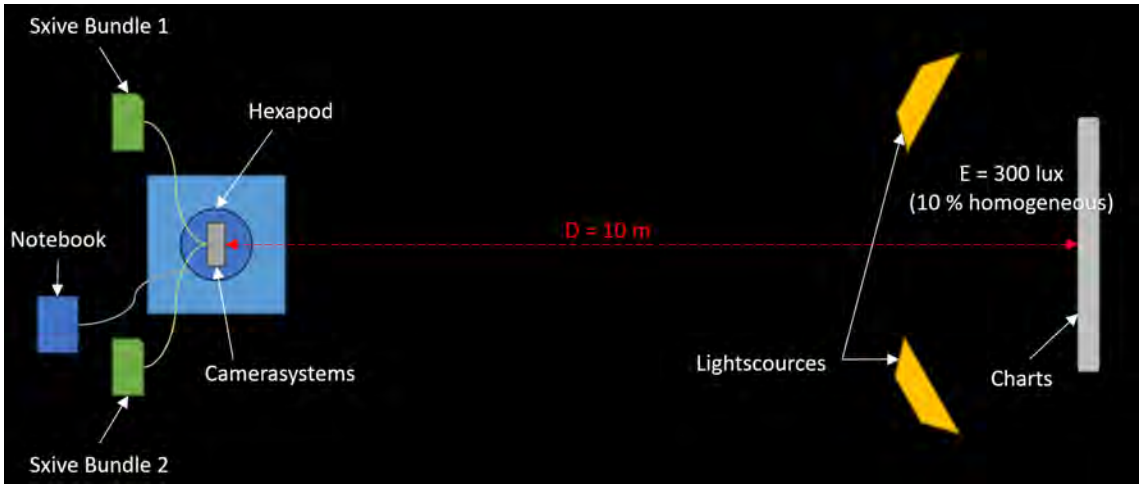


Figure 29 Laboratory Overview

The cameras were enclosed in the 3D-printed housing and stuck to the mounting plate of the hexapod with glue pads (Fig. 30), giving the cameras and housing no room to shake. The housing was developed and printed by *Bosch* specifically for the use in this thesis. Once the camera systems had been focused, the lens threads were fixed with *Teflon* tape. This serves to prevent any loss of sharpness due to accidental defocussing of the lens.

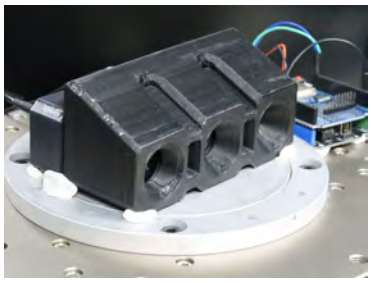


Figure 30 3D-printed camera housing (build by Bosch)

Figure 31 Setup with camera facing charts

Three computers were used to control the test setup. Two *Solectrix SXIVE Bundles* (2.3) each control one camera system. One bundle controlled the Sony IMX490, the other one the Sony IMX623 or Sony IMX728. The bundles each required a monitor, mouse and keyboard and a high-speed solid-state drive (SSD) to record onto. The *STEVE-6D* software was executed on an additional standard *Windows* computer.

4.2.4 Camera Systems

The cameras were used in default mode for the laboratory recordings. No changes were made within the *SoftISP* unless specified otherwise. In order to push the cameras to a challenging limit, an exposure time of 15 ms was selected for the images. As no trivial software solution is offered for this, camera registers have to be set manually. The registers were set with the help of the application notes of the respective camera sensor. These contain information on how to control various parameters in order to change them to a desired value. With the Sony IMX728 and IMX623, the required exposure times can be set one at a time in milliseconds as hexadecimal values within the respective registers. In order to utilise the Sony IMX490, a conversion to lines must be made beforehand, based on two given formulas from the application notes (eq. 14) (eq. 15). First, the exposure time process of the IMX490 is considered. The following parameters are relevant for the formulas:

Register Name	**Description**
INTG_TIME_SP1	Exposure time for SP1 in lines
REG_INTG_TIME_FINE_SP1	Compensation of exposure time error due to A/D conversion
INTG_TIME_SP2	Exposure time for SP2 in lines
REG_INTG_TIME_FINE_SP2	Compensation of exposure time error due to A/D conversion
FMAX	Number of extended frames (in case of long exposures)
VMAX	Number of lines in vertical sync
HMAX	Number of lines in horizontal sync

Table 2 Camera Register Descriptions - IMX490

Number of SP1 exposure time:

$$(\text{VMAX} \times \text{FMAX} + \text{INTG_TIME_SP1}) \times \frac{1}{\text{Frame rate} \times \text{VMAX}} \tag{12}$$

Number of SP2 exposure time:

$$(\text{VMAX} \times \text{FMAX} + \text{INTG_TIME_SP2} + \frac{\text{REG_INTG_TIME_FINE_SP2}}{\text{HMAX}}) \times \frac{1}{\text{Frame rate} \times \text{VMAX}} \tag{13}$$

A subpixel sensor structure (2.1.1, Split Pixel) is integrated within the IMX490, which consists of a large subpixel (SP1) and a small subpixel (SP2). The exposure value should be set the same for both sub-pixels so that the desired exposure time of 15 milliseconds can be achieved. To do this, the variables *INTG_ TIME_ SP1* (first subpixel) and *INTG_ TIME_ SP2* (second subpixel) must be calculated. *REG_ INTG_ TIME_ FINE* for both subpixels and *FMAX* can be neglected and set to zero before calculating the results. The reason for this is that a long exposure is not carried out in the experiment. Furthermore compensation for an exposure time error by means of A/D conversion is not necessary, as modern cameras make this adjustment automatically. This was confirmed by checking the exposure time. It was accomplished by utilising an Image Engineering LED-Panel in frame and counting the LEDs, that light up during the exposure. As each LED light up for one millisecond, the amount of LEDs is equal to the exposure time (Fig. 32). As a result, the formula for both subpixels can be rearranged as follows:

$$\text{INTG_TIME_SP1} = \text{Frame rate} \times \text{SP1_EXPOSURE} \times \text{VMAX} \tag{14}$$

$$\text{INTG_TIME_SP2} = \text{Frame rate} \times \text{SP2_EXPOSURE} \times \text{VMAX} \tag{15}$$

Figure 32 Sony IMX728 Exposure Time Check with LED Panel

For each of the parameters (Tab. 2) listed in there is an associated register which can be accessed via the command line of the *SXIVE bundle*. The register can either be read out

or a value can be set for it. By reading out the values, the following entries were obtained to complete the formulas:

$$VMAX = 1876 \quad \text{(Read out)}$$
$$Framerate = 25\,\text{FPS} \quad \text{(Read out)}$$
$$INTG_TIME_SP = 0,015\,\text{s} \quad \text{(Specified)}$$

Inserting the specified and read variables into the formula, results in the value *703.5* lines, which is rounded up to *704* lines. In hexadecimal notation, this equals *0x02 0xC0*. To initialise the IMX490, a *JSON* file is created that references two configuration files. The *JSON* file to be invoked is called via the console input of the SXIVE bundle:

SoftISP -c config_15ms.json proframe3 gui –channel 0

The two configuration files access all existing registers with their assigned values. To set the exposure time to 15 milliseconds when the camera is started, the calculated values for *INTG_TIME_SP1* and *INTG_TIME_SP2* must be changed manually in these files. Both registers were searched for and the old value was replaced with the new one. The hexadecimal values must be inserted in little-endian format, starting from the bottom and moving upwards (Fig. 33).

```
ARD_SENSOR 3 0x34 0xA1 0x00
ARD_SENSOR 3 0x34 0xA2 0x00
ARD_SENSOR 3 0x34 0xA3 0x00
ARD_SENSOR 3 0x36 0xC0 0xC0 #INTG_TIME_SP1
ARD_SENSOR 3 0x36 0xC1 0x02
ARD_SENSOR 3 0x36 0xC2 0x00
ARD_SENSOR 3 0x36 0xC4 0xC0 #INTG_TIME_SP2
ARD_SENSOR 3 0x36 0xC5 0x02
ARD_SENSOR 3 0x36 0xC6 0x00
ARD_SENSOR 3 0x36 0xC8 0x3C
ARD_SENSOR 3 0x36 0xC9 0x00
```

Figure 33 Register Value Configuration

For the IMX623 and IMX728, no calculation of the exposure time values is necessary, but more steps are required overall during preparation to achieve the desired exposure time. The unit of the values is not defined in *lines* but in *microseconds*. Different registers must be set one after the other in a specific sequence. This is done with the help of a call via in the command line:

sxpfi2c 0 0 0x36 1 0x98 0xAC 0x3

In the command above port 0 addresses the device ID 0x36. Here, a 1-byte register unit is transferred to the camera register 0x98 0xAC, whereby the value is defined as 0x3. To avoid having to enter the command manually in the command line every time the camera is rebooted, a shell script is utilised (Fig. 34). The script executes the respective commands one after the other in the console on the *Solectrix SXIVE* bundle.

Register Name	Description
AEMODE	This register is used to set the exposure control method
AWBMODE	This register is used to select the control method of white balance
FME_SHTVAL	This register is used to set the exposure time for SP1
FME_SHTVAL_UNIT	This register is used to set the unit for FME_SHTVAL
FME_SHTVAL_S1	This register is used to set the exposure time for SP2
FME_SHTVAL_S1_UNIT	This register is used to set the unit for FME_SHTVAL_S1
FME_SHTVAL_S2	This register is used to set the exposure time for SP1VS
FME_SHTVAL_S2_UNIT	This register is used to set the unit for the FME_SHTVAL_S2 register

Table 3 Camera Register Descriptions - IMX728

```
1  gnome-terminal -- bash -c "sxpfi2c 0 0 0x36 1 0x98 0xAC 0x3 \n
2  sxpfi2c 0 0 0x36 1 0xA2 0x48 0x3 \n
3  sxpfi2c 0 0 0x36 4 0x98 0xDC 0x98 0x3A \n
4  sxpfi2c 0 0 0x36 1 0x98 0xE0 0x3 \n
5  sxpfi2c 0 0 0x36 4 0x98 0xE4 0x98 0x3A \n
6  sxpfi2c 0 0 0x36 1 0x98 0xE8 0x3 \n
7  sxpfi2c 0 0 0x36 4 0x98 0xEC 0x0 \n
8  sxpfi2c 0 0 0x36 4 0x98 0xF0 0x3 \n "
```

Figure 34 Shell Skript Exposure Time - IMX728

First, the automatic exposure and white balance modes must be switched off (line 1-2). Then the exposure times including their units are defined for the first subpixel *(FME_SHTVAL)* and for the second subpixel *(FME_SHTVAL_S1)* (line 3-6). The only difference between the IMX623 and IMX728 is the last two register commands (line 7-8). With the IMX728 sensor, it is possible to set a short additional exposure for the first subpixel (SP1). The register name for this is *FME_SHTVAL_S2*, which is set to zero.

In summary, there are two different workflows for starting the cameras. For the IMX490, the configured JSON file with the previously prepared configuration files is loaded into the command line to make the camera ready for operation. The IMX728 and the IMX623 are both first initialised with the given standard JSON file from *Solectrix* and then the shell script with the desired exposure time is executed while the camera is running.

4.2.5 Hexapod

The hexapod is mounted on top of a movable construction, containing its driver and power supply. The driver needed to be connected to the notebook through an Ethernet cable. After connecting the power supply unit to the power supply and switching it on, the boot process had to be waited for, which is signalled by beeping. Only after that the software could be started.

4.3 Data Collection Procedure

4.3.1 Motion Data

A series of different movements was played through to get comparable data from the three camera systems. The cameras stayed next to each other at their position in the 3D-printed housing (Fig. 30). Before each movement, the hexapod was brought into the home position so the same starting position was used each time. In the *STEVE-6D* software, the frequencies and amplitudes for the homogeneous movements were defined. After adjusting the pivot point (Tab. 4), each individually for the camera system under test, the parameters for the movements were set. The individual pivot points were needed to have comparable motion artefacts on each camera, because every camera system was exited with the same movements. The axes x, y and z as well as roll, pitch and yaw were cycled through. The parameters according to the movement ID list (Appendix A.) were dialled in. The motion sequence was always started with a short delay to the video recording to get a reference image without movement.

	IMX623	**IMX728**	**IMX490**
X	0	0	0
Y	20	59	-19
Z	50	50	50

Table 4 Pivot Point for Different Sensors

To be able to read the recorded motion data from the IMU (4.2.1), it must be in the correct format for the *STEVE-6D* software to replicate the movements. The data was formatted according to the software's manual (Fig. 35) and example data using a *Python* script (Appendix E.). As already found in the preliminary investigation, the Arduino's RTC only delivered full seconds. The required milliseconds were interpolated on the basis of the amount of data entries per second. The unit of the gyroscope data must to be converted from degrees per second to radiant per second. Because the acceleration data as well as movements around the yaw axis reached values exceeding the capabilities of the hexapod, the focus was on the remaining gyroscope movements and the acceleration data was set to zero. The input of the script was the CSV-file from the Arduino code and the output was the formatted TXT-file for the *STEVE-6D* software (Fig. 36).

File definition for UVW from the spatial sensor:

HH:MM:SS.ZZZZ	AccX[g]	AccY[g]	AccZ[g]	GyrX[rad/s]	GyrX[rad/s]	GyrX[rad/s]	Mag[µt]	Mag[µt]	Mag[µt]

Use tab as the separator.

Figure 35 Motiondataformat [28]

966	15:40:36.466	0	0	0	0.074525591	0.024609153	-0.19914215300000002	0	0	0		
967	15:40:36.475	0	0	0	0.069289601	0.034208468	-0.197920422	0	0	0		
968	15:40:36.485	0	0	0	0.079936114	0.018151432000000002	-0.20018935100000002	0	0	0		
969	15:40:36.495	0	0	0	0.077841718	0.020245828	-0.20018935100000002	0	0	0		
970	15:40:36.504	0	0	0	0.071383997	0.03839726000000001	-0.20018935100000002	0	0	0		
971	15:40:36.514	0	0	0	0.046949377	0.021467559	-0.19914215300000002	0	0	0		
972	15:40:36.524	0	0	0	0.042586052	0.03211407200000001	-0.19914215300000002	0	0	0		
973	15:40:36.533	0	0	0	0.020245828	0.05654869200000001	-0.19582602600000001	0	0	0		
974	15:40:36.543	0	0	0	-0.002094396	0.03839726000000001	-0.19477882800000001	0	0	0		
975	15:40:36.553	0	0	0	0.009599315000000002	0.021467559	-0.19582602600000001	0	0	0		
976	15:40:36.563	0	0	0	-0.004188792	0.019198630000000005	-0.19163723400000002	0	0	0		
977	15:40:36.572	0	0	0	-0.026703549000000004	0.008552117	-0.18849564000000002	0	0	0		
978	15:40:36.582	0	0	0	-0.031939539	0.023561955000000002	-0.19059003600000002	0	0	0		
979	15:40:36.592	0	0	0	-0.03839726000000001	0.039444458	-0.19059003600000002	0	0	0		
980	15:40:36.601	0	0	0	-0.057595890000000004	-0.02443462	-0.19163723400000002	0	0	0		
981	15:40:36.611	0	0	0	-0.073478393	0.023561955000000002	-0.19163723400000002	0	0	0		
982	15:40:36.621	0	0	0	-0.065973474	0.028797945000000002	-0.19373163000000002	0	0	0		
983	15:40:36.631	0	0	0	-0.074525591	-0.026529016000000002	-0.196873224	0	0	0		

Figure 36 Extract from formatted motion data

4.3.2 Image Data

A number of steps are necessary to obtain analysis results from the video recording (Fig. 38). The procedure of capturing image data started with the initialisation of the camera system. For the *SXIVE Bundle* connected with the cameras, the systems needed to be configured on GMSL2 ports with even numbers; in this case it is zero or two. The *SoftISP* GUI to control the cameras was started with the terminal command below:

SoftISP -c IMX728 proframe3 gui –channel 0

It was necessary to open the terminal in the folder where the recording was to be stored. After a successful start, the next step was to configure the camera system's settings by executing a shell script (Fig. 34), which sets the registers to enable the 15 ms exposure time. This was confirmed with the *Image Engineering LED Panel* (4.2.4).

The preferred duration of the video recording was then set to five seconds for the homogeneous movements and five seconds longer than the recorded movement data to compensate for the initialisation of the hexapod. The recorded video sequences were stored in the *rosbag* format. To extract the frames, another shell script (Fig. 37) was needed. This used the *Recorder* function, and in conjunction with a for-loop, iterated over the specified folders and files and output them as parsed raw data. By calling the *Recorder* with *-p*, the data is interpreted as *parsed raw*. For this, *Solectrix* took the original *Sony* header, unpacked it and sorted it. This made the files easier to process. To get all the data contained in the *rosbag* file, a modified *config* file was called in the script. Otherwise, frames could be dropped depending on the performance of the development board. This modification forced the system to extract all recorded frames. Shorter computing time is exchanged with reliable frame extraction.

```
for i in {0..3}
do

    cd /media/orin/Lab_1TB/IMX728/${i}/RAW/
    #touch ${i}.txt
    Recorder -c '/usr/etc/SoftISP/IMX728/config.json' rosbag -f '/media/orin/Lab_1TB/NewIMX728Test/'${i}'/'${i}'.bag' -p
    sleep 20

done
```

Figure 37 Shell Script Rosbag to Raw Extraction

Until this point, the data was processed on the *Solectrix SXIVE Bundles*. The following

steps were done on regular Windows computers. The raw data for the Sony IMX623 and IMX728 was companded and needed to be linearised by decompanding them (2.1.1). This was achieved with a *Python* script. The values for the decompanding steps were read out from the sensor's register, converted to decimal numbers and inserted into the script. The now decompanded raw files needed one more conversion to be readable by the analysing software. To convert the decompanded raw data to TIFF images, an *ImageJ* macro was utilised, which needed the camera specific resolution, bit depth, endianess (byte order) and offset.

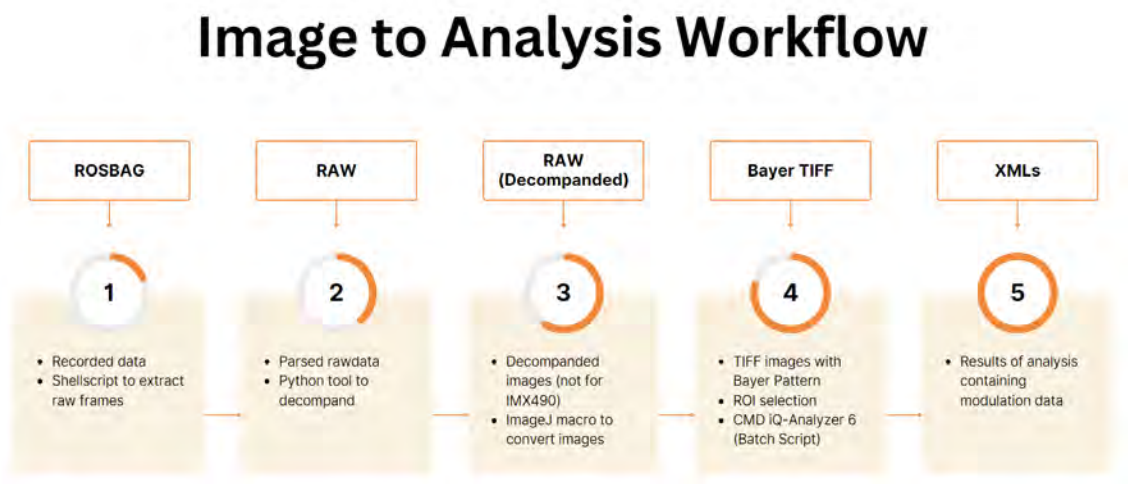


Figure 38 Image Workflow

4.4 Data Analysis Procedure

4.4.1 Template Finder

The following program automates the detection of predefined *regions of interest* (ROI) in the captured TIFF images using template matching (Fig. 39). This is done by using the *OpenCV* image processing library, which is imported into *Python*. An individual ROI file was needed for each TIFF image, as constant movements of the hexapod are carried out. Therefore the chart position in the image is changing frame-by-frame. The program reads in previously defined template images and searches for them within the recorded test laboratory images (list. 1). As soon as a template has been successfully found, the coordinates of the recognised ROIs are saved in text files. These are formatted in such a way that they can be read into the *iQ-Analyzer 6* for a position specification. The most important functions of the program are explained in more detail below.

Figure 39 Example Check Image with marked ROI - IMX623

```
import cv2 as cv
import numpy as np
import os
import glob

base_image_folder_path = '[...]/STEVE-Hauptordner/IMX623/'
base_output_folder_path = '[...]/STEVE-Hauptordner/IMX623/'
template_path1 = '[...]/ROITemplate/IMX623_High_Contrast_template1.tif'
template_path2 = '[...]/ROITemplate/IMX623_High_Contrast_template2.tif'
assert os.path.exists(template_path1), "First template could not be found."
assert os.path.exists(template_path2), "Second template could not be found."
template1 = cv.imread(template_path1, cv.IMREAD_GRAYSCALE)
template2 = cv.imread(template_path2, cv.IMREAD_GRAYSCALE)
w1, h1 = template1.shape[::-1]
w2, h2 = template2.shape[::-1]
methods = ['cv.TM_CCOEFF', 'cv.TM_CCOEFF_NORMED', 'cv.TM_CCORR', 'cv.
    TM_CCORR_NORMED', 'cv.TM_SQDIFF', 'cv.TM_SQDIFF_NORMED']
def apply_template_matching(img_path, output_folder_path):
    img = cv.imread(img_path, cv.IMREAD_GRAYSCALE)
    img2 = img.copy()
    img_basename = os.path.basename(img_path)
    output_text_filename = f"ROI_{img_basename.replace('.tif', '.txt')}"
    output_text_path= os.path.join(output_folder_path, output_text_filename)
    roi_details = []
    for meth in methods:
        img = img2.copy()
        method = eval(meth)
        for template, width, height in [(template1, w1, h1), (template2, w2,
            h2)]:
            res = cv.matchTemplate(img, template, method)
            if method in [cv.TM_SQDIFF, cv.TM_SQDIFF_NORMED]:
                top_left = cv.minMaxLoc(res)[2]
            else:
                top_left = cv.minMaxLoc(res)[3]
            if meth == 'cv.TM_SQDIFF_NORMED':
                roi_details.append(f"{len(roi_details)+1}\t{top_left[0]}\t{
                    top_left[1]}\t{width}\t{height}")
    with open(output_text_path, 'w') as file:
```

```
36          img_path_for_file = img_path.replace('/', '\\')
37          file.write(f"FileName:\t{img_path_for_file}\n")
38          file.write("UserComment:\t-\n")
39          img_height, img_width = img.shape[:2]          file.write(f"
                ImageHeight:\t{img_height}\tImageWidth:\t{img_width}\n")
                file.write(f"numOfROI:\t{len(roi_details)}\n\n")
40          for detail in roi_details:
41              file.write(f"{detail}\n")
42  start_folder = 1
43  end_folder = 99
44  total_images = 0
45  processed_images = 0
46  for folder_number in range(start_folder, end_folder):
47      image_folder_path = f"{base_image_folder_path}{folder_number}/TIFF/"
48      output_folder_path = f"{base_output_folder_path}{folder_number}/
            OutputROIs/"
49      os.makedirs(output_folder_path, exist_ok=True)
50      image_files = glob.glob(image_folder_path + '*.tif')
51      total_images += len(image_files)
52      for img_file in image_files:
53          apply_template_matching(img_file, output_folder_path)
54          processed_images += 1
55          print(f"Verarbeitet: {processed_images}/{total_images} Bilder")
56  print("Fertig mit allen Bildern.")
```

Listing 1 OpenCV Template Matching Python Code (written with AI assistance)

First, the two input and output base folders are defined in the program. The folders for the previously defined template images must also be inserted. *ImageJ* was used to create the templates from cropped images. The first template was selected for the horizontal slanted edge chart (Fig. 40a) and the second template for the vertical version (Fig. 40b). The template creation process had to be repeated for each camera due to the different resolutions. Using the provided *cv.imread* function, the template images are imported (list. 1, lines 6-15).

(a) Horizontal (b) Vertical

Figure 40 OpenCV Slanted Edge Templates

The *apply_template_matching* function is defined to apply the template matching to the test images. As for the template images, the image with the test charts is imported as a grayscale image (list. 1, lines 17-22). Grayscale images are used because it is easier for the system to analyse and the used chart does not contain relevant colour information.
A loop runs through the two templates and the associated dimensions. Once the picture has been evaluated, the *cv.matchTemplate* method is used to obtain a result matrix, *res*.

The *cv.minMaxLoc* method searches for the position of the best match in the result matrix. Depending on the method, different positions are selected (list. 1, lines 27-32).
The coordinates of the top left corner, as well as the dimensions (height and width) of the template, are then saved in the *roi_ details* list. This is used to create a text file in the format that can be used by the iQ-Analyzer 6 in the resolution module (list. 1, lines 35-41). In addition to the upper left corner of the ROI and its dimensions it requires the name of the image, image dimensions and the number of ROIs detected (Fig. 41).

```
FileName:       \\ie-work\testlabor\Camtest\00_sonstige\#999999_IE_INTERN_ShakyCam_BAs\STEVE-Hauptordner\IMX623\1\TIFF\decomp_tif_1.tif
UserComment:    -
ImageHeight:    1552    ImageWidth:     1936
numOfROI:       2

1       980     722     101     42
2       939     867     44      83
```

Figure 41 Example Output ROI File / iQ-Analyzer 6 Input ROI File for Free Edge Tool

A low-contrast slanted edge chart or a high-contrast slanted edge chart was used for certain image series. Therefore, not all folders of a camera could be analysed with the same template. A start and end folder was defined for this purpose. Every folder containing images has the *apply_ template_ matching* function applied to it (list. 1, lines 42-56). The final output of the program is a folder with text files for every analysed folder, containing a ROI file for each image.

4.4.2 Analyzer Settings

The images had to be analysed for their effective resolution. As the custom slanted edge charts were not automatically detectable in the *iQ-Analyzer 6*, the supplied *TE000_ FreeEdgeChart* was selected in the *Resolution* module (Fig. 42).

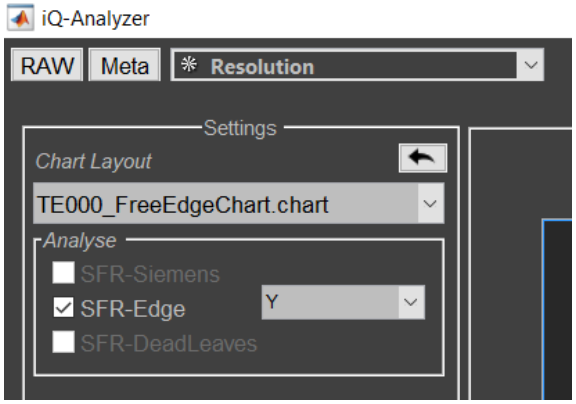


Figure 42 Chat Layout Selection

An image has to be selected as a *Bayer Image* upon import, as it is necessary for the software to be aware that the image has not undergone demosaicing. When selecting the manual detection, and starting the analysis, a window to add ROIs opens (Fig. 43).

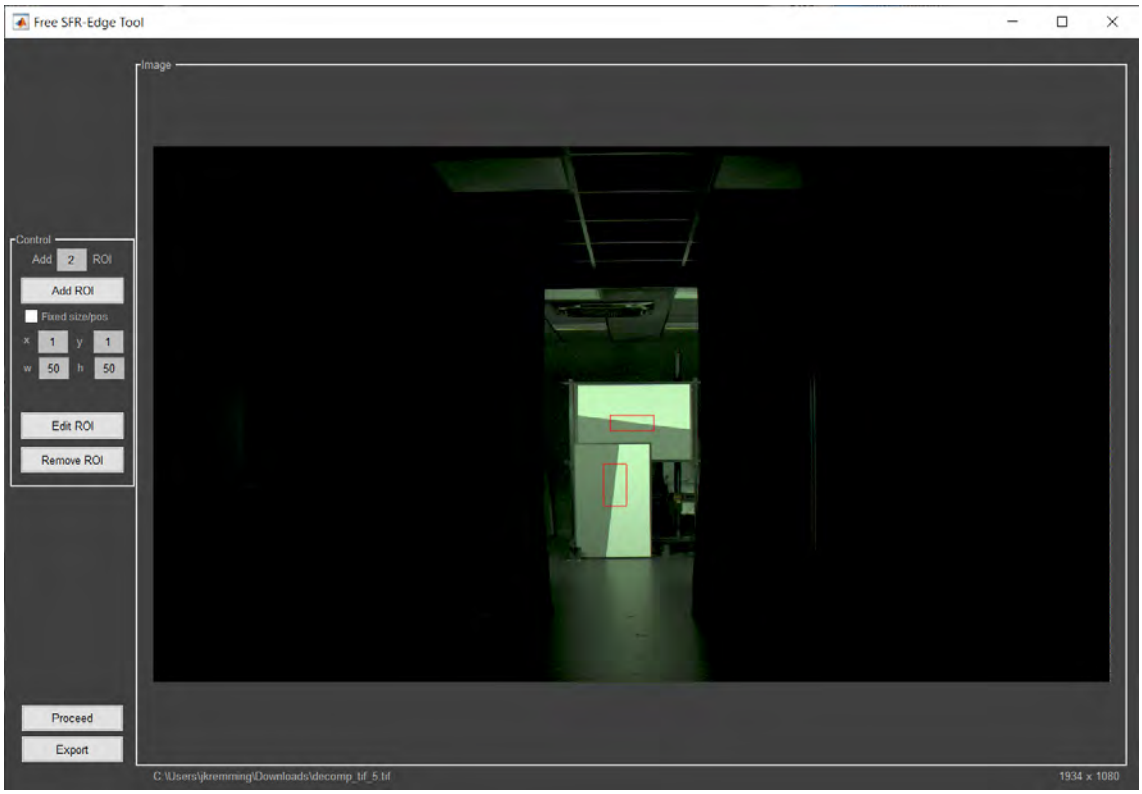


Figure 43 ROI Selection

The added ROIs can be saved in a text file and the same coordinates could be set for another analysis by selecting fixed ROIs instead of manual detection and importing. The software places an *XML* file in the directory of the image, containing the measurement values, as well as a check image, where the ROIs are marked (Fig. 39).

4.4.3 Analyzer CMD

Because the test procedures resulted in a large amount of data, it would have been it too time consuming to manually analyse all the images as described above. Fortunately the *iQ-Analyzer 6* has an *application programming interface* (API) that can be accessed via the command line interface. A batch script was based on this command to iterate through the images and analyse them.

The initial step was to set the path to the *iQ-Analyzer 6* installation, the image folder and the *myset-file* (list. 2, lines 3-5). The file contains basic settings for the respective software modules.

For the folder number the variable is used to iterate trough the paths for the images and ROI files. In this case, the iteration ranges from folder 93 to folder 99 (list. 2, lines 7-9).

As the *Analyzer* creates check images in the same folder of the images to be analysed, these images had to be ignored. This was implemented by searching for the string "check" in the filename and excluding them from the process. To analyse each image with the corresponding ROI file, the image number is extracted from the filename and used to set the filename for the ROI file (list. 2, lines 11-16).

At this point the *iQ-Analyzer 6* API is called (list. 2, line 17). First, the image to be analysed is specified, followed by the path for the settings. The resolution module is called

and the path for the ROI file inserted. As previously stated, the software requires the information regarding whether the image is demosaiced or not, and what the colour filter array pattern is, in this case RGGB. To activate the interpolation of the demosaiced image, this setting was set to *1*. Finally, the bit depth of the image was set.

```
@echo off
setlocal EnableDelayedExpansion
set IQAPath=C:\Program Files\Image Engineering\iQ-Analyzer V6.2.4.23
set BaseImageFolder=[...]\STEVE-Hauptordner\IMX728_b
set SettingsPath=[...]\IEA-settings\Setting\TE000_shakycam.myset
cd /d %IQAPath%
for /l %%H in (93,1,99) do (
    set "ImageFolderPictures=!BaseImageFolder!\%%H\TIFF"
        set "ImageFolderROI=!BaseImageFolder!\%%H\OutputROIs"
    echo Bearbeite Ordner: !ImageFolderPictures!
    for %%G in ("!ImageFolderPictures!\decomp_tif_*.tif") do (
        echo %%G | findstr /i "check" > nul
        if errorlevel 1 (
            set "filename=%%~nG"
            set "filenumber=!filename:~11!"
            set "ROIFilename=!ImageFolderROI!\ROI_decomp_tif_!filenumber!.
                txt"
            iqa6cmd.exe "%%G" "%SettingsPath%" resolution Resolution.Edge.
                ROIFile "!ROIFilename!" filetype "Bay" RAW.Pattern rggb RAW.
                Interpolate 1 RAW.BitDepth 16
            echo Bearbeite: %%G mit ROI-Datei: !ROIFilename!
        ) else (
            echo Ueberspringe: %%G
        )
    )
)
echo Fertig mit der Bearbeitung aller Dateien.
pause
```

Listing 2 Analyzer CMD script

5 Results

The results of the tests are listed and described in the following paragraph. Two types of charts were subjected to analysis within the test series. Both were slanted edge charts, differing only in their level of contrast: one with high contrast, while the other has low contrast (2.4). The results of the low contrast charts are not taken into account as they cannot be evaluated. The results base on the SFR10 value, which is a key measurement factor in this study. It is a measurement for the effective resolution of a camera. The value indicates the spatial frequency at which the camera retains 10 % of the original contrast. In addition, selected movement IDs that show a distinctive movement are used for the further analysis.

5.1 IMX728 - Examination of the Lens

After all images were analysed as explained in the previous chapter, the results are compared. First the differences between two lenses used with the IMX728 are shown. The SFR10 values for the standard lens proved to be particularly low in the preliminary test compared to the IMX490 and the IMX623, which is why it was necessary to examine the lens.

The graph presented (Fig. 44) is a violin plot showing the SFR10 values in line pairs per pixel (lp/px) for two cameras. The output of each analysed image consists of an XML file containing the SFR10 value of the horizontally placed and the vertically placed slanted edge chart, as well as the average value with a software specified weighting. The graphic is therefore divided into three categories: average, horizontal and vertical, with a comparison between the two cameras in each category. The comparison here consists twice of the same Sony IMX728 sensor, with the optics making the difference. The data set "IMX728_a" shows the analysis values of the camera with the 70 degrees HFOV lens supplied by the manufacturer, while "IMX728_b" represents the camera with a 65 degrees HFOV lens. The solution was to replace the standard lens with the 65 degrees HFOV lens of the IMX490 (called IMX728_b in the diagrams). This was possible because the bodies of the two cameras were identical and therefore an exchange was possible without complications and the horizontal field of view was very similar, with a difference of only five degrees. The movement and measurement parameters can be found in the header of the diagram. A movement ID is assigned to each camera data set. This can be viewed within the movement ID checklist (Appendix A.), whereby the deflection on the three axes is also visible in the graphic. In the graphic (Fig. 44), all axes are at zero, so the standstill is being compared. The high contrast (HC) slanted edge chart was used in this case. It can be seen that the median values of the sensor with the standard lens are significantly lower, which confirms the assumption of the previous test. In general, a performance loss of around 40 % to 50 % can be seen in all categories, which is why all future values for the IMX728 sensor will be taken from images recorded with the IMX490 lens.

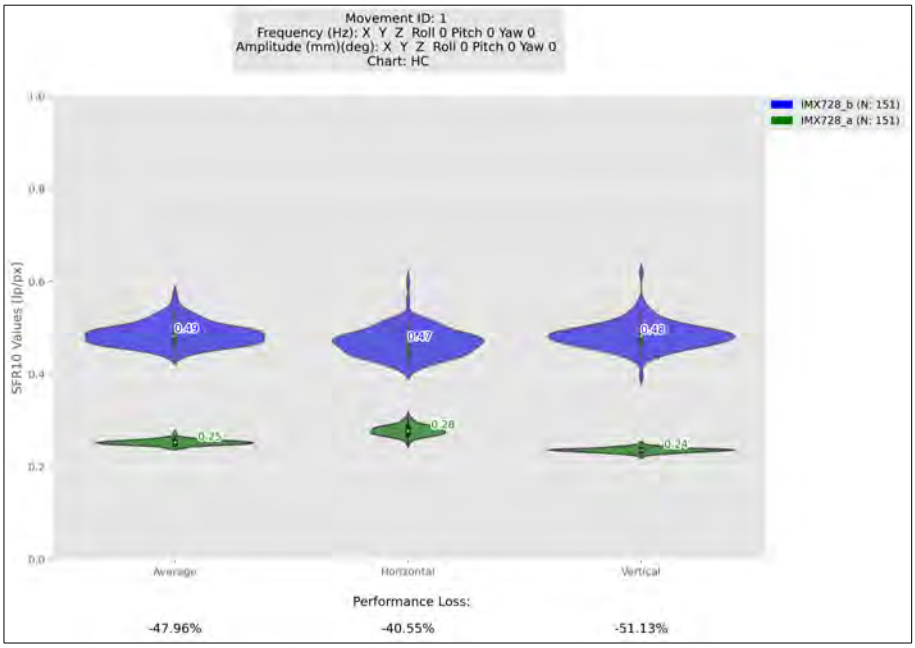


Figure 44 Comparison IMX728_a versus IMX728_b

5.2 SFR Analysis Comparison

5.2.1 Stillstand Comparison of all Cameras

To get an overview of the general performance of the different sensors and a frame of reference for the degradation, they were compared in a diagram at a standstill (Fig. 45). All cameras of the project are compared: IMX490, IMX623 and IMX728. The IMX623 sensor has the highest value with a median value of 0.64 lp/px, followed by the IMX490 with a value of 0.59 lp/px. The IMX728 has the lowest value within the average category, although it has the highest pixel count. A similar distribution can also be seen in the horizontal and vertical values. The graph shows that the IMX623 has the highest SFR10 value in all measured categories.

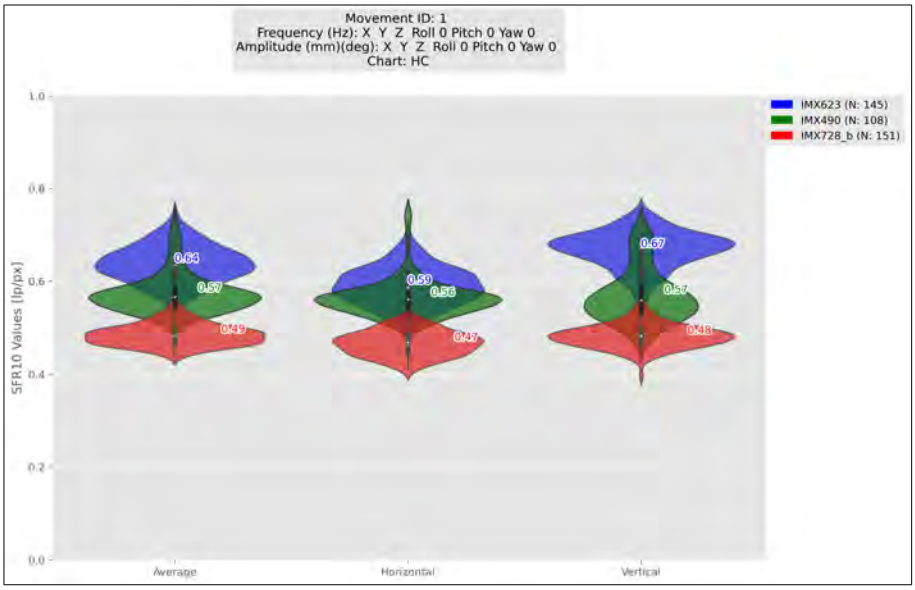


Figure 45 Comparison Stillstand

5.2.2 Comparison of Pitch Movement with all Cameras

In the following three violin plots, movement ID 1 (standstill) is compared with movement ID 17 (pitch: 5 Hz, amplitude: 1,5 mm) for all three cameras. Movement ID 17 was chosen, as is provides the a clear impact on the horizontal chart. The SFR values of the IMX728 (Fig. 46) reveal a clear decrease during a pitch movement. For the average part, the median value drops from 0.49 lp/px to 0.34 lp/px, which corresponds to a performance loss of 29.51 %. At 55.62 %, the loss is greatest in the horizontal direction, which is to be expected with a pitch movement. Due to the diagonal nature of the slanted edge chart, there is no optimal value of zero in the vertical category, but rather 20.30 %. The IMX623 sensor also shows a significant loss of performance in the horizontal SFR10 values (Fig. 47). The median value drops from 0.59 lp/px to 0.35 lp/px. This corresponds to a performance loss of 41.25 %. Vertical and average performance is similar to that of the IMX728. The IMX490 sensor also shows clear overall differences, although these are marginally below the values of the IMX623 in the horizontal and vertical graphs (Fig. 48). In order to get a better overview of the performance loss comparison, every value of the three SFR10 categories is set in relation to each other (Fig. 49). Here it can be seen that all sensors have a significant loss of performance, especially in the analysis values of the horizontal chart. However, it can be seen that the IMX623 and IMX490 are much more stable in all categories compared to the IMX728, indicated by a 15 % difference in the horizontal category.

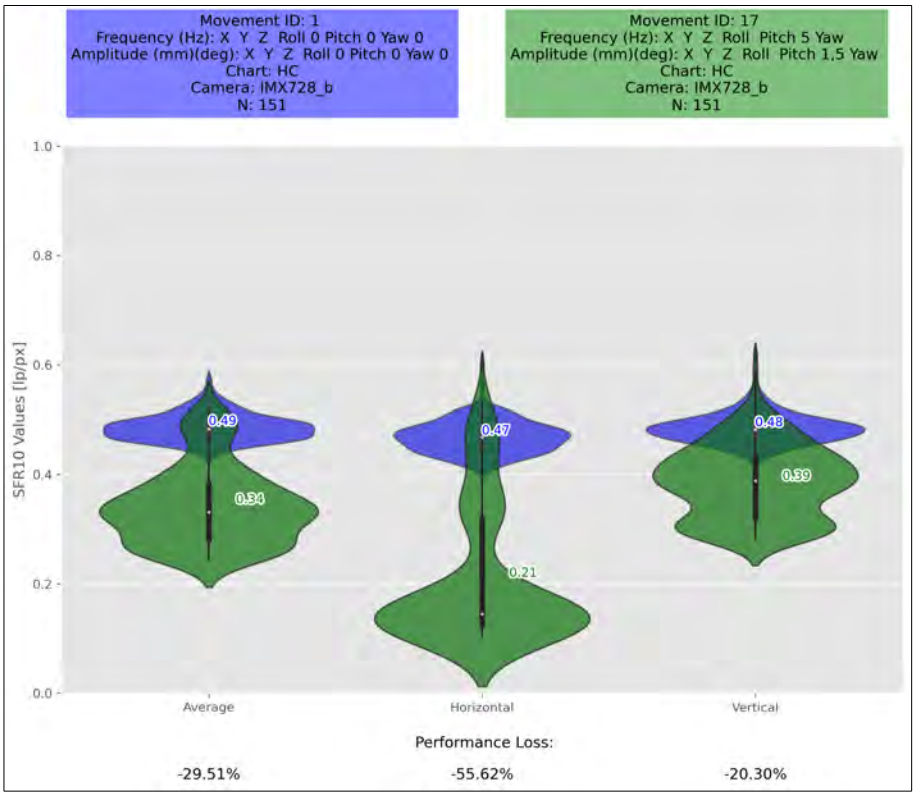


Figure 46 Comparison Stillstand versus 5 Hz Pitch - IMX728_b

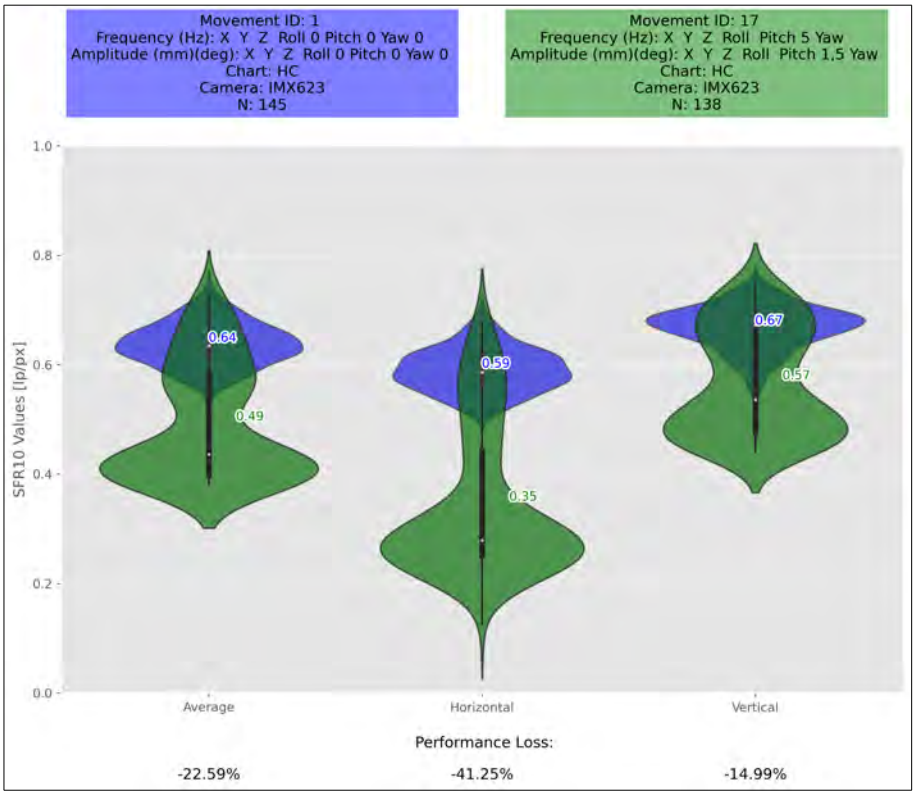


Figure 47 Comparison Stillstand versus 5 Hz Pitch - IMX623

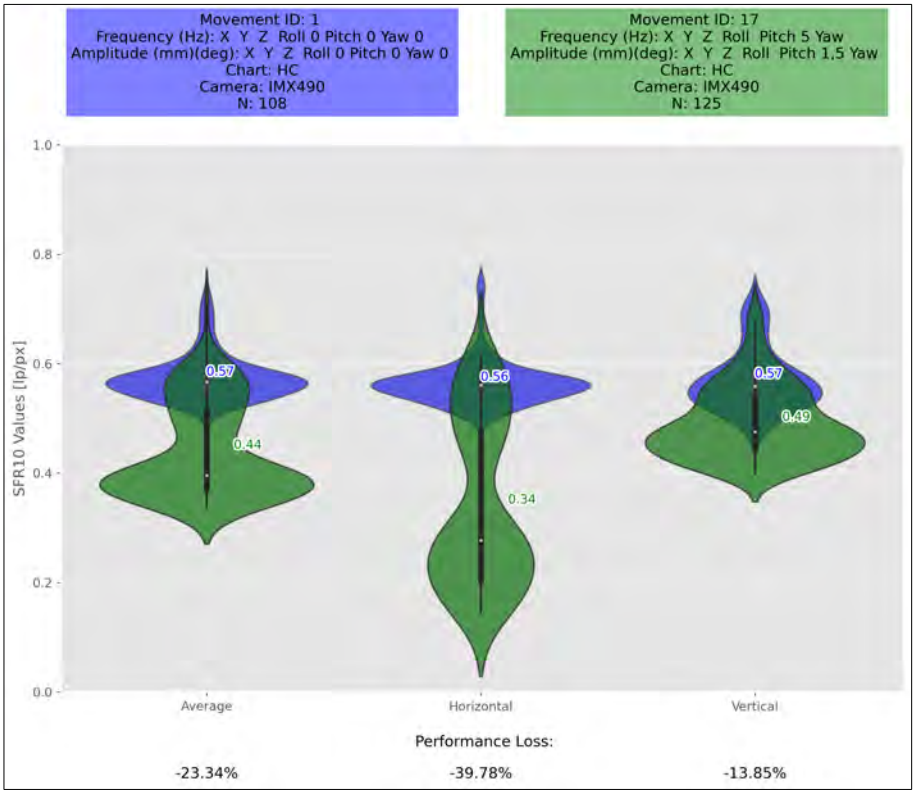


Figure 48 Comparison Stillstand versus 5 Hz Pitch - IMX490

Figure 49 Comparison Performance Loss - 5 Hz Pitch

5.2.3 Specification of the Worst Frames Based on SFR10 Values

To better understand the impact of the movements, all images were taken from one recording with 5 Hz and 1,5 Degrees pitch movement and sorted by their average SFR10 value. This results in getting frames with the lowest SFR10 value. Examining these images came to the conclusion, that the frames with the lowest SFR10 values are primarily derived from the ascending phase of the sine wave, just before the turning point. Some frames with similarly low SFR10 values are derived from the descending side (Fig. 50). The turning point of the sine function is the point with the steepest gradient and therefore the highest speed of the movement.

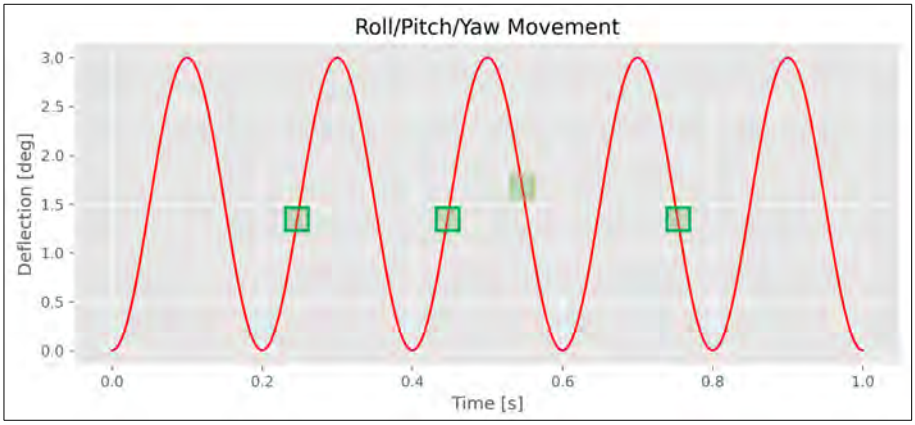


Figure 50 Sine 5 Hz 1,5 deg - Position of Worst Frames (Symbolic image)

5.2.4 Pitch Comparison - Zero Crossing Images (Sony IMX490)

Representative for the behaviour during movements, the Sony IMX490 serves as an example for the three camera systems. In the previous diagrams, all recorded frames of a five second recording were included in the analysis. In the following four graphs, only the five frames with the worst SFR10 values are included. As explained in the previous section, all of the worst images were taken during the zero crossing of the sine wave. This means that

from our sine curve, the point is taken at which the movement is the fastest and therefore the image most likely performs the worst. The five worst images were selected according to their horizontal SFR10 values, as a pitch movement occurred at the movement IDs at different frequencies. This means that only the horizontal value is relevant, as the vertical value would distort the results. In the case of movement ID 17 (5 Hz), the values were sorted according to the SFR10 average values after a subsequent check of the five worst images. It was subjectively determined that this was the only way to select the images of the zero crossing. Three different pitch movements (Movement ID 3, 10 and 17) are compared here with the SFR10 values of the standstill of the Sony IMX490. As with the previous graphs, a clear difference can be seen here in the horizontal category. The perfomance loss is at 40.91 % for movement ID 3 (Fig. 51) and then increases to 58.96 % (Fig. 52) when the frequency is increased from 1 Hz to 3 Hz. There is a difference of around 20 % here, although the impact is significantly lower when the frequency is increased from 3 Hz to 5 Hz. Here the difference is only around 5 % (Fig. 53). It can therefore be assumed that the higher the frequency, the smaller the differences in performance losses. This is visualised in the direct comparison of the performance loss values (Fig. 54). In the laboratory recordings, the variation of the frequency and amplitude also was also tested on the yaw axis in addition to the pitch axis. A similar performance loss and a similar distribution of values can be seen within the diagrams. In the case of a yaw movement, a degradation of the values cannot be seen in the horizontal segment, but in the vertical area (Fig. 55). Here, even with a frequency of 5 Hz and an amplitude of 1.5 degree, a performance loss of 66.75 % can be seen, which differs from the horizontal performance loss of pitch movements by around two percent.

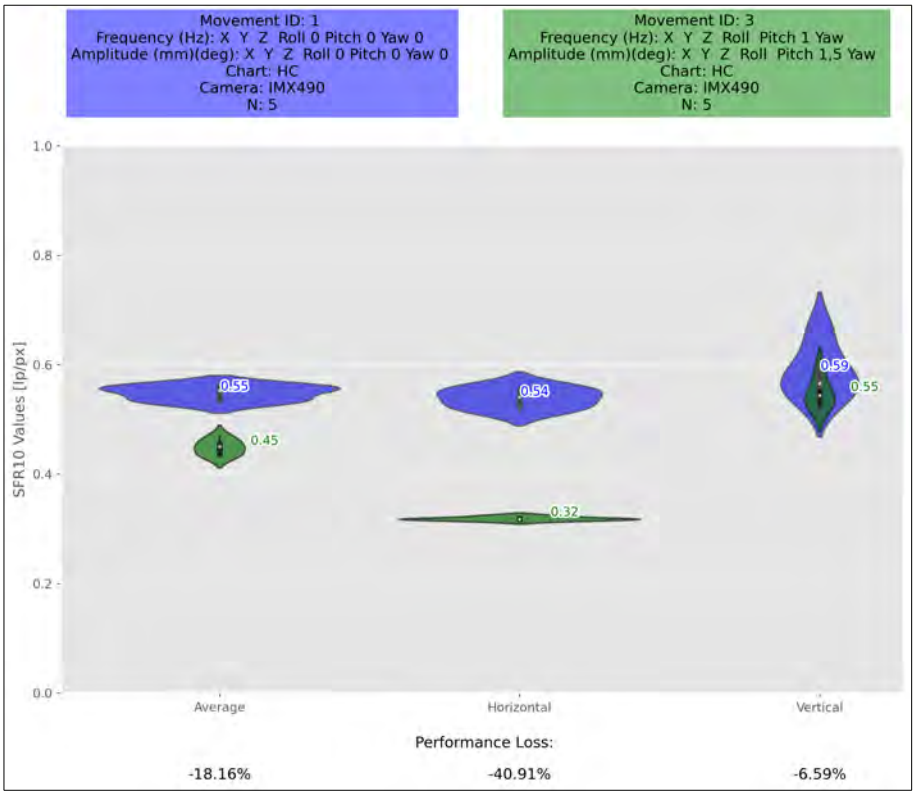


Figure 51 Comparison Stillstand versus 1 Hz Pitch - IMX728_b

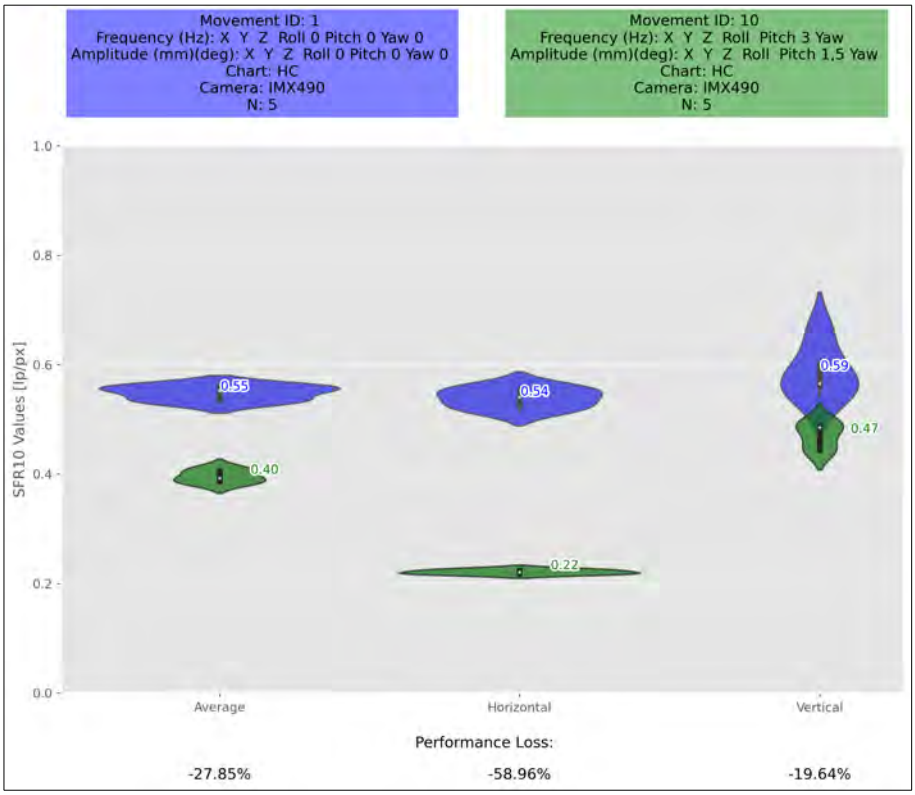


Figure 52 Comparison Stillstand versus 3 Hz Pitch - IMX490

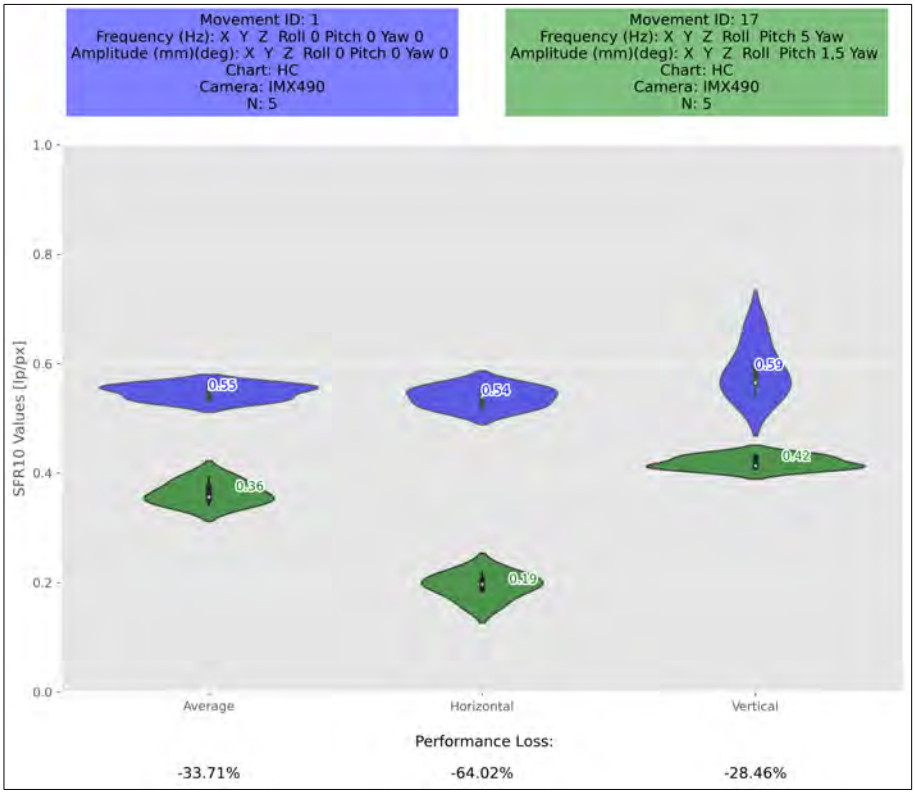


Figure 53 Comparison Stillstand versus 5 Hz Pitch - IMX490

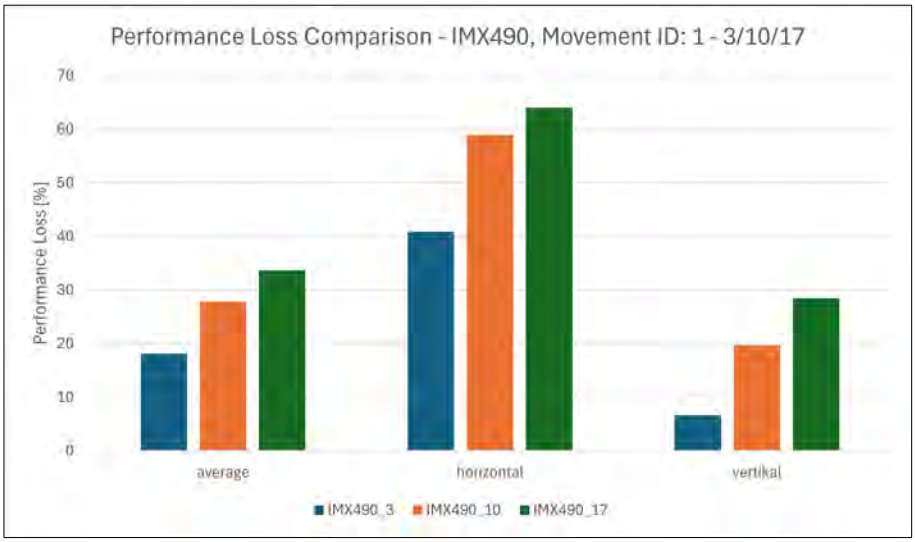


Figure 54 Comparison Performance Loss, Pitch - IMX490

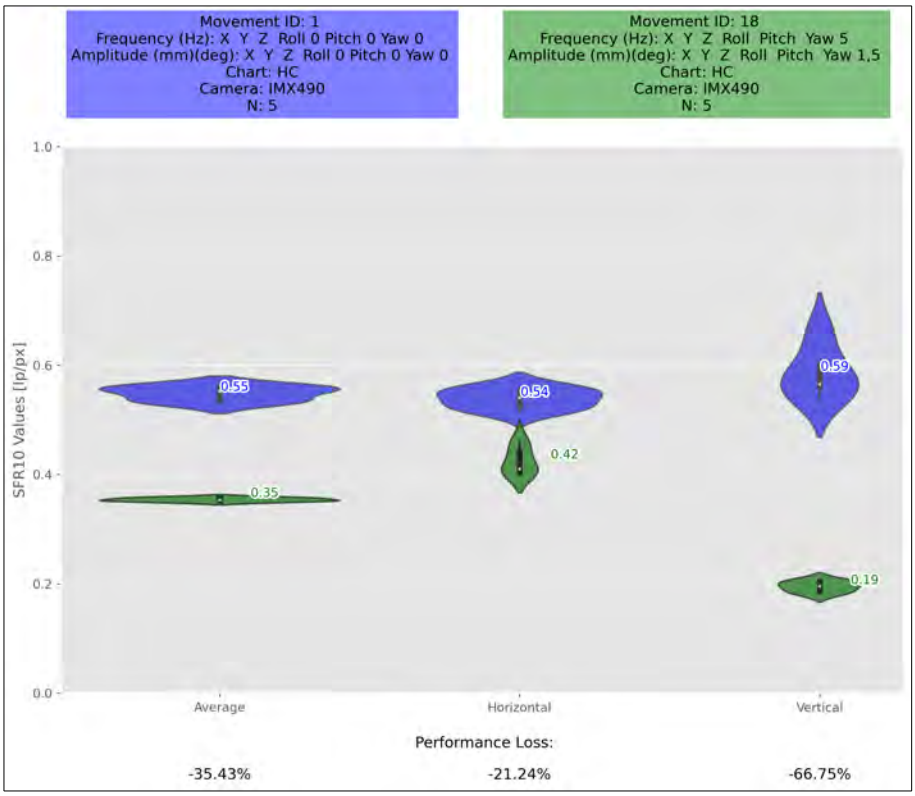


Figure 55 Comparison Stillstand versus 5 Hz Yaw - IMX490

5.3 SFR and Movement Correlation

To correlate real world movements to the SFR results a gyroscope recording was used, where the vehicle was travelling at 7 km/h on a cobbled surface before performing an emergency stop. The SFR10 measured on the horizontal chart (left y-axis) from all frames of the video were plotted and aligned with the z-axis (pitch) gyroscope data (Fig. 56). The gyroscope data was recorded in the vehicle and played back on the *STEVE-6D* hexapod (4.3). The absolute values of gyroscope data in degrees per second are indicated on the right y-axis. After about 15 seconds, the pitch axis reaches its first peak at 8,91 deg/s,

while the SFR10 drops to 0,15 lp/px, proving the correlation between motion speed and SFR degradation. When the car tilts forward during braking, the SFR10 values fall before briefly rising again, to the point when the car is standing. They then fall again sharply when the car raises back to the neutral position.

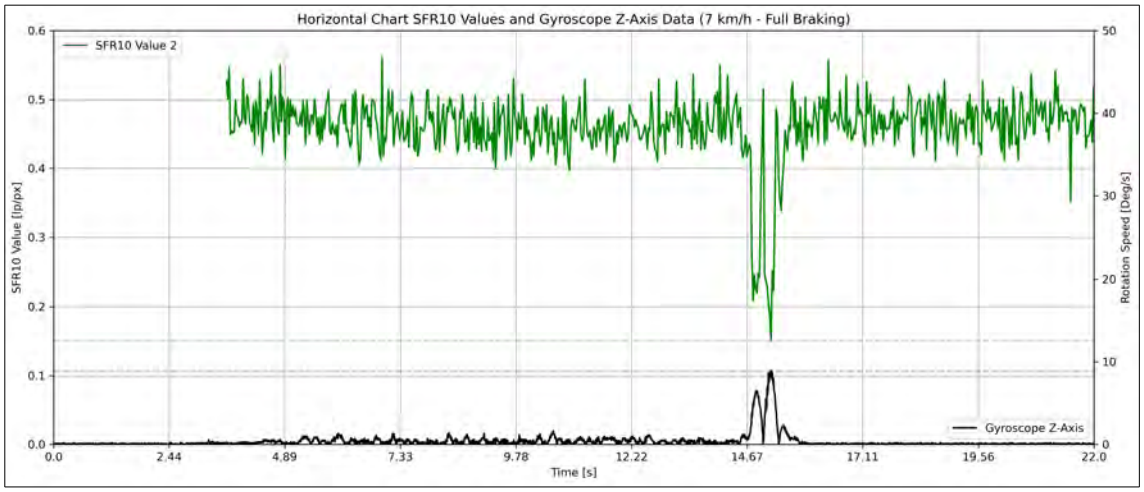


Figure 56 Correlation SFR10 and Gyroscope Data

5.3.1 Frequency Components of Movement Data

The recorded motion data with a sampling rate of approximately 108 Hz can show the frequency components up to a Nyquist frequency of 54 Hz. After a Fourier transformation, the frequency components along with their proportional amplitude can be plotted. For the movement ID 73 and the z-axis (pitch movement) (Fig. 57), the highest amplitudes are in the range of 1 Hz to 3 Hz, but also show an increase starting from just under 30 Hz. In movement ID 78 (Fig. 58), the distribution of lower frequencies and the 30 Hz region is similar, but this time with noticeable peak at around 17 Hz. Over all frequencies, the amplitude is higher. Regarding the x-axis (roll movement) (Fig. 59), there are increases in the lower frequencies, caused by the slalom movement, but no increase around 30 Hz. The data obtained from the stationary car (Fig. 60), with the engine off, indicates the presence of only a minimal amount of noise over all axes.

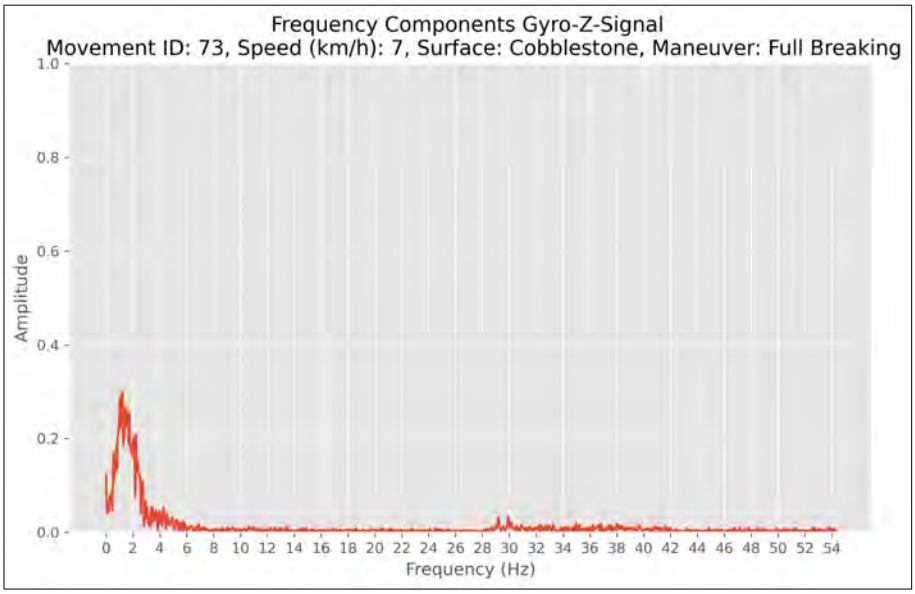


Figure 57 Frequency Components Movement ID 73

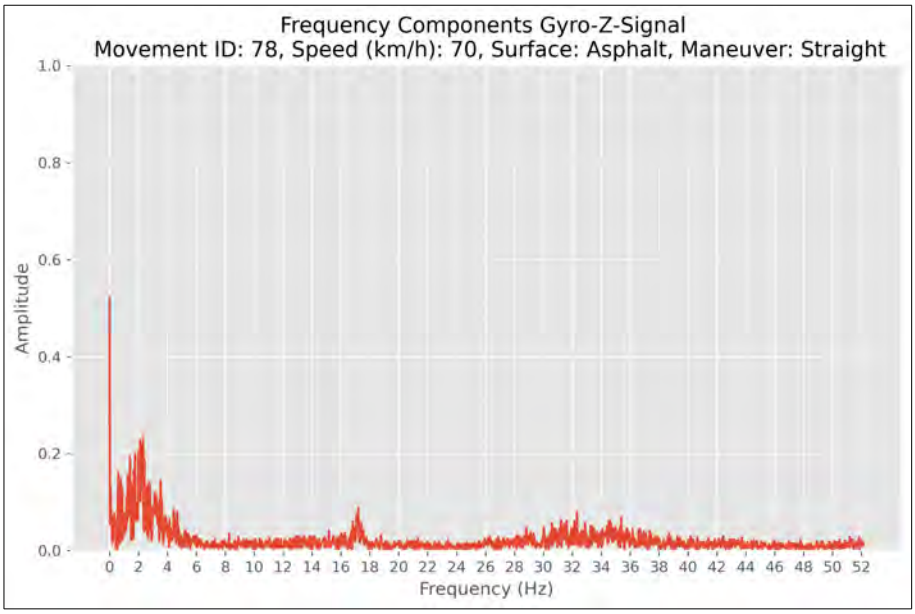


Figure 58 Frequency Components Movement ID 78

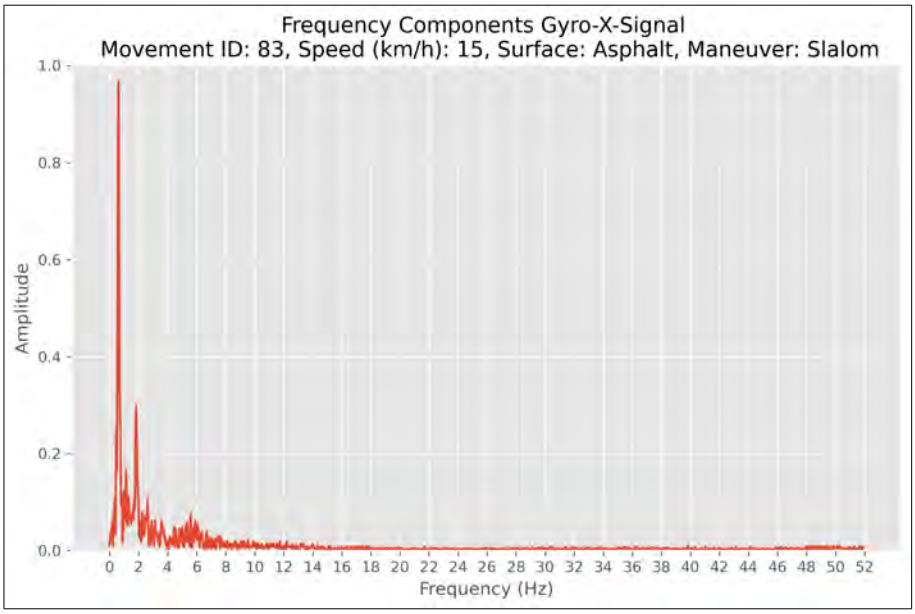


Figure 59 Frequency Components Movement ID 83

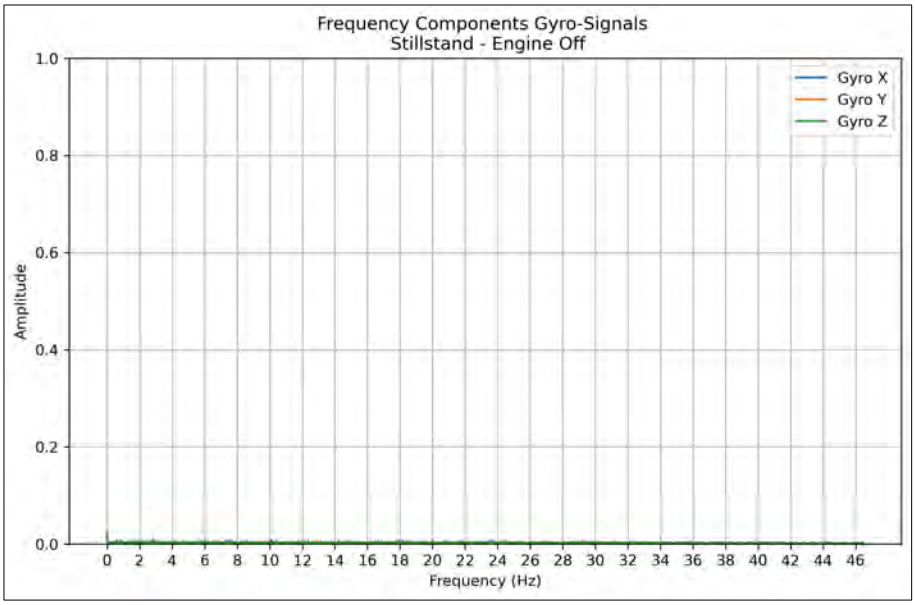


Figure 60 Frequency Components Movement Stillstand

5.4 Lost Cargo

A potential application for automotive cameras in ADAS is the detection of lost cargo. Luggage falling from the roof of a car or a truck losing some of its load can present a significant hazard to other road users. The camera systems of a following car must therefore be able to detect the object from a far distance in order to engage the brakes or to safely switch to a different lane. On roads with fast speeds such as highways or the Autobahn, a distance of 150 m is common to be a parameter for the lost cargo detection [29, page 55].

The SFR10 measurements from the laboratory tests were used to calculate the minimum size that an object must have to be just distinguishable from its surroundings (eq. 16). As the camera systems do not have completely identical viewing angles, the influence of the HFOV is calculated and visualised (Fig. 61). As the SFR10 performance increases, the difference in the minimum distinguishable object size decreases.

$$Min.Objectsize = Distance \times \tan\left(\frac{\frac{HFOV}{Image\ Width}}{SFR10}\right) \tag{16}$$

Min. Objectsize : Minimum Size of a Detectable Object (m)
Distance : Distance to Object (m)
HFOV : Horizontal Field of View of the Camerasystem (rad)
SFR10 : SFR10 value from Slanted Edge Chart (lp/px)
Image Width : Width of the Image (Pixel)

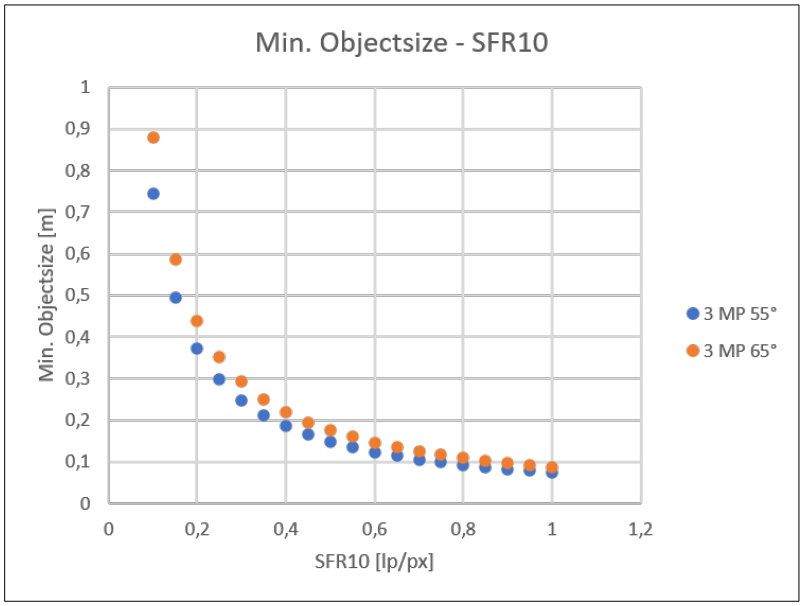


Figure 61 LostCargo SFR FOV

At first, the three camera systems are compared at their performance standing still. The Sony IMX728 is capable of detecting objects with a minimum size of 7.6 cm, while the

IMX623 requires an object with a minimum size of 12.7 cm. The IMX490 falls somewhere in between with a minimum size of 10.3 cm (Fig. 62). In order to compare these results to those from SFR10 measurements of the cameras moving on the hexapod, movement ID 73 was chosen and the worst 15 frames in terms of lowest horizontal SFR10 measurements selected (Fig. 63). This movement ID was selected as it shows a distinctive correlation between movement and decline in SFR10 values (Fig. 56). The results obtained for this movement ranged from 14.9 cm on the IMX728 to 17.6 cm on the IMX623.

The relative differences between the systems in question become less pronounced when comparing the stillstand measurements to those taken during the simulated movement (Fig. 64). The IMX728 as the sensor with the smallest minimal objectsize is set as the reference with 100 %. While the IMX623 sensor required a subject that was 66 % larger than that of the IMX728 during stillstand, the object size during a moving period (ID 73) only needed to be 18 % larger. The increase in objectsize compared to the IMX728 for the IMX490 went down from 35 % in standstill to 12 % in movement. The results indicate that the benefit derived from the enhanced pixel count of the IMX728 diminishes when transitioning from a stationary to a moving scenario.

Sensor	**Image Width (Pixel)**	**HFOV (°)**	**Distance (m)**	**SFR10 (lp/px)**	**Min. Object Size (m)**
IMX623	1936	60	150	0.64	0.127
IMX728	3857	55	150	0.49	0.076
IMX490	2897	65	150	0.57	0.103

Table 5 Lost Cargo Calculation (ID 1, Stillstand)

Sensor	**Image Width (Pixel)**	**HFOV (°)**	**Distance (m)**	**SFR10 (lp/px)**	**Min. Object Size (m)**
IMX623	1936	60	150	0.46	0.176
IMX728	3857	55	150	0.25	0.149
IMX490	2897	65	150	0.35	0.168

Table 6 Lost Cargo Calculation (ID 73, Full Braking from 7km/h)

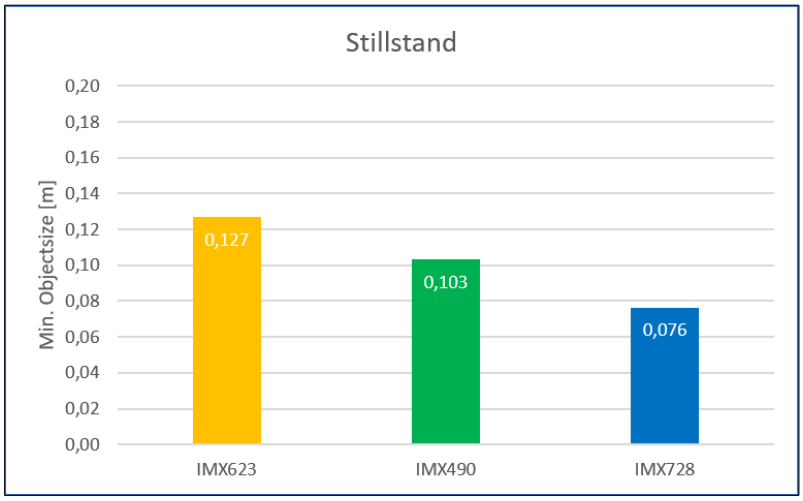


Figure 62 LostCargo Stillstand

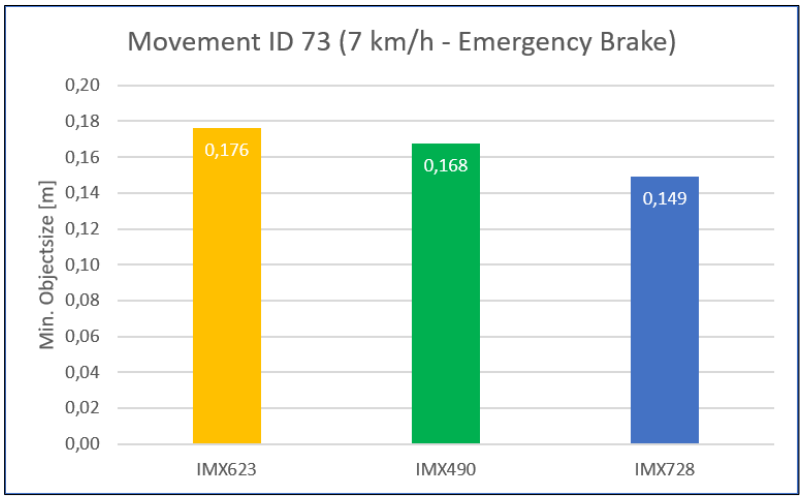


Figure 63 LostCargo 73

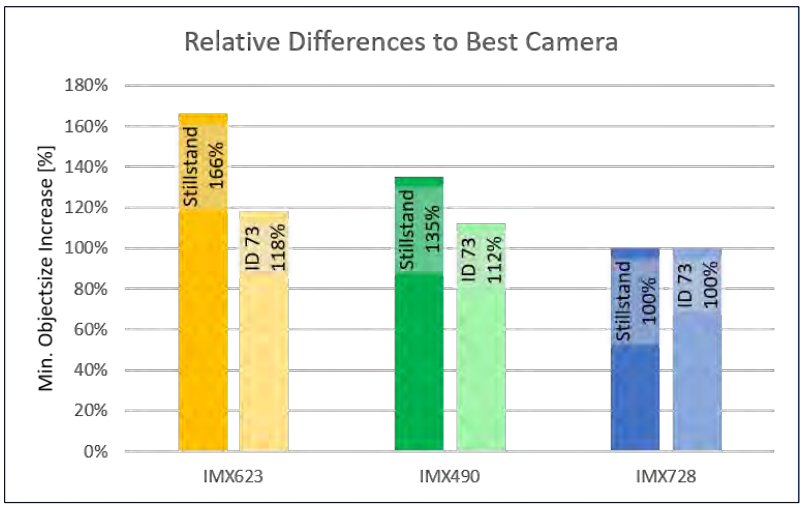


Figure 64 LostCargo Stillstand vs 73 percent

6 Conclusion

The following section discusses and interprets the results explained above. It also provides an outlook for future research steps based on the experiences in the project.

6.1 Discussion

The investigations of the SFR10 values in combination with the recorded movement data show that the cameras exhibit significant drops in the analysis values in the targeted extreme movement simulations of a real test drive (emergency braking, slalom, fast acceleration). A representative example of this is the emergency braking process at a speed of 7 km/h on a cobblestone surface. The movement peaks can be assigned to the lowest SFR10 values in the data sets. This suggests that motion artefacts such as blurring during an intensive motion process strongly influence the quality of the image. Therefore, the motion data is well suited to perform simulations with cameras that can be tested in a laboratory environment. One aspect to keep in mind however, is the limitation of the hexapod, which cannot simulate a real car movement on all axes. The IMU output frequency can also be classified in this context. At around 106 Hz, this was sufficient for the project.

A comparison of the individual cameras shows that the higher the movement frequency is set, the lower the differences of performance loss are. However, it must be mentioned here that due to the limitations of the hexapod, the amplitude also had to be set lower at increasing frequencies, when combining the movements on multiple axes.

The performance loss of the SFR10 values is clearly visible in the case of separate horizontal and vertical movements. The decrease of the effective resolution always takes place within the associated movement range. As the charts are slanted edges, they contain a diagonal edge. If, for example, a pitch movement is carried out, a significant loss of values can be seen within the horizontal range, but also a slight loss of performance in the vertical range. When comparing all cameras with vertical movements of 5 Hz with an amplitude of 1.5 mm, it can be seen that the IMX728 performs significantly worse than the other two cameras.

In contrast to the observation of the SFR10 values alone, it can be seen that the IMX728 performs best in the case of a lost cargo. The results show that the IMX728 can detect objects with a minimum size of 7.6 cm in a stationary state. It was to be expected that the camera with the highest resolution would generally perform best. However, the industry is currently moving in the direction that a high resolution contributes to a strong improvement in image quality. If you look at the difference between the minimum object sizes at standstill compared to motion, it becomes clear that the percentage advantage of a high-resolution camera decreases significantly. The IMX623 requires a 66% larger object at standstill compared to the IMX728, whereas in motion it only requires an 18% larger object for minimum detection. With the IMX490, the difference is reduced from 35% to 12%. There is a significant advantage in the stationary state, but this is partially relativised in the case of strong movements, which occur more frequently in hazardous situations in road traffic.

The examination of the standard lens of the Sony IMX728 camera initially shows that

a faulty lens from the manufacturer was part of the camera system supplied. This may be an isolated case, which was compensated by using the optics of the IMX490. By changing the optics, a performance increase of 40 % to 50 % was achieved. However, it cannot be assumed that the new lens is able to show the full potential of the IMX728 sensor. There is no additional comparison for this, as ordering another lens for the IMX728 sensor was not possible for this study.

This fact must be taken into account when comparing the SFR10 performance at standstill of the three cameras systems. The IMX623 performs best here, while the IMX728 performs worst, contrary to previously defined expectations. As the SFR10 value is a measurement factor for just noticeable differences, the camera with the highest resolution should deliver the best value at standstill. The IMX728 offers around 8.4 MP, followed by the IMX490 with 5.4 MP and the IMX623 with 3 MP. The order of the graphs (Fig. 45) is therefore the opposite of what is initially expected. At 2.1 micrometers, the IMX728 has the smallest pixel size of all three sensors. However, this should not make a significant difference, as larger pixels of the others sensors can capture more light in low light conditions, but should not provide a significant advantage in a stable and well-illuminated (300 lx in this study) laboratory environment.

To summarise the experiment, it can be said that a simulation in a laboratory is possible by combining the movement data of an IMU and a hexapod. However, this is limited in terms of hardware and not all axes can be fully taken into account. There is also a visible correlation between the extreme driving manoeuvres and the corresponding low points of the SFR10 values. In this context, it is possible to define and analyse characteristic value ranges and images for the respective manoeuvres. Looking at the more practical case of a lost cargo, it can be stated that the movement has a significant influence on the image quality. Although the highest-resolution sensor performs best here, the percentage advantage in image quality decreases as soon as movement is involved.

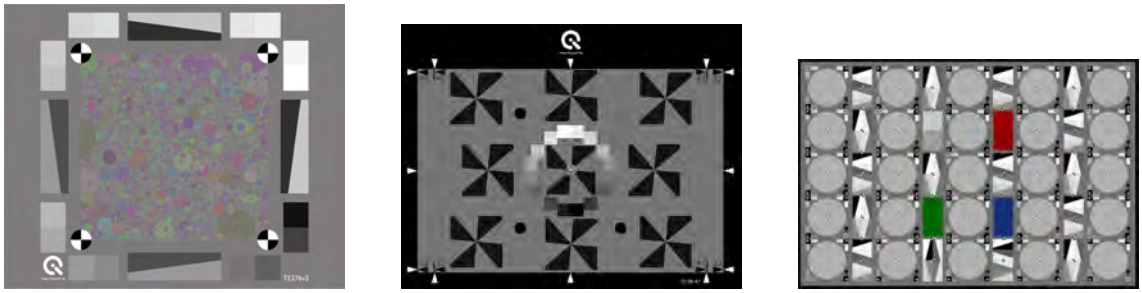
6.2 Suggestions for Future Research

The extent to which the influence of a car's movements on the camera systems can be analysed is very large. However, it was only possible to scratch the surface of this topic in this thesis. In future studies, other charts could be used to assess the effective resolution of the cameras. On the one hand, dead-leaves charts (Fig. 65a) would be suited to analyse the impact on the reproduction of fine details and texture, but other charts like the TE296 (Fig. 65b) with more slanted edges in different orientations are also conceivable in order to be able to better assess the local influence of the roll movement on effective resolution. Furthermore, the influence of different exposure times can also be analysed. In this thesis, only the exposure time of 15 ms was analysed. Longer exposure times, and therefore longer integration times, may be perceived as a low-pass filter, whereby movements occurring during this period are less likely cause blurred images, depending on the type of movement. A short and quick movement during exposure may not be recognisable on the images with longer exposure times.

The influence of the installed optics could also be investigated. For this task, a different chart layout could be employed to analyse the sharpness distribution over a larger area

of the image. The TE268 (Fig. 65c) chart would be a conceivable option for measuring the resolution with 25 distributed Siemens stars. This could be achieved by utilising concentric circles around the optical centre, thus enabling the investigation of the influence of movement (Fig. 65b) or the lens (Fig. 65c).

Parts of the concept of motion detection via an IMU could also be revised. A central point for improvement would be the synchronisation of the recording of the motion with the recording of the image data. With more precise IMUs, the frequencies of the movements could be captured more finely and also reliably recorded up to a higher level of frequencies. This would also require a different way of reproducing the movements. A larger motion playback device would be suited to also simulate the acceleration on the linear axes as well as potentially play back higher frequencies and amplitudes. The choice of vehicle could also have an influence on the motion and image data depending on the type of suspension used, which correlates with the class and price of the vehicle. All in all, a lot of different factors are involved in this topic and are waiting for their impact to be explored.



(a) TE276 [30] (b) TE296 [31] (c) TE268 [32]

Figure 65 Potential Charts for Future Research

References

[1] Sony Semiconductor Solutions Corporation, "Image Sensor for Automotive Use [Product]". [Online]. Available: https://www.sony-semicon.com/en/products/is/automotive/automotive.html (visited on 05/15/2024).

[2] J. Nakamura, 1st ed. Taylor & Francis, 2006, "Image sensors and signal processing for digital still cameras", ISBN: 0849335450; 9780849335457.

[3] Leopard Imaging Incorporated, "IMX490 Datasheet". [Online]. Available: https://leopardimaging.com/wp-content/uploads/2024/01/LI-IMX490-GMSL2-xxxH_Datasheet.pdf (visited on 06/27/2024).

[4] Leopard Imaging Incorporated, "IMX623 Datasheet". [Online]. Available: https://leopardimaging.com/wp-content/uploads/2024/05/LI-VENUS-IMX623-GMSL2-xxxH_Datasheet.pdf (visited on 06/27/2024).

[5] Leopard Imaging Incorporated, "IMX728 Datasheet". [Online]. Available: https://leopardimaging.com/wp-content/uploads/2023/12/LI-IMX728-9295-xxxH-_Datasheet.pdf (visited on 06/27/2024).

[6] Advanced Micro Devices, Incorporated, "HDR Decompanding". [Online]. Available: https://docs.amd.com/r/2023.2-English/Vitis_Libraries/vision/api-reference.html_2_46 (visited on 06/27/2024).

[7] B. E. Bayer, "Color imaging array", 1976, US3971065A.

[8] B. M. Deegan, 2014, "The effect of split pixel HDR image sensor technology on MTF measurements", 90230Z. DOI: 10.1117/12.2039327. [Online]. Available: https://doi.org/10.1117/12.2039327.

[9] Lucid Vision Labs GmbH, "Sony IMX490 CMOS Sensor: On-Sensor HDR for 24-bit Imaging". [Online]. Available: https://thinklucid.com/tech-briefs/sony-imx490-hdr-sensor-and-flicker-mitigation/ (visited on 05/18/2024).

[10] V. C. Venezia, A. C.-W. Hsiung, K. Ai, *et al.*, "1.5μm Dual Conversion Gain, Backside Illuminated Image Sensor Using Stacked Pixel Level Connections with 13ke-Full-Well Capacitance and 0.8e-Noise", 2018. DOI: 10.1109/IEDM.2018.8614484.

[11] NXP B.V, "UM10204", 2021. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf (visited on 05/18/2024).

[12] DFRobot Corporation. [Online]. Available: https://www.dfrobot.com/product-1716.html (visited on 05/18/2024).

[13] Arduino, "FastIMU". [Online]. Available: https://reference.arduino.cc/reference/en/libraries/fastimu/ (visited on 05/18/2024).

[14] DFRobot Corporation. [Online]. Available: https://www.dfrobot.com/product-838.html (visited on 05/18/2024).

[15] AZ-Delivery Vertriebs GmbH. [Online]. Available: https://www.az-delivery.de/products/datenlogger-modul (visited on 05/18/2024).

[16] Solectrix GmbH, "SXIVE-Bundles für den schnellen Einstieg". [Online]. Available: https://solectrix.de/sxive/ (visited on 05/15/2024).

[17] K. Masaoka, T. Yamashita, Y. Nishida, and M. Sugawara, *Opt. Express*, vol. 22, no. 5, pp. 6040–6046, Mar. 2014, "Modified slanted-edge method and multidirectional modulation transfer function estimation". DOI: 10.1364/OE.22.006040. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-22-5-6040.

[18] Arne Valberg. Wiley, 2007, "Light Vision Color", ISBN: 9780470012123.

[19] Image Engineering GmbH & Co. KG, "Chart Sizes". [Online]. Available: https://www.image-engineering.de/products/charts/all/1139-text-custom-chart#chart-sizes (visited on 05/15/2024).

[20] Image Engineering GmbH & Co. KG, "Resolution". [Online]. Available: https://www.image-engineering.de/library/image-quality/factors/1055-resolution (visited on 05/18/2024).

[21] "Photography — Electronic still picture imaging — Resolution and spatial frequency responses," International Organization for Standardization, Standard, 2017.

[22] Image Engineering GmbH & Co. KG, "STEVE-6D". [Online]. Available: https://www.image-engineering.de/products/equipment/measurement-devices/825-steve-6d (visited on 06/27/2024).

[23] Image Engineering GmbH & Co. KG, "STEVE-6DL datasheet". [Online]. Available: https://www.image-engineering.de/content/products/equipment/measurement_devices/steve-6d/downloads/STEVE-6DL_data_sheet.pdf (visited on 06/27/2024).

[24] Sergey Velichko, "Resolutions", 2024. [Online]. Available: https://www.onsemi.com/company/news-media/blog/automotive/en-us/a-journey-through-advancements-in-automotive-image-sensors (visited on 05/15/2024).

[25] Sony Semiconductor Solutions Corporation, "News Releases", 2023. [Online]. Available: https://www.sony-semicon.com/en/news/2023/2023091201.html (visited on 05/15/2024).

[26] OmniVision Technologies Incorporated, "OMNIVISION Announces Automotive Image Sensor with TheiaCel™ Technology Now Compatible with NVIDIA Omniverse for Autonomous Driving Development", 2024. [Online]. Available: https://www.ovt.com/press-releases/omnivision-announces-automotive-image-sensor-with-theiacel-technology-now-compatible-with-nvidia-omniverse-for-autonomous-driving-development/ (visited on 05/15/2024).

[27] DFRobot Corporation. [Online]. Available: https://wiki.dfrobot.com/Gravity__BMI160_6-Axis_Inertial_Motion_Sensor_SKU__SEN0250 (visited on 05/15/2024).

[28] Image Engineering GmbH & Co. KG, "STEVE-6D Manual", 2021. [Online]. Available: https://www.image-engineering.de/content/products/equipment/measurement_devices/steve-6d/downloads/STEVE-6D_manual.pdf.

[29] VDI Wissensforum GmbH, Ed., *"Fahrerassistenzsysteme und automatisiertes Fahren 2018"* (VDI-Berichte), 1st ed. Düsseldorf: VDI Verlag, 2018, vol. 2335.

[30] Image Engineering GmbH & Co. KG, “TE276”. [Online]. Available: https://image-engineering.de/products/charts/all/401-te276 (visited on 06/27/2024).

[31] Image Engineering GmbH & Co. KG, “TE296”. [Online]. Available: https://image-engineering.de/products/charts/all/1252-te296 (visited on 06/27/2024).

[32] Image Engineering GmbH & Co. KG, “TE268”. [Online]. Available: https://image-engineering.de/products/charts/all/584-te268x (visited on 06/27/2024).

Appendix

Equipment Overview

Manufacturer	Model	Type
Aladdin Lights	AMS-200BTD	light source
AZ-Delivery	Datalogger Module	datalogger
DFRobot	BMI160 SEN0250	sensor
DFRobot	DFRduino UNO R3 DFR0216	microcontroller board
Image Engineering	iQ-Chartmount-V	mounting equipment
Image Engineering	LED-Panel	measurement device
Image Engineering	Slanted Edge Chart High Contrast	chart
Image Engineering	Slanted Edge Chart Low Contrast	chart
Image Engineering	STEVE-6D	hexapod
Leopard Imaging	LI-IMX728-9295-070H	camera system
Leopard Imaging	LI-IMX490-GMSL2-065H	camera system
Leopard Imaging	LI-IMX623-GMSL2-060H	camera system
Solectrix	SXIVE-Bundle	development board

Table 7 List of Equipment

Appended Files

The included CD contains source code of the programs written for this thesis, as well as a ID checklist for the movements on the hexapod. Additionally, a digital version of this thesis is included as well. The code was written with AI assistance.

A. MovementID.xlsx

B. BMI_160_ACC_GYRO_SD

C. Bag2Raw_p728.sh

D. CS2_TemplateFinder.py

E. csv2steveFormatted.py

F. IQA6_run_IMX728.bat

G. TIFFConversion.ijm

Declaration of Authorship

I, Julian Kremming, hereby declare that the declared parts in this bachelor thesis were independently composed and written by myself.

I, Mike Kupezki, hereby declare that the declared parts in this bachelor thesis were independently composed and written by myself.

All content and ideas drawn directly or indirectly from external sources are indicated as such.

All sources and materials that have been used are referred to in this thesis.

The thesis has not been submitted to any other examining body and has not been published.

Place, Date	Signed: Name
Place, Date	Signed: Name