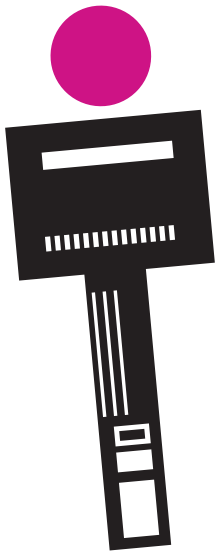


RAM ROBOTICS AND MECHATRONICS

# THE DESIGN OF A PROTOTYPE HANDHELD 3D SURFACE RECONSTRUCTION SETUP FOR MONITORING SKIN DISEASES

M. (Mike) Evers

MSC ASSIGNMENT

**Committee:**
dr. ir. F. van der Heijden
dr. ir. J.F. Broenink
ir. E. Molenkamp

July, 2021

037RaM2021
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

*This page is intentionally left blank.*

**Abstract**

This master thesis describes the design and implementation of a prototype handheld 3D surface reconstruction system designed to monitor skin diseases. The goal of the prototype system is to function as a test bench that can be used for testing new methods, better optimization, and other improvements. The project focuses on data acquisition and constructing a working prototype system pipeline that extends from the data acquisition to the SLAM server itself.

The prototype system consists of two main components; a handheld scanner and the main processing unit. The handheld scanner is equipped with a stereo camera setup for the acquisition of depth and color information, an inertial sensor for rotational data of the scanner itself, an illumination unit that can illuminate the surface and can project images/patterns onto it, and a single-board computer with a dedicated GPU for synchronization of the acquired data, (optional) undistortion of the images, and for the GPU optimized landmark finding algorithm. The embedded controller sends the resulting landmarks and the rotational data from the inertial sensor wireless to the external main processing unit. The external main processing unit processes the incoming data through a GPU optimized extended Kalman filter simultaneous localization and mapping (EKF-SLAM) algorithm. This SLAM algorithm can create a point cloud of gathered measurements, and so a 3D reconstruction of a surface can be constructed.

The results of the run-time evaluations show that the system is, at its maximum, capable of handling input image sizes of $980\,\text{px} \times 620\,\text{px}$. The minimal throughput time for this resolution is around 1 s, this includes; loading the data, finding the landmarks, and sending the landmarks to the external SLAM server. Experiments with the prototype system show that a surface can be reconstructed, but results show that the system is sensitive for errors in the landmark acquisition outside of a test environment.

# Contents

# 1 Introduction

This document contains the *thesis* for the master thesis project *"The design of a Prototype handheld 3D Surface Reconstruction setup for Monitoring Skin Diseases"* of the master Embedded Systems student *Mike Evers*. The following sections give an introduction to the project by explaining the project context, problem statement, project goals, and shortly discussing the prior work done by the research group regarding this project.

## 1.1 Project Context

Skin diseases are more common than we think, according to a study of Tizek et al. (2019). In the study, 2701 randomly selected individuals were tested for skin diseases. The results show that 1662 participants (64.5%) had at least one skin abnormality. Basra and Shahrukh (2009) shows that the burden on people with skin diseases is immense. For example, the study indicates that 47% of patients with atopic eczema, which affects approximately 10-20% of children in western communities, are frustrated with their disease. The same research also shows that approximately 10% of psoriatic patients suffer from suicidal ideation. Because of the significant adverse psychological and physical effects, patients diagnosed with severe skin disease need to undergo treatment to combat the effects.

## 1.2 Problem Statement

Treatment of skin diseases is, most of the time, combined with long-term monitoring of the progression of the disease. A majority of patients diagnosed with moderate to severe forms of psoriasis undergo systematic therapy. This type of therapy makes use of toxic medication (Jyothi et al., 2021). Therefore, patients undergoing this kind of treatment need to be checked up frequently by a medical professional to monitor the condition's growth/intensity. Based on the progression, the type and amount of medication are adjusted accordingly to keep the medication's adverse side effects to a minimum.

One common skin disease is psoriasis. The progression of a patient's psoriasis symptoms is monitored by evaluating the so-called Psoriasis Area Severity Index score (PASI score). The PASI score assessment evaluates the skin's redness, thickness, and scaling on different anatomic parts of the body (Okun, 2008). Currently, a medical practitioner calculates/evaluates the PASI score by hand. This method of assessing the severity works, but it is an imperfect subjective method, prone to human error. Another disadvantage of the current practice is that the severity assessment has to be done by a medical professional. It can become time-consuming for patients who need to be checked frequently and expensive when the patient has no or little medical insurance.

## 1.3 Project Goal

At the Robotics And Mechatronics research group (RAM) at the University of Twente, multiple students and professors work together to research and develop a handheld 3D surface reconstruction camera system for monitoring skin diseases. This novel monitor system aims to remove the human interpretation of the severity assessment and replaces it with a quantitative method, making the results more reliable and less subjective to human error. This master project contributes to this research project.

The goal of this master project is to develop a prototype handheld scanner that can measure depths, log inertial data, and can be used to reconstruct the surfaces of objects. Some component requirements are linked to the monitorization of skin diseases, but within this project, the focus is on developing a prototype system that can measure arbitrarily test objects. This means that there will not be experimented with actual body parts or patients with skin diseases. The purpose of the to-be-designed prototype is to function as a test bench for the research group. This test bench can then be used to test new methods, experiment with optimizations, or even eventually function as proof of concept when the prototype is further developed. So to summarize, the focus of this project is on data acquisition, system architecture, and the development of a functioning prototype, thereby

also implementing the results of the research group. The list below gives a short impression of the design goals of this project.

- The system can measure the different parameters needed for calculating a PASI score (needed for monitoring skin diseases).
- The system is be equipped with an inertial sensor for sensor fusion with the SLAM algorithms.
- The system embeds a stereo camera system with control over the focus, white balance, and exposure.
- The system embeds new optimization methods from prior research of the project group.
- The system can be used to acquire data that can be used to reconstruct 3D surfaces by using SLAM algorithms.
- The system is able to illuminate and project patterns onto an object/surface.

## 1.4 Related Work

As discussed in the previous section, this project is part of larger project with multiple people working on different parts of the system. The paragraphs below discuss the related prior work of the research group. The results of the prior work are used within this project to develop a better and more complete prototype system.

**3D imaging system for psoriasis assessment** Grimm (2020) worked on the design of a 3D imaging system for psoriasis assessment, and therefore Grimm's project is the predecessor of this project. Grimm designed a system consisting of two main components: an imaging system that uses two cameras, a small projector, a single-board computer as a controller, and a wireless connection. The second main component is the PC with a Matlab environment which receives the acquired data from the handheld device and processes it locally. A global impression of the system is illustrated in Figure 1.

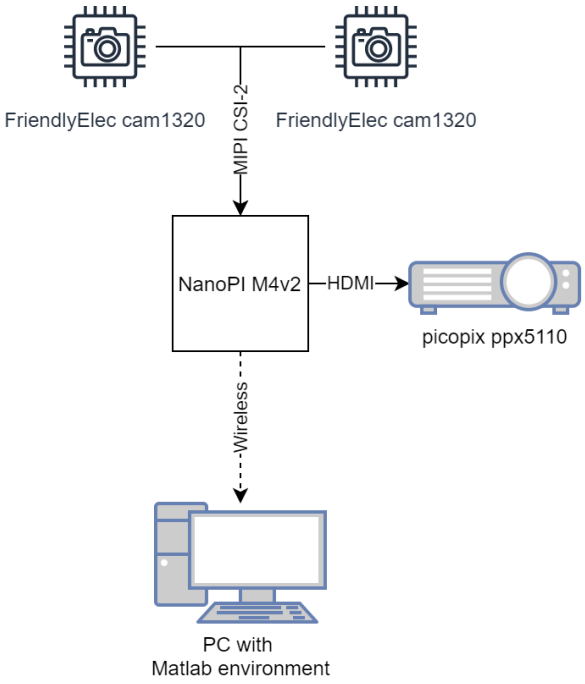



Figure 1: System architecture of the first-generation prototype

**ES-EKF-SLAM** ES-EKF-SLAM stands for Error State Extended Kalman Filter Simultaneous Localization And Mapping. Shah (2020) build a ES-EKF-SLAM algorithm formulated by Heijden (2020b) for a Matlab environment. The SLAM algorithm localizes and maps objects in 3D space while moving through that space and using that information to keep track of its position. The idea is that using a SLAM algorithm, the field of view of a camera system can be expanded so that a detailed 3D reconstruction of an objects' surface can be made. More about SLAM and how it works can be read in Section 2.3.

**GPU optimizations** The 3D imaging system for psoriasis assessment prototype of Grimm (2020) was combined with the ES-EKF-SLAM algorithm developed by Shah (2020). As described in the paragraphs above, the system works by a 3D imaging system that captures and sends stereo images to a PC SLAM server that processes the data. There was a significant bottleneck in this system approach; all images had to be wirelessly transmitted to the PC with the Matlab SLAM algorithm running. Therefore, Wolters (2021) designed a GPU optimized landmark acquisition algorithm that enables a change in the system architecture. With this new optimized landmark acquisition algorithm, the stereo images could be pre-processed on the imaging system itself and thereby reduce the data needed to be sent to the PC, thereby reducing/removing the bottleneck of the wireless communication.

Wolters (2021) also designed a GPU accelerated SLAM algorithm that performs faster than the original Matlab implementation of Shah (2020). All the optimizations are intended to get a system that can process up to ten stereo images per second.

## 1.5 Document Outline

This thesis is organized in a way that takes the reader through all steps of the development process.

**Chapter 2** After the introduction section, the background section is meant for the reader who wants more information about specific topics of importance within the rest of this thesis.

**Chapter 3** This chapter describes the design objectives and the requirements for this project which are derived from the project goal described in Section 1.3.

**Chapter 4** After the objectives and requirements are known, this information will be used in this chapter to perform a design space exploration.

**Chapter 5** In this chapter the design and implementation are discussed.

**Chapter 6** To validate the designs there are experiments performed with the system. In this chapter, the performed experiments are explained, and the results are discussed.

**Chapter 7** The results of the previous chapter are discussed in this chapter, as well as a reflection on the design goals and a list of recommendations for future work.

**Chapter 8** The last chapter summarizes the information of all chapters and gives a conclusion on the project.

# 2 Background

## 2.1 Psoriasis

Psoriasis (derived from the Greek word psora, meaning "to itch") is a common skin disease affecting an average of 1.5% of men and women, usually in the age ranges of 15-20 and 55-60 worldwide (Chiu et al., 2008; Wolf-Henning Boehncke, 2015). What causes psoriasis is unknown. Studies show that stress, smoking, alcohol, obesity, skin trauma, and medication are exacerbating factors. There are multiple forms of psoriasis, the most common condition, around 90%, is chronic plaque psoriasis (psoriasis Vulgaris), recognizable by pink or red plaques with a scaling surface. The affected skin can range in size from millimeters up to whole parts of the skin.

**Psoriasis Area Severity Index** There are various studies on Psoriasis assessment. One of them is the PASI (Psoriasis Area Severity Index) score method. A PASI-score is calculated by evaluating four parameters:

- **Erythema (redness):** Erythema is redness of the skin that results from capillary congestion. Erythema can occur with inflammation, as in sunburn and allergic reactions to drugs (*Medical Definition of Erythema* 2018).
- **Induration(thickness):** Induration is the localized hardening of the soft tissue of the body. The area becomes firm but not as hard as bone. Induration can is measured by the thickness of the lesions relative to the normal skin (*Medical Definition of Induration* 2018).
- **Desquamation (scaling):** Desquamation is the shedding of the outer layers of the skin. For example, when the rash of measles fades, desquamation occurs (*Medical Definition of Desquamation* 2018).
- **Area:** The area of the disease as ratio per body part.

The Erythema, Induration, and Desquamation are assessed using a 5-point scale, ranging from 0 "No sign" to 4 "very marked disease". A table with score images is displayed in Appendix A. The affected skin area is evaluated by classifying each anatomic area in a percentage-based class, Table 1 is used to classify the areas (Okun, 2008).

Table 1: Affected skin area scale

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 0% | $0-10\%$ | $10-30\%$ | $30-50\%$ | $50-<70\%$ | $70-90\%$ | $90-100\%$ |

When all parameters are evaluated, the PASI-score can be calculated by using Equation 1, where $E, I, D, A$ are the erythema, induration, desquamation, and area, respectively, and $h, u, t, l$ are the head, upper extremities, trunk, and lower extremities, respectively.

$$\begin{aligned} PASI = {} & 0.1(E_h + I_h + D_h)A_h + 0.2(E_u + I_u + D_u)A_u \\ & + 0.3(E_t + I_t + D_t)A_t + 0.4(E_l + I_l + D_l)A_l \end{aligned} \tag{1}$$

## 2.2 Navigation

Navigation is as old as humankind and is used in all kinds of different applications. In this list below, the most commonly used techniques are grouped into five basic navigation modes:

1. **Pilotage** relies on knowing your surroundings to estimate where you are and your orientation.
2. **Celestial navigation** uses the angle between a local vertical (e.g., the horizon) and known celestial objects (e.g., sun, moon, stars). The angle information in combination with time is enough to calculate the current position.

3. **Dead reckoning** relies on knowing the initial location, heading direction, speed, and elapsed time to calculate a path. Thus, with a correct initial position, a current position can be calculated.

4. **Radio navigation** uses radio frequencies from sources with known locations to estimate locations (e.g., GPS).

5. **Inertial navigation** is like an automated form of *dead reckoning*. It relies on an initial location, acceleration, and angular rate and uses these measurements to calculate the velocity and orientation through integration.

This section focuses on methods that can be used in the *handheld 3D skin reconstruction setup* project. One of the requirements of this project is that the product can be handheld. Also, the product must be used inside, where *celestial* and *radio* navigation is not a viable option. From this, we can conclude that the best option is to focus on *inertial navigation*.

**Inertial Navigation** One primary navigation form is inertial navigation. This form of navigation uses accelerometers and gyroscopes to calculate the linear acceleration and angular rate. By integrating the acceleration and angular rate, the translational and rotational displacement is calculated. By keeping track of the displacement, one can calculate an object's location in space only when the initial position is known.

**Accelerometers** Accelerometers are inertial sensors that measure non-gravitational acceleration, also called specific force (Grewal, Weill, and Andrews, 2001). The output of an accelerometer is called specific force because these kinds of sensors do not measure gravitational acceleration. Accelerometers depend on Newton's second law ($F = ma$) to measure acceleration ($a$) by measuring force ($F$), and the scaling constant ($m$) called "proof mass". There multiple accelerometer designs that are based on Newton's second law to measure the specific force. The IEEE standard terminology for inertial sensors (IEEE, 2019) describes the different types of accelerometers and other terminology.

**Gyroscopes** A gyroscope (a.k.a. gyro) is an inertial sensor that measures angular rotation with respect to inertial space around its input axis(es). There are multiple different implementations of a gyroscope. As with the accelerometer, The IEEE standard terminology for inertial sensors (IEEE, 2019) describes the different types of gyroscopes and other terminology.

**Inertial Measurement Units** An Inertial Measurement Unit (IMU) is a unit that combines multiple inertial sensors, such as gyroscopes, accelerometers, and magnetometers. IMU's are like an all-in-one package for doing inertial navigation. Most of the time, a barometer and temperature sensor are embedded into an IMU to calibrate and compensate for different environments. There is much terminology regarding inertial measurement units, in Appendix B some relevant terminology of IMU's from the IEEE standard terminology for inertial systems (IEEE, 2009) is given.

## 2.3 Simultaneous Localization And Mapping

Simultaneous Localization And Mapping is the action of localizing an object while simultaneously mapping the environment. SLAM is used in applications where a robot, vehicle, or person must navigate in unexplored territory. SLAM can be implemented with various techniques such as a distance measuring laser on a bearing or with the aid of visual information like a stereo camera setup. The general procedure of a visual EKF-SLAM (Solá, 2014) is as follows:

1. A moving vehicle is moving around while acquiring images.

2. The vehicle's motion is smooth such that knowledge of the past trajectory gives information about the future. Thereby it is possible to predict the next pose of the device.

3. The environment consists of static objects with enough structure and texture to detect feature points. The feature points translate to 2D/3D landmarks, which are identifiable.
4. From the measured positions, the vehicle's location can be estimated as well as the locations of new and existing landmarks.
5. A list of landmarks is updated with the positions of the landmarks.
6. The system repeats itself with the newly gathered information.

In Figure 2 a sketch of a point feature-based SLAM can be seen. The x and y axes represent the world coordinate system. Within this system is a vehicle with its own coordinate system because the range laser is mounted to the vehicle. More about EKF-SLAM can be read in the tech report of Heijden (2020a).

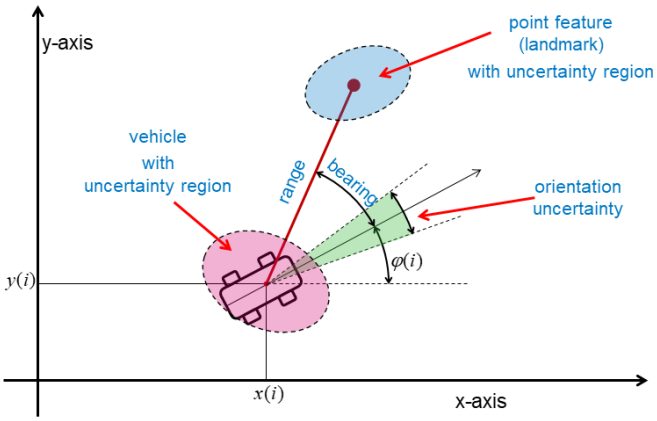



Figure 2: Point feature-based SLAM (Heijden, 2020a)

**Visual Inertial SLAM** Visual Inertial SLAM (VISLAM) is the method of adding inertial information to a VSLAM algorithm. This fusion of measurements can lead to accurate estimations of camera motion and 3D mapping of the environment (Liu et al., 2018). In this study, a method for visual-inertial SLAM is discussed. The paper focuses on using points and lines in images to improve the accuracy of the SLAM algorithm in combination with fusing IMU data. Bai et al. (2019) propose a novel feedback system for stereo visual-inertial SLAM, which also shows improvements of accuracy in contrast to a system with no IMU in the pipeline. Li and Lang (2019) published a paper about stereo-based visual-inertial optometry for SLAM. This paper discusses the implementation of an IMU and visual SLAM in one system so that both measurements can complement each other, which results in a more accurate and robust system.

# 3 Design Objectives & Requirements

Before a design can be made for the prototype setup, the design objectives and requirements must be researched and determined. From research of the prior work (Section 1.4) a list of problems and optimization opportunities has been derived. In this chapter, the different design objectives of this project are discussed. In every section, a design objective is given with the corresponding requirements. These requirements state what needs to be accomplished to reach the design objective. At the end of this chapter, the design objectives are summed up again and prioritized to utilize the MoSCoW principle (Section 3.12).

**System overview** The to-be-designed system will consist of two main parts; a handheld scanner and a main processing unit, which can run the SLAM algorithm (also called the SLAM server). In Figure 3 an abstract diagram of the system is displayed. It shows the handheld scanner, which contains all the peripherals (e.g., a stereo camera for depth measurements), a controller where all peripherals are connected to and which can perform some pre-processing of the data if necessary, and lastly, the external main processing unit, which acts as a SLAM server.

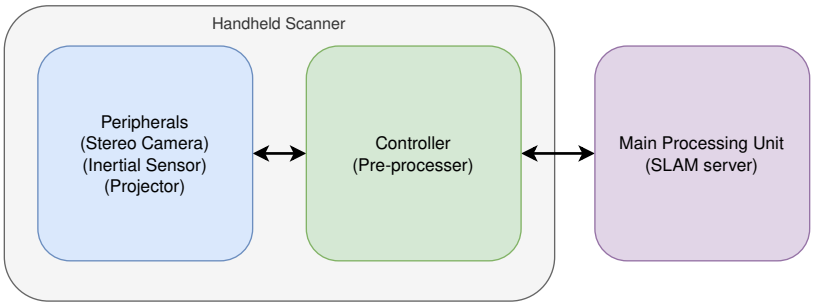



Figure 3: Abstract system overview

## 3.1 Skin Surface Data Acquisition

**Objective** The eventual goal of the research group is to monitor skin diseases in an automated fashion. For the skin disease psoriasis, the PASI assessment method is often used to indicate the severity of the condition. As discussed in Section 2.1, three parameters have to be measured at every anatomical body part to calculate a PASI score, the erythema (redness), the induration (thickness), and the desquamation (scaling) of the lesions. This topic's design goal is *Design a camera system that can measure depths and can acquire color information such that the system can measure the size, thickness, and redness of the lesions.*

**PASI erythema score** The system must be able to measure the redness of lesions, there the camera modules must contain an RGB color sensor. Research shows that the indexation of erythema in the PASI method can be done objectively (Fadzil, Ihtatho, et al., 2009). The PASI erythema score can be defined quantitatively in RGB color space (see Table 2). The results show that 8-bit RGB color depth is a high enough resolution to distinguish between the lesion colors.

Table 2: Objective PASI erythema score assessment

| PASI erythema score | Color representation | RGB color space | | |
|---|---|---|---|---|
| | | R | G | B |
| 1 | Light pink | 255 | 182 | 193 |
| 2 | Pink | 255 | 192 | 203 |
| 3 | Red | 255 | 0 | 0 |
| 4 | Dark red | 139 | 0 | 0 |
| 4 | Purple | 128 | 0 | 128 |

**PASI induration score** For the measurement of thickness, the system must be able to measure depths. A stereo camera setup can be used for this purpose. A study from Fadzil, Fitriyah, et al. (2010) with 125 PASI induration score labeled images shows the following thicknesses related to a PASI induration score (Table 3). This research concludes that a depth resolution of 0.1 mm is enough to classify the thickness of the lesions into a PASI induration score.

Table 3: Objective PASI induration score assessment

| PASI induration score | Thickness |
|---|---|
| 1 | 0.032 - 0.202 mm |
| 2 | 0.208 - 0.410 mm |
| 3 | 0.463 - 0.689 mm |
| 4 | 0.911 - 2.268 mm |

**PASI desquamation score** No studies were found that specifically described an objective method for assessing the PASI desquamation score. A study of Fink, Enk, and Haenssle (2018) showed that image processing could be used to measure the ratio of desquamation. The paper did not describe the exact method/algorithm for calculating desquamation but explained that pixel intensity was used to localize the desquamation. Because the measurement of desquamation could be done by performing image processing techniques on standard color images, there is no need for further hardware requirements.

**Requirements** In the three paragraphs above, the objective measurement of the erythema, induration, and desquamation is described. Below is a list of requirements for the system to be able to acquire PASI parameter data accurately.

- The system must consist of a stereo camera setup for depth measurements.
- The system must be able to acquire at least 8-bit RGB color image data.
- The resolution of the depth measurements should be at least 0.1 mm.
- The point density of the surface should be more than 20 $px/mm^2$.

## 3.2 Inertial Measurements

**Objective** The currently used SLAM algorithm formulated by Heijden (2020b), and build by Shah (2020) has a problem with the implementation. There is an ambiguity in the estimated rotation or translation of the localized object. The system cannot distinguish between small rotations and small linear translations. This ambiguity reduces the accuracy and stability of the system. Research of Shah (2020) has shown that, in a simulated environment, the introduction of gyroscopic information can reduce the rotation/translation ambiguity. This design goal is *Implement an inertial sensor such that the SLAM algorithm can use this data to increase its performance.*

**Requirements** This design objective is based on the hypothesis that an inertial sensor would improve the accuracy and stability of the SLAM algorithm. Because this design objective is about an experiment, no concrete requirements regarding the inertial sensor can be made. This design objective only concerns the implementation of the inertial sensor, not the processing of the inertial data into the SLAM algorithm.

- The system should at least be able to measure angular rotational data.
- The data of the inertial sensors should be synchronized with the image acquisition.

## 3.3 Surface Illumination

**Objective** For reliable imaging, a reliable light source is necessary. The final system should be able to operate in different light conditions. For this to be accomplished, an illumination unit is necessary to light up the skin surface. Furthermore, the system must be able to project patterns on the surface. The design goal is *Implement an illumination unit that can light up the skin surface and project patterns onto it.*

**Requirements** The requirements for this design goal are not concrete, but they describe the need for the system. The wanted behavior of the system is that the captured images are sharp with low noise. The requirements below contribute to this goal.

- The illumination must be bright enough for the camera modules to have an exposure time of less than $1ms$.
- The illumination unit must be able to project patterns with a pixel resolution of $854\,\text{px} \times 480\,\text{px}$, as used in the research of Grimm (2020).

## 3.4 Camera Sensor Control

**Design Objective** Before one can apply stereo image reconstruction, one has to calibrate the cameras. Camera calibration is done to estimate the intrinsic, extrinsic, and distortion coefficient parameters of a camera. These camera parameters are used to remove distortion in the image and to calculate 3D landmarks by triangulation. In the design of Grimm (2020), the camera modules have a magnetic focus system controlled by an auto-focus instruction given by a driver. The problem with the magnetic focus is that the focus will be lost whenever power is discontinued. The change in focus means that the camera has to be re-calibrated every time the device is used. It is highly impractical and time-inefficient to calibrate the camera every time it is used. Therefore the design goal is *Implement camera modules that give manual control over the focus, white balance, and other camera parameters to improve the reproducibility and stability of the system.*

**Requirements**

- The focus of the camera module must be controllable and must not change when not specifically altered.
- The focus must keep its position when not specifically altered.
- The image acquisition of the two cameras must be done simultaneously.
- The white balance of the camera modules must be controllable.
- The exposure of the camera modules must be controllable.

## 3.5 Visual Inertial EKF-SLAM

**Design Objective** As already discussed in Section 3.2, an inertial sensor can be used to reduce the rotation/translation ambiguity of the visual EKF-SLAM. Where Section 3.2 describes the need for an inertial sensor (hardware), this design objective is about actual sensor fusion between the IMU data and the current visual SLAM algorithm (software). The design goal is *Embed the inertial sensor data into the visual EKF-SLAM algorithm to reduce the translation/rotation ambiguity.*

**Requirements**

- The inertial data must be embedded into the current SLAM algorithm.

## 3.6 Embedded Software Pipeline

**Design Objective** Within the research group, a lot of new methods and optimizations are developed. In the current prototype, these new methods and optimizations are not implemented. As discussed in Chapter 1.4, Wolters (2021) designed a GPU optimized landmark acquisition algorithm and a GPU optimized EKF-SLAM algorithm. This optimization uses GPU CUDA cores to perform fast matrix multiplications, speeding up the SLAM algorithm's computation time. The system architecture is also optimized by introducing a single board computer with GPU at the system's handheld device side. This change of architecture enables the system to perform pre-processing at the client-side. Performing pre-processing at the client-side reduces the data throughput because not all images but only key-points descriptors with color data have to be sent to the server. The design goal is to *Design a system such that the GPU optimizations can be implemented and that a working system pipeline can be constructed.*

**Requirements**

- A GPU must be available in the handheld device.
- The stereo images from the data acquisition must be ported to the landmark acquisition algorithm.
- The inertial data for the data acquisition and landmarks must be sent to the SLAM server.
- A dedicated PC/server must be built for the SLAM algorithm to run on.

## 3.7 Cost Effective Design

**Design Objective** This project is part of a preliminary study. The funds for such a project are limited. Therefore, aside from designing a prototype skin diseases monitor system for experimental use, this project aims to create a proof of concept used to acquire further research and development funds. The design goal is *Design a system that will fit within the given budget.*

**Requirements**

- The cost for this project should be around €1500 to €2000.

## 3.8 Noise Patterns Projection

**Design Objective** Grimm (2020) introduced a method for reducing the adverse effects of specular reflection with respect to stereo reconstruction. The hypothesis was that acquiring two stereo image sets with negative noise patterns with respect to another makes the stereo reconstruction more robust than using a single stereo image set without noise. The results for testing the hypotheses were promising. Therefore it is interesting to implement this technique in this project as well. This topic's design goal is *Implement the noise pattern projection technique such that the image data is more robust to reflections on the surface/object.*

**Requirements**

- The pattern projection must be synchronized with the image acquisition.

## 3.9 Fast Imaging

**Design Objective** For a scan to be successful, the user must stand as still as possible while acquiring the image sets. Depending on the patient's condition and age, standing still for more extended periods can be challenging. Therefore the system must be fast in its image acquisition. The design goal is *Design a system that can acquire images fast so that the patient's discomfort is kept to a minimum.*

**Requirements** Within the research of Grimm (2020) the maximum imaging time is set to be a maximum of 3 s. For this project, the requirement will be copied. For a better, more precise requirement for this design goal, more research has to be done. Researching this design goal at this stage of the project takes much time because there is no actual functioning prototype yet.

- One complete set of images should be acquired within 3 s.

## 3.10 Handheld System

**Design Objective** When the system is portable and can be handheld, the device can be easily manipulated and used to scan body parts with ease. The design goal here is to *Design a lightweight system such that it can be handheld.*

**Requirements** A handheld device's weight should be less than 2.3 kg (*Human Factors criteria for hand held devices* 2004). In some cases, the limit of 2.3 kg is lower. This depends on the amount of precision that is needed to manipulate the handheld device. The lower limit given by the standard that describes the criteria for handheld devices is 400 g. After some qualitative (personal) experiments with some weights is determined that the right amount of weight should be around 600 g. Note that this experiment has many limitations as it is only the input of one single person.

- The weight of the device must be at most 600 grams.

## 3.11 Wireless System

**Design Objective** A wireless system is more portable than a system that has to be connected to a PC by cables. The design goal here is *Design a wireless system, both power, and communication wise.*

**Requirements**

- The system must have a stable wireless connection with the SLAM server.
- All data has to be transmitted successfully, e.g., no loss of data.
- the system should be battery powered.

## 3.12 Prioritization

The design objectives described above are categorized into four classes following the *MoSCoW method* (Rouse, 2020). MoSCoW describes the priority of an objective/requirement. Must have objectives have to be reached for the project to be a success. Should have objectives are important for the completion of the project, but are not necessary. Could have objectives are nice to have features to the project, but have a smaller impact when left out of the project. Lastly, Will Not objectives are ideas or requirements that are acknowledged but are left out of the project. The objectives of this project are shown below in Table 4.

Table 4: Prioritized Design Objectives

| Nr. | Priority | Design Objective |
|---|---|---|
| 1 | Must | Design a camera system that can measure depths and can acquire color information such that the system can measure the size, thickness, and redness of the lesions. |
| 2 | Must | Implement an inertial sensor such that the SLAM algorithm can use this data to increase its performance. |
| 3 | Must | Implement an illumination unit that can light up the skin surface and project patterns onto it. |
| 4 | Must | Implement camera modules which give manual control over the focus, white-balance, and other camera parameters to improve the reproducibility and stability of the system. |
| 5 | Should | Embed the inertial sensor data into the visual EKF-SLAM algorithm to reduce the translation/rotation ambiguity. |
| 6 | Should | Design a system such that the GPU optimizations can be implemented and that a working system pipeline can be constructed. |
| 7 | Should | Design a system that will fit within the given budget. |
| 8 | Could | Implement the noise patter projection technique such that the image data is more robust to reflections on the surface/object. |
| 9 | Could | Design a system that can acquire images fast so that the patient's discomfort is kept to a minimum. |
| 10 | Could | Design a lightweight system such that it can be handheld. |
| 11 | Could | Design a wireless system, both power, and communication wise. |

# 4 Design Space Exploration

In this chapter, different possibilities for solving the project's challenges are discussed and rated employing a design space exploration (DSE). A DSE explores the properties of alternative solutions regarding one subject by comparing them to a list of criteria with each a specific weight. Designs will be awarded a score from a symbolic scale, which indicates, by using '+' or '-' symbols, how well they fit the specific criteria. Next, the total score of a design/solution will be calculated by $\sum_n^N w_n \delta_n$ where $N$ is the number of criteria, $w_n$ is the weight of the n-th criteria, and $\delta_n$ is the symbolic score of the n-th criteria which is mapped to $(---, --, -, N/A, +, ++, +++) \mapsto (-3, -2, -1, 0, 1, 2, 3)$. The criteria and their weights set for this design space exploration are given in the paragraphs below.

**Accuracy (weight 12)** The accuracy of the system is of most importance. When the system lacks accuracy, the results cannot be used. Giving the accuracy criterion a weight of 12 ensures that the accuracy is heavily weighted in the designs' decisions.

**Implementation time (weight 10)** The time necessary to implement a system or idea is of great importance in this project. Time is limited to 1120 hours in total, which includes all paperwork, meetings, and presentations. The time available cannot be extended. Therefore a weight of 10 is set on this criterion is.

**Expandability (weight 7)** The different functionalities and features of a solution are important when looked at the expandability of the system. This criterion has been set at a weight of 7 because this criterion enables further research and development on the prototype.

**Reliability (weight 5)** A system that does not operate reliably is a hassle to work with, and results may become useless. This criterion is essential, but because this project is regarding a prototype, reliability is not the most critical factor. Therefore the weighting of the reliability criteria is set to 5.

**Cost (weight 4)** The cost criterion has two sides, one being the available budget of this project, the other being the cost for the end consumer of the eventual final product. For the budget this means, the components cannot be too expensive, or else the RAM research group cannot afford to spend that amount on money on this project. For the component costs, this means that no high-end components will be used, as this would make an eventual end product too expensive. Therefore, this criteria guards the price of the product and is weighted at 4.

**Mobility (weight 3)** The eventual product must be lightweight, small in size, and easy to transport, such that people can use the product at home and can store it somewhere convenient. This criterion is essential to the eventual final product, but not that much for this prototype version. Therefore, the weight of this criteria is set to 3.

## 4.1 Navigation Method

One of this project's primary goals is to introduce navigation. This data can be used to improve the accuracy and stability of the current used ES-EKF-SLAM algorithm. In this section, multiple designs for acquiring navigational data are discussed. Some of these techniques are more common than others, but this section is specially made to give some inside into other non-common techniques for performing navigation of a handheld device. The three paragraphs below show alternative designs for navigation.

**Inertial Measurement Unit** An IMU, short for Inertial Measurement Unit, is a unit containing different inertial sensors, like gyroscopes and accelerometers. IMU's are used in applications where rotation and linear displacement have to be known, such as a drone that has to use this kind of data to stabilize itself in the air. An IMU gives acceleration, angular rate, and sometimes also temperature and magnetic data. The angular velocity has to be integrated to get an orientation. Acceleration data has to be double integrated to get a position. These calculations have to be done externally as the IMU only gives the measurements. Some sensors provide an all-in-one solution. They measure accelerations and angular rates, then directly integrate the measurements to offer positions. The problem with inertial navigation is that uncertainty of the results, without external references, will grow over time because the uncertainty errors accumulate.

**Beacon-Based Navigation** Beacon-based navigation (a.k.a. lighthouse navigation) is an extended form of only using IMU sensors. With this kind of navigation, an external reference with a fixed location is needed. The idea is that the to-be-tracked device can detect the fixed beacon(s) and use those beacons to determine its position and orientation in space with respect to the world coordinates. This type of tracking is used in virtual reality equipment. For example, the SteamVR™ Tracking (Steamworks, n.d.) is a VR tracking system that works with one or two "lighthouses" on a fixed location which emits light. The to-be-tracked object has multiple photosensitive diodes place on the exterior of the object. The lighthouse beacon uses an array of infrared rays to synchronize all trackable devices. Next, the lighthouse emits a rolling fan of infrared laser light among the horizontal and vertical axis alternatively. The to-be-tracked device can calculate its pose in 6-DOF by calculating the time and angle of impact between the different photosensitive diodes. This navigation method is more accurate than using only an IMU sensor because there is no uncertainty accumulation. On the other hand, the implementation is much more complicated and less portable because of the needed external beacons.

**Motorized Navigation** In the world of quality assurance, a measurement arm is sometimes used for validating product dimensions. This measurement arm is constructed with motorized arms with servo motors that can be manipulated simply by moving the arms. Because servos control all joints of the arm, the joints' angles are known. This enables the device to precisely calculate the tool's orientation and location. This kind of design can be used in this project as well. When the camera setup is connected to the motorized arm, the camera's location will be known at all times. The drawback of this idea is that the arm is anything less than portable, and the system can become expansive and harder to implement.

Table 5: Navigation Method Exploration

| **Criteria** | **Weight** | **Inertial Measurement Unit** | **Beacon Based Navigation** | **Robotic Navigation** |
|---|---|---|---|---|
| Accuracy | 12 | ++ | +++ | ++++ |
| Implementation time | 10 | +++ | + | - - |
| Expandability | 7 | N/A | N/A | N/A |
| Reliability | 5 | +++ | ++ | ++ |
| Cost | 4 | +++ | ++ | + |
| Mobility | 3 | +++ | ++ | - |
| **Total Score** | | **90** | 70 | 39 |

## 4.2 Embedded Controller

The second system component design that has to be explored is the embedded controller. The embedded controller is responsible for reading inertial sensor data, acquiring stereo images, controlling the illumination unit, pre-processing data, and sending it to a server where the SLAM algorithm runs. Multiple system architectures would work in this project with all different embedded controller

types, each with its advantages and disadvantages. In the paragraphs below, multiple embedded controllers are described with their advantages and disadvantages.

**Microcontroller** The first embedded controller that will be considered is a microcontroller. A microcontroller is an integrated circuit containing a microprocessor, RAM, ROM, timers, clocks, and some I/O ports. A micro-controller is cheap, has a small form factor, and can be compared to a PC integrated into one chip but with less functionality and performance. Using a microcontroller as an embedded controller in this project can be a solution, but (pre-)processing data like key-point matching of the stereo images and integrating the IMU data is impossible or too computational expansive for the low-end hardware.

**Single Board Computer** A single-board computer (SBC) works a lot like a micro-controller but has better hardware and works with a complete operating system like a Linux distribution. The cost of a single board computer is much higher than a micro-controller, but as said before, the hardware is better and is more feature-rich. A single board computer can be easily implemented in a prototype like this. Most systems are well documented, and SBC has a variety of features that make implementation easier.

**Single Board Computer with GPU** The single-board computer with a GPU (graphics processing unit) works the same as the regular SBC, but with an onboard GPU. The addition of a GPU enables the system to do specific processes faster, like performing key-point matching or stereo image reconstruction. This implementation also allows Wolters (2021) landmark finding algorithm to be implemented in the design, reducing the communication bottleneck between the hand-held device and the SLAM-server. The implementation time is comparable with that of the SBC, but it gives more freedom to expand the system in the future.

**FPGA** An FPGA is a Field-Programmable Gate Array and is used in applications where and fast throughput and low latency are essential. Rather than having a general processor like the other options have, an FPGA is build and programmed for a specific application. Using an FPGA would be beneficial when looking at acquiring data at high speeds, but the processing done at an FPGA is more limited and complicated, and therefore more time-consuming.

**No Embedded Controller** The last option for an embedded controller design is not to use any. It is possible to run all cables from the sensors and the illumination unit to a PC/server where all the processing and control are done. This makes the device lighter and faster because more resources are available on the server-side. One of this approach's downsides is that the system is not wireless and that long cables are necessary. The introduction of long cables may result in problems. Long cables are not desirable for fast data transfer because they have more electrical resistance and are more prone to noise interference. Counteracting this problem introduces more error detection and correction methods, which will reduce the data throughput. The long cables will reduce the system's mobility.

Table 6: Processing Unit Exploration

| Criteria | Weight | Micro Controller | Single Board Computer | SBC with GPU | FPGA | No Embedded Controller |
|---|---|---|---|---|---|---|
| Accuracy | 12 | N/A | N/A | N/A | N/A | N/A |
| Implementation time | 10 | + | ++ | ++ | - | +++ |
| Expandability | 7 | + | ++ | ++++ | ++ | - |
| Reliability | 5 | ++ | ++ | ++ | +++ | - |
| Cost | 4 | +++ | ++ | + | + | +++ |
| Mobility | 3 | ++ | ++ | + | ++ | - - |
| **Total Score** | | 45 | 58 | **65** | 29 | 24 |

## 4.3 Stereo Camera

For this project, a set of two cameras is necessary to perform stereo camera imaging. The camera modules must comply with some requirements for the design objectives to be achieved. Hence, some of the camera modules' design choices are fixed, like the use of RGB sensors for acquiring color information, the use of small CMOS cameras instead of, for example, a sizeable DLSR camera. There are some choices to be made when choosing a type of camera module: the camera's communication interface. In the paragraphs below, two different communication options are discussed.

**USB Cameras** An USB camera can be easily implemented in most applications. USB cameras are available in a wide range of specifications, and most of the time, they are low-cost.

**CSI Camera** A camera module with a CSI (Camera Serial Interface) communication has a higher bandwidth than standard USB camera modules. The USB interface is used for communication between devices and hosts of a variety of devices. On the other hand, CSI is specially made for camera sensor communication between image sensors and hosts. In most embedded controllers, the CSI interface is connected to an ISP (image signal processor). The IPS runs separately from the CPU and therefore does not influence the CPU performance. Some embedded controllers have the ISP located on the GPU, giving it even more freedom. All this parallelization is not the case with USB, as the CPU needs to read and write to the USB drivers.

Table 7: Camera Type Exploration

| Criteria | Weight | USB Cameras | CSI Cameras |
|---|---|---|---|
| Accuracy | 12 | + | ++ |
| Implementation time | 10 | + | + |
| Expandability | 7 | + | + |
| Reliability | 5 | + | ++ |
| Cost | 4 | ++ | + |
| Mobility | 3 | + | + |
| **Total Score** | | 45 | **58** |

## 4.4 Inertial Sensors

An inertial sensor is necessary to reduce the currently used slam algorithm's translation rotation ambiguity as explained in Section 3.2. There are multiple ways that inertial data can be acquired in the system. In the paragraphs below, five alternative solutions are given.

**MEMS Gyroscope IC** The first inertial sensor to be considered is the MEMS gyroscope. Shah (2020) used in his experiments a gyroscope model in a simulated environment to reduce the ambiguity in the SLAM algorithm. This experiment showed a positive effect on the SLAM algorithm. 3D gyroscopes are available in small form factor IC's and contain three gyroscopes positioned orthogonal to each other's principle axis. Because the MEMS gyro IC's are in a small package, they cannot be directly connected to an embedded controller. It is needed that the sensor is soldered onto a PCB in combination with some headers so that it can be connected to an embedded controller.

**MEMS Inertial Measurement Unit IC** An expansion of the MEMS gyroscope IC is the MEMS IMU IC. A 3D IMU contains multiple inertial sensors such as three gyroscopes, three accelerometers, and other possible sensors like a magnetometer, barometer, or temperature sensors. A gyroscope can only be used to measure angular rates and thereby calculate the orientation. An IMU enables calculating the pose in 6 degrees of freedom. A MEMS IMU IC has the same form-factor as MEMS gyroscope IC, and the price between the two sensors does not differ that much. Because of the same form-factor as the gyro, the IMU needs to be soldered onto a PCB before it can be connected to the embedded controller.

**Industrial-grade Inertial Measurement Unit** The industrial-grade IMU is a more accurate, easy to implement, reliable, but more expansive alternative to the MEMS IMU IC. An industrial-grade IMU has connectors and communication protocols such that a sensor can be implemented more efficiently. These kinds of sensors can also calibrate measurements by correcting the data by temperature and magnetic environmental interference. Most industrial-grade sensors are larger than their IC counterparts. This makes it less portable but makes it more robust.

**Industrial-grade Vertical Reference Unit** The data of an IMU has to be integrated (angular rate ones, acceleration twice) to get 3D pose information. The integration calculations have to be performed on an external processing unit, which can be quite a computationally heavy task when dealing with high sample frequencies. A vertical reference unit (VRU) is, in principle, an IMU with an onboard processor that calculates the pose such that the data available is directly the pose data. Acquiring pose data from a sensor reduces the implementation time with respect to IMU data, but this comes with a cost. VRU systems are more expansive than IMU systems.

**Industrial-grade Attitude and Heading Reference System** As last is the Attitude and Heading Reference System (AHRS). This sensor system is an upgrade to the VRU system. The VRU calculates the position and orientation of itself with respect to world coordinates. A VRU can calculate the roll and pitch with respect to the gravitational force, but the yaw rotation is with respect to its initial position. An AHRS does the same thing as a VRU, but the yaw rotation is with regard to the earth's magnetic north. In this way, the yaw movement is not dependent on the initial orientation but is always with respect to the north.

Table 8: Inertial Sensor Type Exploration

| **Criteria** | **Weight** | **MEMS Gyro IC** | **MEMS IMU IC** | **Industrial IMU** | **Industrial VRU** | **Industrial AHRS** |
|---|---|---|---|---|---|---|
| Accuracy | 12 | ++ | ++ | +++ | +++ | +++ |
| Implementation time | 10 | + | + | ++ | +++ | +++ |
| Expandability | 7 | + | ++ | ++ | +++ | ++++ |
| Reliability | 5 | + | + | ++ | +++ | +++ |
| Cost | 4 | +++ | +++ | + | - | - - |
| Mobility | 3 | +++ | +++ | ++ | ++ | ++ |
| **Total Score** | | 67 | 74 | 90 | 104 | **107** |

Table 9: Design Space Exploration configurations based on different criteria

| Configuration | Navigation Method | Embedded controlelr | Camera Module | Inertial Sensor | Total Score |
|---|---|---|---|---|---|
| Best overall score | IMU only | SBC /w GPU | CSI camera | Industrial AHRS | 320 |
| Second best overall score | IMU only | SBC /w GPU | CSI camera | Industrial VRU | 317 |
| Most cost effective | IMU only | Micro-controller | USB camera | MEMS Gyro IC | 247 |
| Most portable | IMU only | Micro-controller | USB camera | MEMS Gyro IC | 247 |
| Most expandable | IMU only | SBC /w GPU | CSI camera | Industrial AHRS | 320 |
| Most time efficient | IMU only | No controller | USB camera | Industrial AHRS | 279 |

## 4.5 Configurations

From the results gained in the design space exploration performed in the sections above, a list (shown in Table 9) with configurations can be derived, which offers alternative configurations that each suite a different need. In this concluding section, some interesting configurations are discussed, such as the configuration with the best and second-best overall score. Also, designs that win on a single criterion like most cost-effective are discussed.

**Best overall scores** The best and second-best overall score configuration only differ 3 points. The difference between them is the inertial sensor. The best overall score configuration is equipped with an industrial AHRS sensor, the second-best packs an industrial VRU sensor. The slight difference comes from the AHRS sensor being slightly more expensive but has more capabilities than its VRU counterpart. Next to the inertial sensor, the best overall scoring configurations uses a single-board computer combined with two CSI cameras. This configuration enables the system to embed the GPU optimized landmark acquisition as described at Chapter 1.4. It also gives more freedom for further development of the system.

**Most cost-effective / Most portable** When the cost of the design is only taken into consideration, the system design will consist of a micro-controller, USB camera modules, and a MEMS Gyro IC. Using a micro-controller requires some more components to work, like, power regulators, hardware for controlling the illumination unit, connections for cameras, and a PCB design to connect all the hardware. Because all sub-systems can and have to be designed for this specific application, the system can be made very compact. Going for this configuration is not recommended at this time. The state of development is still preliminary, and time and effort should first be put into a working prototype. The goal is to get a working prototype that can be used for testing and optimization. Focusing on mobility and costs at this point misses the project goal of designing a prototype system that could function as a test bench.

**Most expandable** The most expandable configuration is the most future-proof configuration. This configuration includes the most extensive feature sets of each chosen design. The most expandable configuration gives the most possibilities for further development and testing of the system. As can be seen, the best overall score configuration is the same as the most expandable configuration. This is because the expandability is weighted a 7 and thus more critical than mobility and cost. Also, this is because the expandability is indirectly correlated to accuracy and reliability in this case.

**Most time efficient** The last configuration that has been taken into consideration is the most time-efficient configuration. As said before at the beginning of Chapter 4, time is of the essence in this project because there is no more time available than the 1120 hours given. Therefore it might be wise to go for the most time-efficient configuration that embeds not controller, the best scoring inertial sensor, USB cameras, and wire to an external server with enough processing power and IO. This might be the most straightforward option, but this results in much less of a challenge.

The configuration's mobility is the worst of all designs because of all the cables running from the hand-held device to the external server. Therefore this configuration does not fit this project.

**Conclusion** A design space exploration is designed to find the best configuration, including all different kinds of criteria. Therefore, the best overall scoring configuration should be the configuration that fits this project the best. This configuration will be used in the preliminary design, and components that match the configuration will be chosen.

Table 10: Final Design Configuration

| **Sub-system** | **Design** |
|---|---|
| Navigation Method | Inertial measurement unit only |
| Embedded Controller | Single board computer with on-board GPU |
| Camera Modules | CMOS camera modules with CSI interface |
| Inertial Sensor | Industrial Altitude and Heading Reference System |

# 5 Design & Implementation

This chapter describes the design and the implementation of the hardware and software used in the prototype system. The designs discussed in this chapter are derived from the results of the design space exploration discussed in the previous chapter.

**System overview** Before discussing all components and implementation separately, it is better first to understand how the system works and what kind of components there are in the system. As described in the abstract system overview of Chapter 3, the whole system exists of two sub-systems; a handheld scanner and a SLAM server. Figure 4 shows a block diagram of the entire system. The colors in the diagram indicate what components have the primary focus, what have the secondary focus, and which parts already exist and are just implemented.

On the top side of the figure, the handheld scanner is shown. There are three peripherals within the handheld scanner: a stereo camera setup, an inertial sensor, and an illumination unit. The peripherals are connected to an embedded controller which controls the peripherals and gathers the information from the sensors. The embedded controller synchronizes all the collected data before processing it. The processing of the data on the handheld device consists of the undistortion of images and a landmark acquisition algorithm. The resulting data (3D landmarks and orientation data) from the processing is sent wireless to the SLAM server. The SLAM server receives the data for the handheld scanner and processes this data through a SLAM algorithm. This will result in a 3D surface reconstruction.

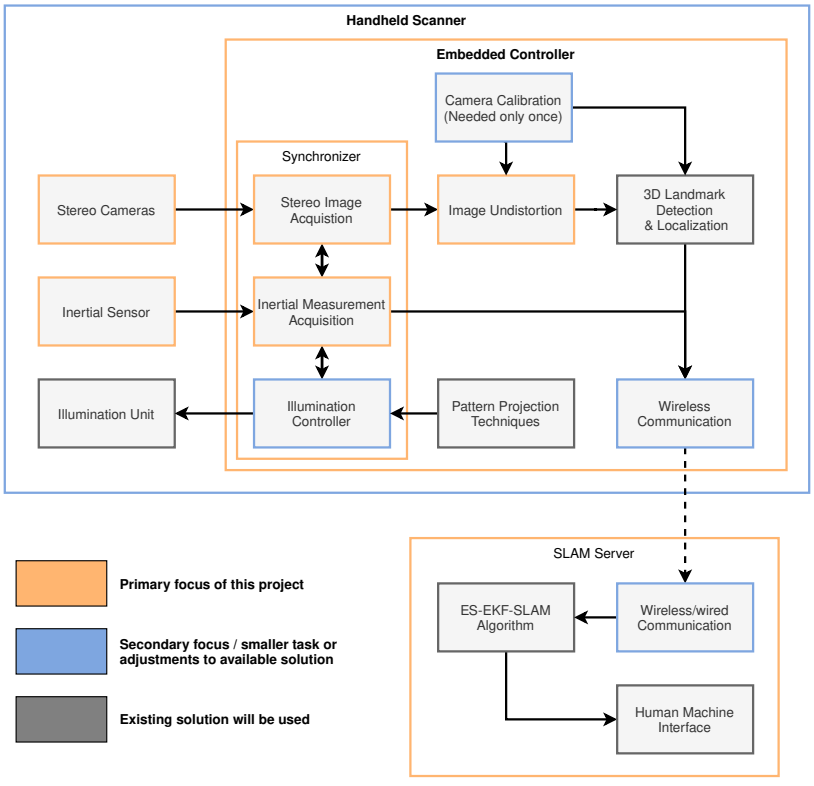



Figure 4: System overview with focus indication

## 5.1 Embedded Controller

The first component to be discussed is the embedded controller. The controller is the center of the whole device as all other components are connected to it. Therefore, choosing a controller also determines the features and available interfaces of the other components.

The embedded controller type chosen in the DSE is a single board computer with an onboard GPU. The onboard GPU allows for embedding the GPU optimized landmark acquisition software designed by Wolters (2021). Implementing the landmark acquisition on the embedded controller provides for a reduction in transmission data, as only the results of the landmark acquisition have to be sent to the SLAM server instead of full stereo camera images.

The market for single-board computers with onboard GPUs that contain the needed CUDA cores is limited. One supplier of these kinds of systems is Nvidia with their Jetson series. In Table 11 four Jetson versions are shown. The currently used and available version is the Jetson Nano (4GB). From conversations with Wolters, it can be concluded that Jetson Nano has its limits; the performance of the Jetson Nano can limit a high throughput when dealing with larger image sizes (such as the 13.1 Mpx camera used in the first-generation prototype). If the budget does not play a part, the recommendation is to purchase a better version of the Jetson, for example, the Jetson Xavier NX. The table below gives a quick impression of the Xavier series's performance increase compared to the Nano series. The Xavier NX uses a 6-core processor instead of a quad-core, has triple the amount of CUDA cores, the RAM is doubled, and it has 48 extra Tensor cores for fast matrix multiplication instead of zero. The products shown below are all development-kit versions and Nvidia recommended prices (without taxes and shipping).

Table 11: Nvidia Jetson Comparison

| **Specifications** | **Jetson Nano 2GB** | **Jetson Nano** | **Jetson Xavier NX** | **Jetson AGX Xavier** |
|---|---|---|---|---|
| CPU | quad-cores | quad-cores | 6-core | 8-core |
| CUDA cores | 128 | 128 | 384 | 512 |
| Tensor cores | 0 | 0 | 48 | 64 |
| RAM | 2 GB | 4 GB | 8 GB | 32 GB |
| Vision Accelerator | No | No | Yes | Yes |
| Camera | 1x MIPI CSI-2 | 2x MIPI CSI-2 | 2x MIPI CSI-2 | 16x MIPI CSI-2 |
| Display | HDMI | HDMI | HDMI & DP | HDMI2.0 |
| Price | $59.00 | $99.00 | $399.00 | $699.00 |

The Nvidia Jetson Xavier NX Dev Kit has access to two 2-lane CSI ports for the two camera modules, it has an HDMI port for the illumination unit, an m.2 slot for WiFi or other extension modules, and it has a 20x2 GPIO header with support for various communication protocols like $I^2C$, SPI, and UART. Concluding, this embedded controller fits this project the best in case of performance. However, its already available smaller brother has almost the same connectivity and will, except for the lack of better performance hardware, function the same.

To keep the total costs of the project low, there is chosen to use the available Jetson Nano (4 GB). Because the Jetson Nano is just a lower-performing device than the Jetson Xavier NX, this choice keeps the cost down at this stage of the research and allows for an "easy" upgrade in the future.

## 5.2 Camera Sensors

The system uses camera modules to capture stereo images. Stereo camera images can be used to calculate depth in images with the use of triangulation. For the camera modules, four different CSI-type camera modules were compared. In principle, the selected camera modules use only two different sensors, the IMX219, and the IMX477. These sensors are 8 Mpx and 12.3 Mpx sensors, respectively. The first camera module, the Raspberry Pi Camera Board V2, is the successor of the 5 Mpx camera module V1 from the Raspberry Pi foundation. This module is a cheap single 8 Mpx camera with a CSI interface with a fixed focused lens, which can be manually adjusted. The Raspberry Pi camera module drivers enable the adjustment of the white balance and the sensor's exposure. The successor to the V2 is the Raspberry Pi HQ camera module. This module is a 12.3 Mpx sensor without a lens. The lens has to be purchased separately; the mount for the lens is compatible with industrial C and CS-mounted lenses. The IMX219-83 Stereo Camera uses the same

sensor as the Raspberry Pi V2 module. This camera module is already configured as a stereo camera setup with a fixed baseline. It can be advantageous to have a stereo camera system right out of the box because of its simplicity, but this removes the ability to change the baseline distance or camera angles. For a device that serves as an experiment setup, these kinds of limitations are undesired. The last camera module is a modification of the Raspberry Pi HQ. This version from Arducam uses the same sensor as the HQ but already includes a small fixed lens. Using the Arducam module is more convenient and cheaper than a large C-CS-mounted lens.

Table 12: Camera Module Comparison

| **Specifications** | **Raspberry Pi Camera board V2** | **Raspberry Pi HQ Camera board** | **IMX219-83 Stereo Camera** | **Arducam MINI HQ 12.3MP IMX477** |
|---|---|---|---|---|
| Number of active pixels | 3280x2464 (8 Mpx) | 4056x3040 (12.3 Mpx) | 3280x2464 (8 Mpx) | 4056x3040 (12.3 Mpx) |
| Focal length | 3.04 mm | Depends on lens | 2.6 mm | 3.9 mm |
| Pixel size | 1.12 $\mu$m | 1.55 $\mu$m | 1.12 $\mu$m | 1.55 $\mu$m |
| AOV horizontal | 62.2 ° | Depends on lens | 83 ° | 75 ° |
| AOV vertical | 48.8 ° | Depends on lens | 73 ° | |
| Price | € 23.13 | € 46.38 | € 49.99 | € 55.08 |

From the four options mentioned in Table 12 there are two that would best fit this project, these are the *Raspberry Pi Camera Module V2* and the *Arducam MINI HQ*. At first sight, the 12.3 Mpx Arducam MINI HD looks like the best option in this comparison because of its larger sensor size. However, the Raspberry Pi camera module v2's smaller-sized camera sensor may not be such a disadvantage. With larger-sized sensors comes more data, which has to be processed, which puts more load on other components/sub-systems. Therefore if the 8 Mpx Raspberry Pi camera module complies with the set requirements from Section 3.1, this is the better solution for this project because it is cheaper than its counterpart.

**Camera control** Camera parameters as the exposure time, white balance, and focus need to be adjustable according to the requirements of Section 3.4. The Raspberry pi camera module v2 is equipped with a manually adjustable focus, making sure that the focus does not unintentionally change. In addition, the Linux drivers enable the control of other camera parameters like white balance, exposure time, digital gain.

**Pixel density** For the system to reach the *Skin Surface Data Acquisition* design goal, the requirement is that the density must be higher than 20 px/mm$^2$ at the skin surface. The pixel density for a specific camera module at a specified working distance can be calculated. In Figure 5 the geometry of a two-axis pinhole camera model is shown. This model of a camera can be used to calculate the pixel density at specific working distances. First, the ratio between the focal length and the working distance has to be calculated; this can be done with Equation 2. Next, the number of pixels per millimeter in both the horizontal and vertical direction has to be calculated; this can be done by Equation 3. Finally, the pixel density of a square millimeter at a specific working distance is given by Equation 4.

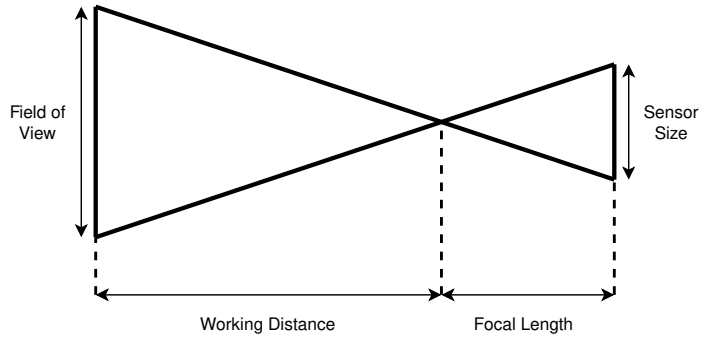



Figure 5: Geometry of a two axis pinhole camera model

$$ratio = \frac{workingDistance}{focalLength} \tag{2}$$

$$pixelDensity = \left\lfloor \frac{1}{pixelSize \cdot ratio} \right\rfloor \tag{3}$$

$$pixelDensity_{area} = pixelDensity_{horizontal} \cdot pixelDensity_{vertical} \tag{4}$$

Figure 6 shows the result of the above calculations with given the specifications of the Raspberry Pi camera module V2 with working distances varying from 100 mm to 600 mm. The dashed line at the bottom of the plot indicates the minimum pixel density of 20 px/mm$^2$.

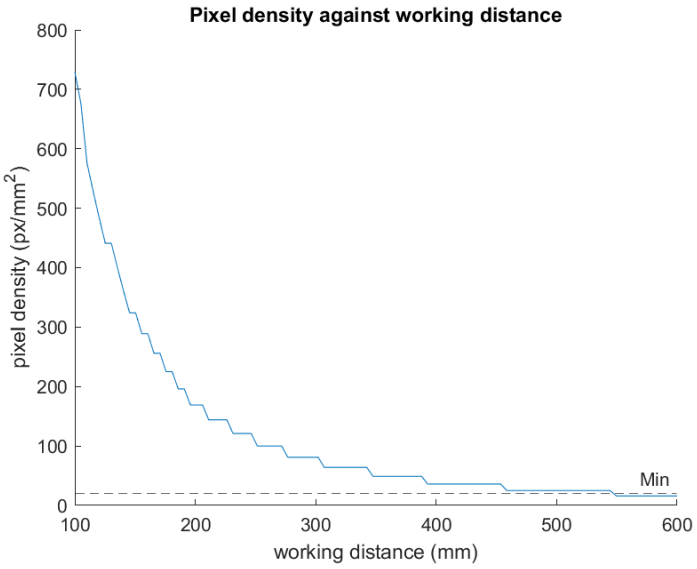



Figure 6: Pixel density against working distance

The results from Figure 6 show that the Raspberry Pi camera module v2 meets the requirement of 20 px/mm$^2$ from Section 3.1 if the working distance is not larger then 550 mm. To put this working distance in contrast with previous research, Grimm (2020) used a 230 mm working distance in his research. Thereby, it can be concluded that the Raspberry Pi camera module V2 with its 8.08 Mpx is adequate. Therefore, the more expansive Arducam MINI HQ12.3MP camera is unnecessary, and the Raspberry Pi camera module fits this project well. In Table 13, the detailed specifications of the Raspberry Pi Camera Module V2 are shown.

Table 13: Raspberry Pi Camera Module Detailed Specifications

| Property | Value |
|---|---|
| Image sensor | Sony IMX219 PQ CMOS |
| Focus mechanism | Fixed-focus manual adjustable |
| Sensor size | 3.68 mm (H) × 2.76 mm (V) (4.6 mm diagonal) |
| Number of active pixels | 3280 px (H) × 2464 px (V) ≈ 8.08 Mpx |
| Pixel size | 1.12 $\mu$m (H) × 1.12 $\mu$m (V) |
| Focal length | 3.04 mm |
| Angle of view | 62.2 °(H) x 48.8 °(V) |
| Interface | 15pin MIPI CSI-2 |
| Dimensions | 23.86 mm × 25 mm × 9 mm |
| Weight | 3 g |

## 5.3 Stereo Camera Setup

A stereo camera setup can be used to measure depths in images; this is accomplished by point triangulation (Heijden, 2016). The relative location and rotation of the cameras (also called the extrinsic parameters) determine the depth accuracy and full field of view of the stereo camera system. In Figure 7 a representation of a stereo camera setup is shown. This figure shows that a stereo camera setup needs two cameras placed next to each other with a certain "baseline" distance and rotation between them. The working distance is the distance from a camera to an object. This distance is important as it also determines the angle of the cameras.

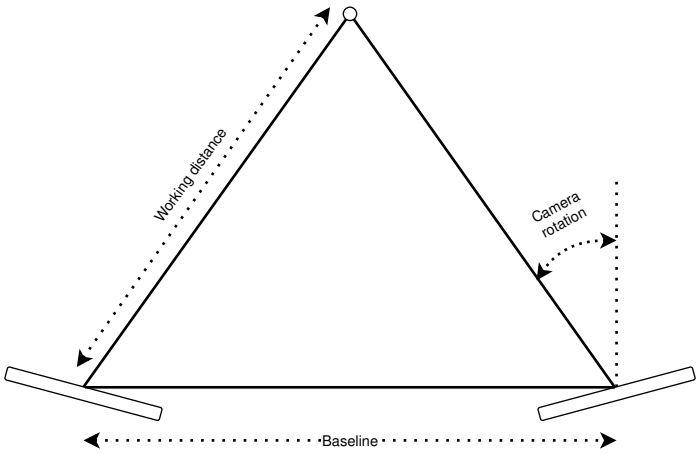



Figure 7: Representation of a stereo camera setup

For the stereo camera setup there is one requirement stated; the system must be able to measure depths with a resolution of 0.1 mm or higher (Section 3.1). Kytö, Nuutinen, and Oittinen (2011) describes the way the depth resolution can be calculated. This can be done by using Equation 5, where $dZ_c$ is the theoretical depth resolution, $f$ is the focal length, $b$ is the baseline,and $dp_x$ is the disparity accuracy. The focal length (3.04 mm) and disparity accuracy (1.12 $\mu$m) are known from the specifications of the Raspberry pi camera module v2 (Table 13).

$$dZ_c = \frac{Z^2}{fb} dp_x \tag{5}$$

To find a combination between the baseline distance and working distance that satisfies the of a depth resolution of at least 0.1 mm there has to be experimented with the two parameters. Using

Equation 5 with varying baseline and working distances gives two plots (Figure 8) showing the depth resolution for multiple configuration.

By choosing a suitable configuration of baseline and working distance, two things should be considered; one being the requirement of having a depth resolution of at least 0.1 mm, the second being the resulting angle between the two cameras with a specific combination between baseline and working distance. For example, a large baseline and a small working distance will result in a high depth resolution, but the camera rotations will be significant, resulting in almost no overlap in the captured images, therefore being useless. Thus, choosing a combination between baseline and working distance should not be solely based on the greatest depth resolution but also on minimizing the baseline distance.

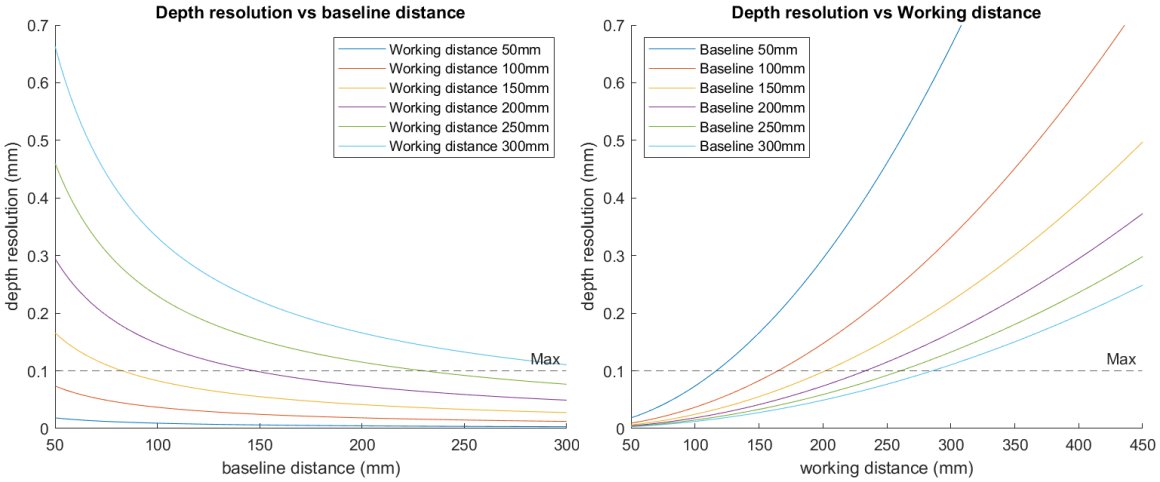



Figure 8: Depth resolution for different baseline and working distances

With all the criteria discussed above, the configuration of baseline and working distance are chosen to be a 100 mm baseline and a 150 mm working distance. This configuration will theoretically result in a depth resolution of:

$$dZ_c = \frac{150^2}{3.04 \cdot 100} \cdot 0.00112 = 0.0829\,\text{mm} \tag{6}$$

and the camera rotation in degrees relative to the other camera is:

$$\phi = 2 \cdot \sin^{-1}\left(\frac{50}{150}\right) = 38.94\,^\circ \tag{7}$$

A depth resolution of 0.0829 mm and a relative rotation of 38.94 ° complies with the set requirement while keeping the relative camera rotations as low as possible.

## 5.4 Inertial Measurement Unit

The DSE shows that this project's best option is to use an industrial sensor for inertial measurements. The company Xsens has a variety of motion tracking sensors. In Table 14 a summation of a subset of their product line is shown. The MTi-x series are small compact modules that have reasonable stability and precision. The MTi-x series are bare-bone modules that can be implemented onto a PCB. This version is therefore not resistant to vibrations, water, dust, and other environmental influences. The MTi-6x0 series comes within a case and is built for easy implementation because of its build-on connector. As with the MT-x series, the MTi-6x0 series is not protected against the environment. The MTi-x0 and MTi-x00 series come in a metal casing with IP67 rated protection. The difference between the two series is some stability improvement but, most of all, the better protection from external forces like vibrations and magnetic interference. **Note**, the MT x-series is

officially called the MTi-1 series containing the MTi-1, MTi-2, MTi-3, and MTi-7. The x-notation is made to eliminate confusion, but the MTi-1 series notation is used in official documentation and schematics.

Table 14: Inertial Sensor Comparison

| Version | MTi x-series | MTi 6x0-series | MTi x0-series | MTi x00-series |
|---|---|---|---|---|
| Gyro Bias Stability | 10 °/h | 8°/$h$ | 18°/$h$ | 10°/$h$ |
| Roll Pitch | 0.5° | 0.2° | 0.2° | 0.2° |
| Yaw | 2° | 1° | 1° | 1° |
| Dimensions (W/L/H) | 12.1x12.1x2.6 mm | 31.5x28.0x13.0 mm | 57.0x42.0x23.5 mm | 57.0x42.0x23.5 mm |
| Weight | 0.6 g | 8.9 g | 52 g | 52 g |
| Price IMU | € 129.00 | € 429.00 | € 699.00 | € 1399.00 |
| Price VHR | € 199.00 | € 569.00 | € 999.00 | € 2599.00 |
| Price AHRS | € 259.00 | € 649.00 | € 1199.00 | € 2799.00 |
| Price AHRS Dev Kit | € 399.00 | € 799.00 | € 1299.00 | € 2899.00 |

With the budget and mobility in mind, the best option for this project is the MTi-x series. It is the cheapest series and also the one with the smallest form factor. When chosen for the MTi-1 (IMU version), the system's pose must be calculated and estimated externally. Therefore, it is better to select the MTi-2 (VHR version) or the MTI-3 (AHRS version). For the faster implementation of the sensor in the prototype, choosing the available development kit version of the MTi-x series is beneficial. A development kit enables accessible communication with the sensor, faster experimentation, and better debugging features. The dev kit is only available for the MTi-3 version of the series. Therefore, the best choice for this project is the MTi-3-DK (dev kit).

**Sensor structure** In the diagram shown in Figure 9 the internal structure of the MTi-3 is displayed. The MTi-3 module has three 3D sensors; gyroscope, accelerometer, and magnetometer. The sensor also measures temperature to compensate for the temperature bias of the sensors. The MTi-3 has two synchronization options: a GPIO pin on the sensor and a request data package over the communication line. The sensor supports $i^2c$, SPI, and UART communication. The UART communication will be used in this project, as it is convenient with the Jetson Nano embedded controller.

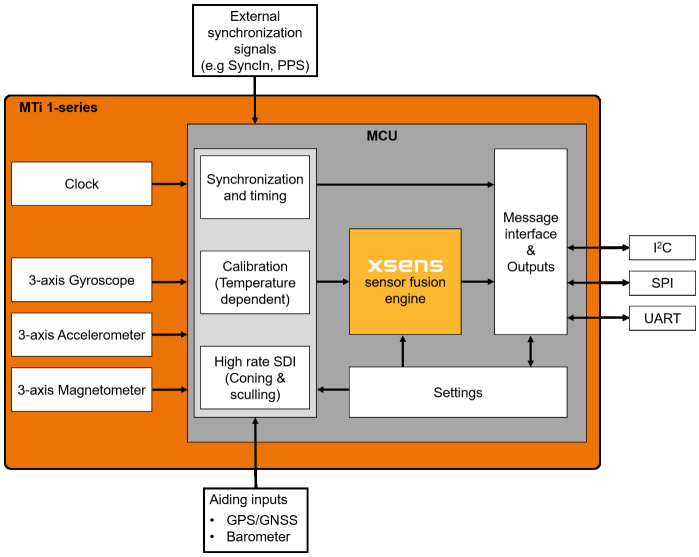



Figure 9: Xsens MTi 1-series module diagram (Xsens, 2019)

**Sensor parameters** The IMU sensor is set to use a baud-rate of 115 200 Bd. In addition, the sensor sends its current orientation in the form of 32-bit floating-point quaternions with an interval of 100 Hz. The messages from the Xsens sensor can be extended with more information about the sensor, like status messages, temperatures, and other information. It is also possible to extract the acceleration data or change in acceleration from the last data package.

## 5.5 Illumination Unit

For the illumination of the skin and the projection of light patterns (for reducing the adverse effects of specular reflection), the same illumination unit as the first generation prototype will be used. There may be smaller and cheaper alternatives to the one used, but this specific project does not focus on optimizing and improving the illumination capabilities but only on implementing one. Therefore, the **Philips PicoPix ppx5110** will be used. The specifications of this light projector are given in Table 15.

Table 15: Philips PicoPix PPX5110 specifications

| Property | Value |
|---|---|
| Light Source | RGB LED |
| Resolution | 854 x 480 pixels |
| Projection Distance | 53cm - 319cm |
| Brightness | 100 lumen |
| Interface | Mini-HDMI |
| Dimensions | 98x98x27.5 mm |

## 5.6 3D Printed Prototype

**Handheld scanner** For the hardware components discussed above, a 3D prototype setup is designed to house all the equipment for the handheld scanner. On the left side in Figure 10 the design for this prototype is shown. The prototype consists of a uni-body base with modular mounts for the different components. For example, the camera module is mounted in a camera module holder, designed for a specific baseline and working distance. The camera module holders can be interchanged to serve a different configuration. To give an impression of the time it takes to create a new camera configuration. A new camera mount can be redesigned and printed within the hour. There is also a controller board mount that allows for introducing different kinds of controllers and IMU's. In the middle of the base frame, there is a mounting pole for the projector. The location of this mounting pole aligns the projector lens with the center of the base. Finally, the prototype can be raised and lowered by the four threaded feet, giving more control in adjusting the working distance to an object.

**Rotary test setup** For validation of the system and testing future changes/optimizations, a test setup is needed to generate test data. The current method of creating a reliable data set is by placing a globe before a stationary positioned stereo camera setup and rotating the globe before the camera. This method will work with this prototype system if the IMU data is not used in the SLAM algorithm. The prototype has to move to create valuable data, and the (to-be-scanned) object has to be stationary. A test platform has been designed and build to which the prototype handheld device can be mounted. With the handheld device connected to the rotary test platform, the handheld device can rotate around a central platform with a fixed radius. This enables the scanner to scan an object placed in the middle of the test setup while simultaneously gathering inertial data. On the right side of Figure 10 the 3D design of the rotary test setup with the handheld device connected to it is shown.

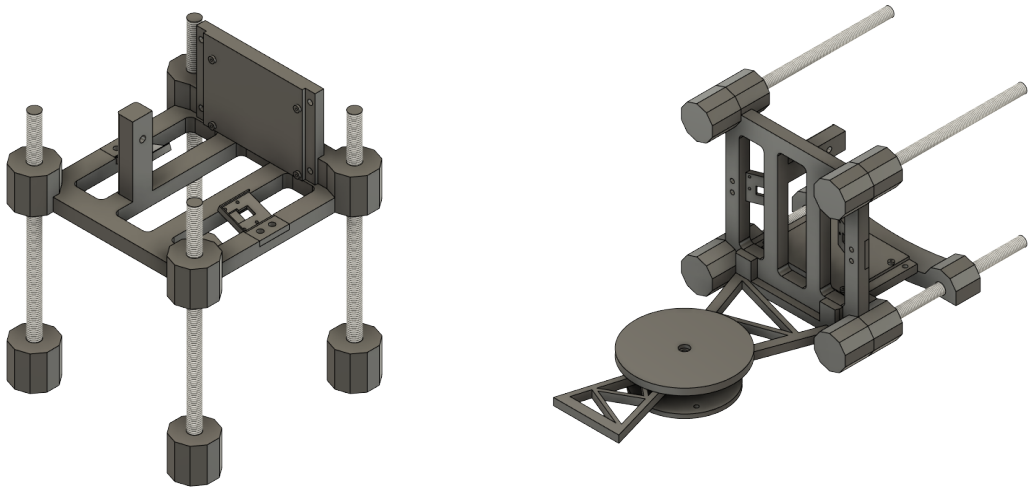


Figure 10: 3D designs of the prototype handheld scanner and the rotary test platform

## 5.7 SLAM Server

In Section 3.6 the need for embedding the GPU optimization designed by Wolters (2021) is described. One part of his optimization project is the design and implementation of an ES-EKF-SLAM algorithm that makes use of a GPU utilizing its CUDA cores. Currently, there is no dedicated workstation for the SLAM server. Therefore, a workstation is build consisting of the components given in Table 16. This setup is compatible with newer GPUs with a PCIe-4.0 interface. The power supply is rated for 650 W of power, the current system is calculated to draw 379 W, but the recommended PSU wattage is 429 W (Appendix C). The chosen 650 W PSU is also adequate for the RTX 3000 series GPUs. The Nvidia RTX 2080 Ti GPU used in this system is from the BMPI - HAPI project group and can be used for this project.

Table 16: Component list of the SLAM server work-station

| **Sub-system** | **Component** |
|---:|---|
| Processor | AMD Ryzen 5 5600X |
| Graphics Card | Gigabyte GeForce RTX 2080 Ti Gaming OC |
| Random Access Memory | Corsair Vengeance LPX 2x8GB 3200MHz |
| Motherboard | Gigabyte B550M DS3H |
| Solid Sate Drive | Kingston A2000 500 GB |
| Power Supply Unit | Corsair TX-M Series TX650M V2 |
| Case | be quiet! Pure Base 500 Black - Midtower |


Figure 11: Photo of the SLAM server work-PC

## 5.8 System Architecture

Now that all component designs are discussed separately, it is time to discuss how the components relate. The system architecture describes the topography of the system; a graphical representation is shown in Figure 12.

As discussed earlier, the system consists of two main parts; the handheld scanner and the separate SLAM server. The handheld scanner equips two cameras (Raspberry pi camera modules v2) to capture stereo camera images connected using the CSI communication ports on the Jetson Nano. The Xsens MTi-3 dev kit is controlled using UART and provides the system with rotational information. A projector is connected using the HDMI connection to illuminate the surface and for projecting patterns onto it. Lastly, a USB Wi-Fi adapter is used for wireless communication between the handheld scanner and the SLAM server. An M.2 compatible Wi-Fi module can replace this USB Wi-Fi adapter in the future, but it works fine for performing experiments in this stage of the project.

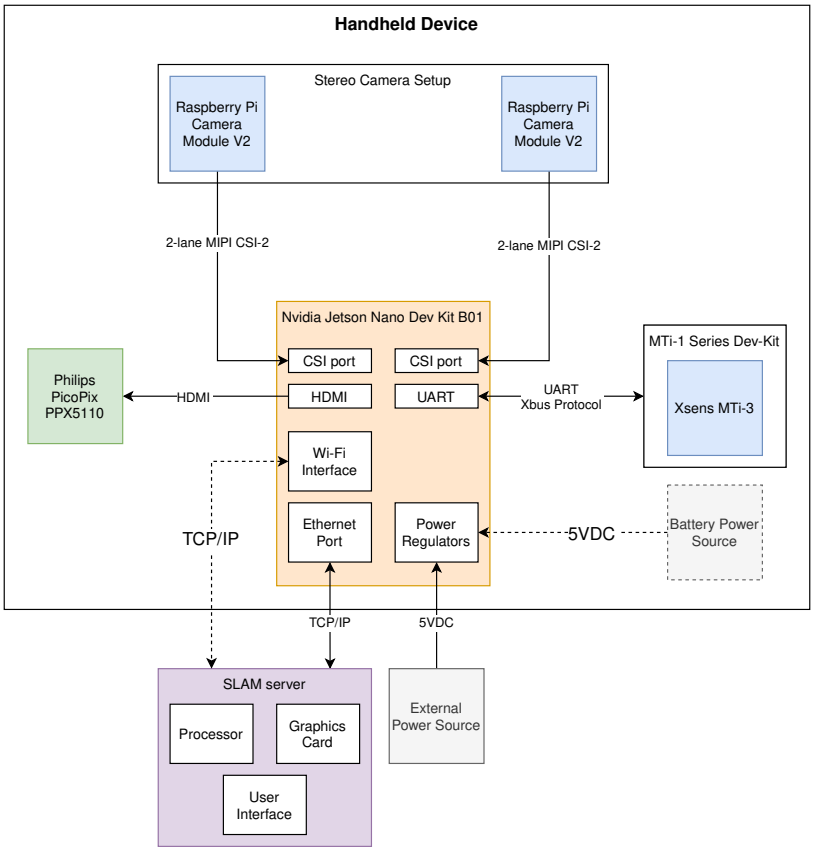



Figure 12: System architecture

## 5.9 Stereo Image Acquisition Software

Transitioning from the hardware to the software, we start at the image acquisition software. For the image acquisition software, there has been made use of the *Nvidia Libargus Camera API* in combination with a *GStreamer pipeline*. In the paragraphs below, the components of the stereo image acquisition software are explained.

**Nvidia Libargus Camera API** Libargus is an API for acquiring images and associated metadata from cameras (Nvidia, 2021). The Libargus is specially designed for the Nvidia Jetson devices but is also supported on Android devices. In Figure 13 the camera architecture stack of the Jetson series is shown. In this diagram, the path from the camera sensor to the GStreamer application is indicated with red arrows. As one can see, the *libargus* API and the *nvarguscamerasrc* are used to create a pipeline to the GStreamer application.

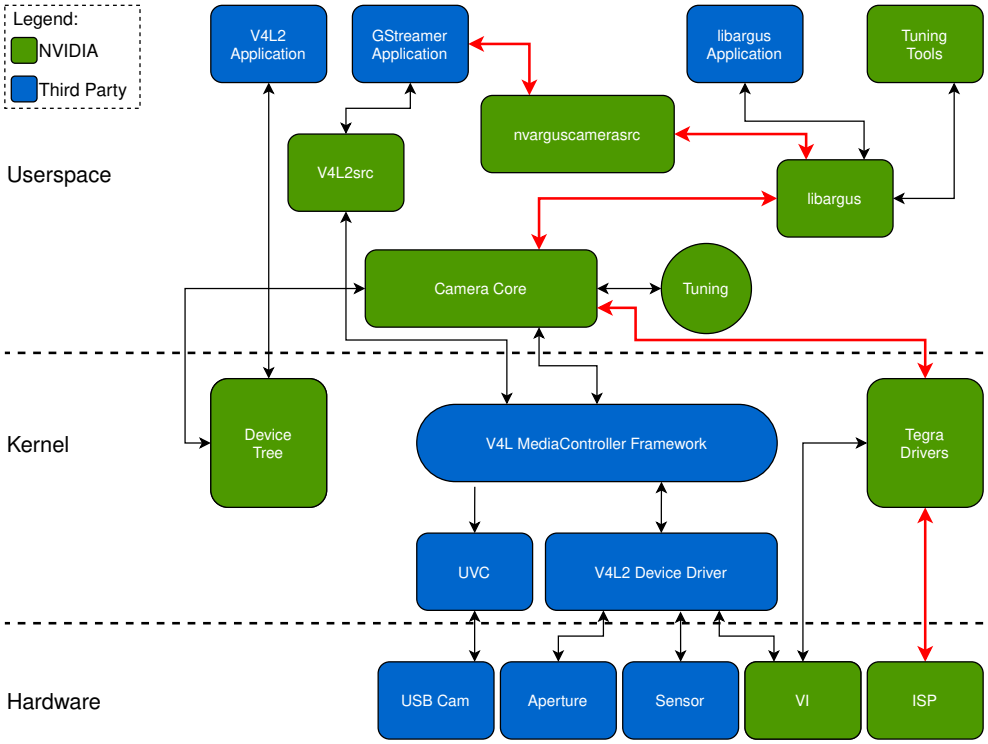



Figure 13: Jetson Camera Architecture Stack (Nvidia, 2018)

**GStreamer Pipeline** As discussed in the previous subsection, the Libargus camera API is used to create a camera capture pipeline. The Libargus API is implemented employing a GStreamer pipeline. This pipeline uses the *nvarguscamerasrc* GStreamer source module to get the images from the camera stack. It then performs pre-processing on the gathered images by using *nvvidconv*, the pre-processing consists of flipping/rotating and cropping the image (the capture and pre-processing are both performed on the GPU). Next, the data will be sent to the CPU where the *videoconvert* module converts the format of the images to BGR format with a color depth of 8-bits. Lastly, the resulting image will be stored in the *appsink* buffer, ready to be used by an application like OpenCV. The described pipeline is visualized in Figure 14.

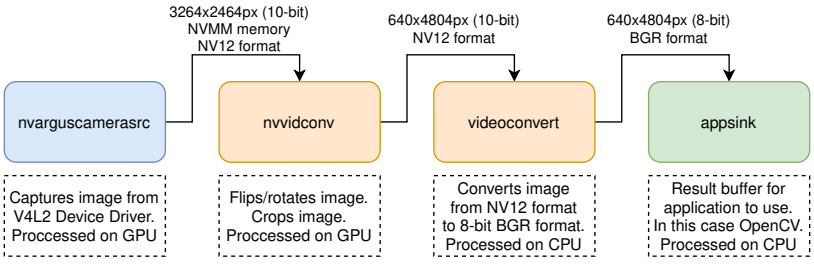



Figure 14: GStreamer pipeline from Libargus to OpenCV

**OpenCV** After the GStreamer pipeline, an OpenCV VideoCapture instance will capture the images from the GStreamer appsink and further process them. It is essential that the GStreamer appsink buffer is read continuously by the OpenCV VideoCapture instance, as failing to read the camera pipeline buffers fast enough will result in a buffer overflow in a matter of time. Therefore, in the embedded software, two threads are running, continuously retrieving new data from the image pipeline and storing the resulting images in a circular buffer for the rest of the program to use.

## 5.10 Xsens IMU Controller Software

For the control of the Xsens IMU, the Xsens Device API (XDA) is used. As illustrated in Figure 15, the XDA provides an interface between host applications and serial communication drives. As described in Section 5.8, a UART communication will be used to communicate with the Xsens IMU. The XDA aids in the communication, parsing of messages, and configuration of the Xsens sensor. For debugging purposes, the Xsens MT manager software suite is a great tool to inspect messages and experiment with settings.

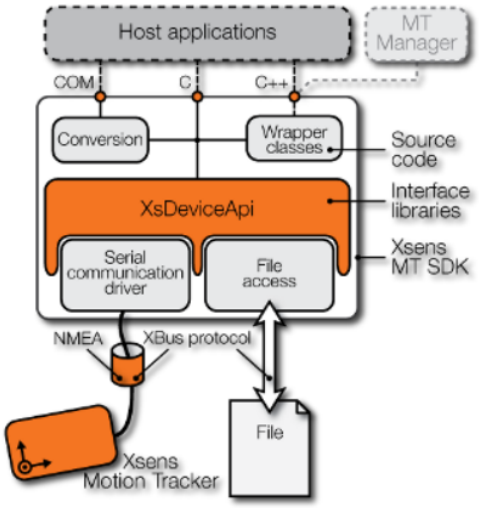



Figure 15: Xsens Device API (Vydhyanathan and Bellusci, 2018)

The Xsens Device API is not natively available for the aarch64 (64-bit ARM) architecture that the Jetson uses. To make this library work on the Jetson, the open-source, less feature-rich public XDA library is built from source and used. The public XDA provides enough functionality for essential communication with the sensor and does currently apply no limitations on the project.

## 5.11 Data Synchronization

The data from the stereo camera setup and the Xsens IMU must be synchronized for the data to be valuable. A multitude of mechanisms is used to accomplish this goal. Unfortunately, the used cameras do not have any form of hardware synchronization, meaning the clocks and the shutters of both modules are not synchronized. Because of the lack of hardware synchronization, the synchronization issue has to be solved softwarematically.

**Cameras** Synchronizing the cameras begins with making sure the cameras start simultaneously so that the camera's clocks run as synchronous as (softwarematically) possible. This is accomplished by opening each camera pipeline on a separate thread running parallel to each other. After that, the timestamps of the captured two frames are compared, and a frame offset is calculated to make sure that the frames of the two cameras have as small of a sync delay as possible. Next, the camera images need to be read simultaneously and continuously. As explained in Chapter 5.9, a GStreamer pipeline is used to communicate with the camera modules. The GStreamer pipeline and the Libargus API make use of FIFO buffers to store the acquired images. Finally, a thread runs for each camera in the camera controller class, continuously extracting frames from the GStreamer pipeline and storing the results in a circular buffer with the available metadata of the captured images.

**Inertial sensor** For the synchronization of the inertial data from the Xsens sensor, two methods are tried. The first method works by sending a data request package to the sensor and waiting on the result. The Xsens sensor does not send any data until it receives such a data request packet. With this method, the measurements are softwarematically synchronized with the image data. This system worked on a Windows development PC, but on the Linux distribution running on the Jetson Nano, this method resulted in deadlocks. This may be the result of how the UART drivers work on the Jetson. Because of this issue, the synchronization is now done by letting the inertial sensor output its data at its highest frequency (100 Hz). All the packages from the inertial sensor are being captured by the Jetson and stored in a circular buffer. When the synchronizer class wants to construct a data package, it requests the image buffers' image buffers and the inertial data from the inertial data buffer and merges this into an acquired data packet.

**Projector** The design of the prototype does also has access to a projector connected to the embedded controller using an HDMI connection. The synchronizer class can control the projector by showing figures on the HDMI port. However, the functionality of the projector controller is limited and made to be expanded in the future.

## 5.12 Camera Calibration

For the undistortion of the acquired images and the 3D point triangulation, all intrinsic and extrinsic camera parameters have to be known. A camera calibration program can estimate these camera parameters. In this project, Matlab's stereo camera application is used to estimate the parameters.

**Calibration using a chessboard pattern** One popular method for calibrating a camera is by using a chessboard pattern and localize the crosses of the chessboard pattern in the image (Heijden, 2019). The same method is used for stereo camera calibration, but then, instead of the intrinsic parameter, the extrinsic parameters are calculated. In this project, the Matlab camera calibration app and the stereo camera calibration app will be used. First, the camera calibration app is used to calculate the camera's intrinsic parameters separately. Next, the stereo camera calibration app is used to combine the two and calculate the extrinsic parameters. Then, a Matlab script is made to convert Matlabs' stereo parameter object to a YAML file which the embedded software can read on the handheld device.

**Image cropping** The field of view is relatively small in this prototype. This is caused by the short working distance in combination with the set focus of the cameras for that specific working distance. Therefore, for good calibration images, the size of the entire image is used. Consequently, the result of the camera calibration is only valid for that specific setting (focus and image size) of the cameras and cannot be directly used on cropped images. Equation 8 shows the structure of an intrinsic camera matrix, here $f$ is the focal distance, $c$ is the principle point offset, and $s$ is the axis skew.

$$\begin{bmatrix} f_x & 0 & c_x \\ s & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The principal point offset indicates the number of pixels from pixel (0,0) to the optical center. When cropping an image, this offset changes. Equation 9 calculates the new principal point offset based on the current values and the set cropped image size. The intrinsic camera matrix is automatically adjusted to the user's desired cropped image size within the embedded software when the "full size" camera calibration is inserted.

$$\begin{aligned} c_{x_{new}} &= c_x - 0.5 \cdot (img_{width} - cropped_{width}) \\ c_{y_{new}} &= c_y - 0.5 \cdot (img_{height} - cropped_{height}) \end{aligned} \tag{9}$$

With the new principle point known the next operation is to calculate the the parameters for the cropping operation. Four parameters have to be known to perform a cropping operation; horizontal and vertical starting position, and horizontal and vertical crop size. Within the embedded software the crop size is given by the user at the start of the program. The crop operation is always performed from the center of the image, the starting $x$ and $y$ positions are calculated by the following equation (Equation 10):

$$\begin{aligned} pos_x &= \frac{img_{hor} - crop_{hor}}{2} \\ pos_y &= \frac{img_{ver} - crop_{ver}}{2} \end{aligned} \tag{10}$$

**Matlab scripts** Two Matlab scrips are made for the processing of the stereo camera parameters acquired from the Matlab stereo camera calibration app. One file contains a function that takes the original stereo camera parameter object as an input plus the set image crop size and will result in a new and adjusted stereo camera parameter object. The second Matlab script also takes a stereo camera parameter object and converts this to a YAML file which can be used as input for the embedded software.

## 5.13 Data Flow

The software forms one large pipeline from the sensors on the handheld scanner to the external SLAM server. In the figure below (Figure 16), the data flow of the system is graphically displayed. The difference between computation on the CPU or GPU is differentiated by orange and green color. The dashed boxes indicate the two physical systems; the handheld scanner and the SLAM server.

It all starts at the cameras; these are connected via a CSI connection directly to the GPU. On the GPU of the Jetson Nano, there are two ISP's (image sensor processors) controlling and retrieving the image data from the camera sensors; these ISPs also have some image processing/computer vision capabilities. Within the designed pipeline, the ISP's perform rotating, cropping, and color depth downscaling operations on the gathered full size (3264 px × 2464 px) images. Next, on the CPU, two image capture threads are running to continuously read the output buffers of the ISP's on the GPU and storing those images in circular buffers.

A synchronizer class can access the image frame buffers when a data request is received. The synchronizer class can also control the projector and read the latest rotational data from the inertial data circular buffer. The Xsens inertial sensor fills the inertial data buffer with a sample rate of 100 Hz. The stereo images are ported into the image undistortion algorithm from the synchronizer, which can be executed on the CPU or GPU. The undistortion of the input images is in this configuration with the camera modules not necessary because of the low radiant distortion of the lenses and the cropping of the images, and therefore can be skipped. Next, the images are put into the landmark acquisition algorithm executed on the GPU, which results in landmarks that are sent with the orientation data from the Xsens sensor to the external SLAM server utilizing a TCP/IP connection.

On the SLAM server, the send data will be received and processed through the GPU-optimized EKF-SLAM algorithm. The results of the SLAM algorithm are stored locally and can be visualized using a Matlab script afterward.

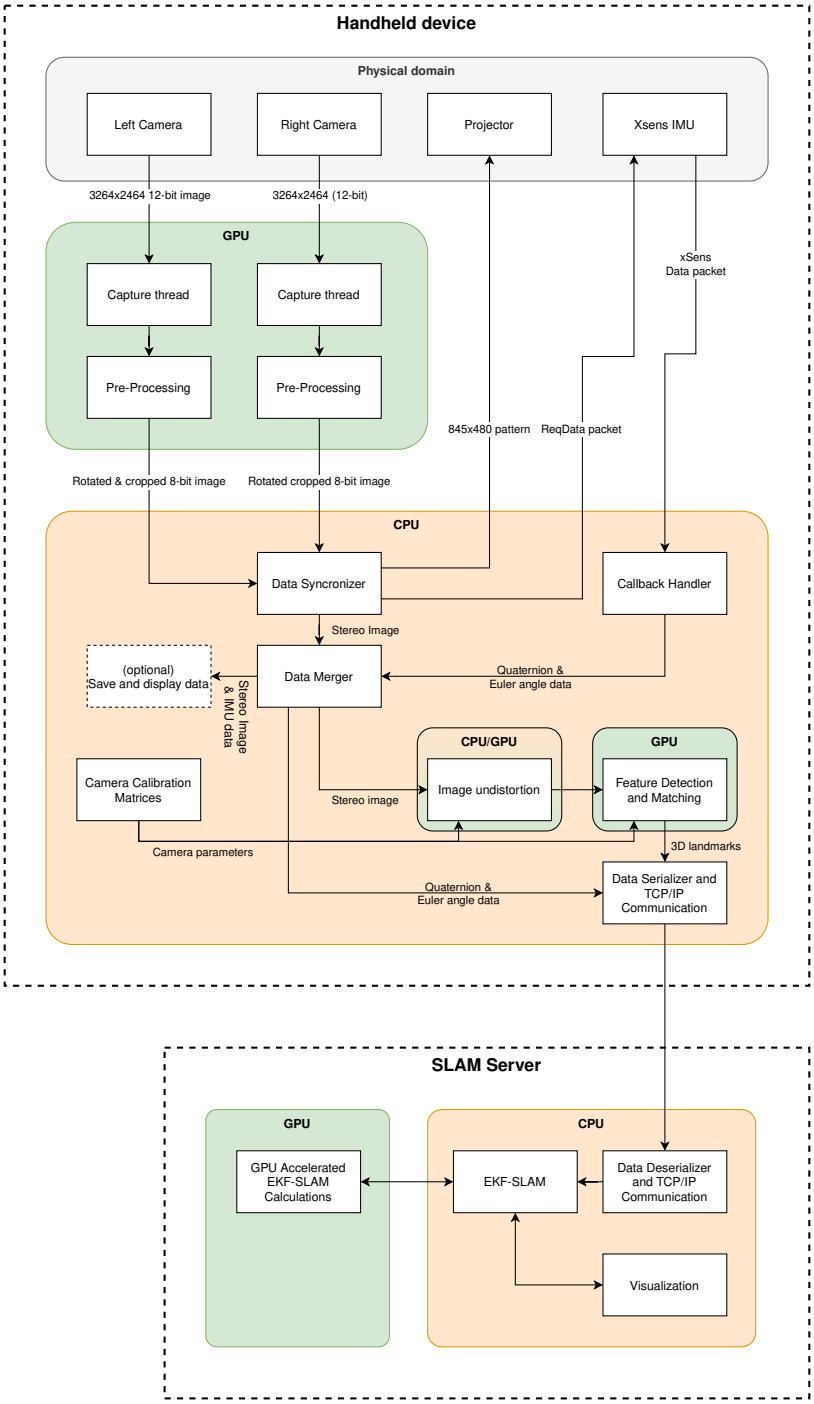



Figure 16: Dataflow system pipeline

## 5.14 Class Diagram

The class diagram shown below in Figure 17 displays the structure and relations of the classes used in the embedded software of the handheld device. The class diagram shows that the system consists of two "main" classes; the synchronizer and the main class. The synchronizer handles the inputs and outputs of the peripherals. The main class retrieves synchronized data packages from the synchronizer class, runs the pre-processing functions and sends the data to the SLAM server.

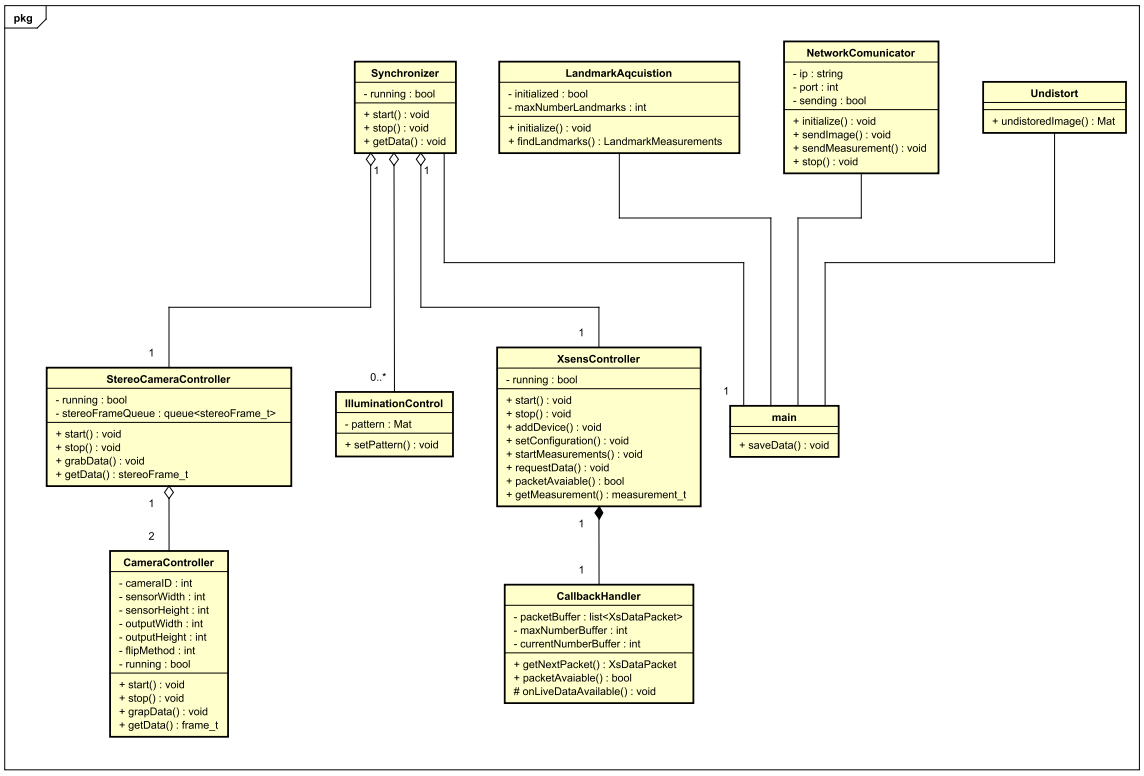



Figure 17: Class diagram embedded systems

## 5.15 Development Software

Different development methods and software are used to ensure code quality and easy installation with the embedded software development. In the sections below, the various development software is briefly discussed.

**CMake meta build system** One of the goals of this project is to create a prototype system that can be used as a test bench. Therefore, it is essential that the software can be easily used. Setting up a C or C++ development workspace/toolchain can be complicated with all the different compiler settings and libraries used. To make the setup and maintenance of the code more manageable, a program named CMake is implemented. CMake is a cross-platform meta-build system tool designed to build, test, and package software. CMake is implemented by inserting special *CMakeLists.txt* files into the code tree. These files describe how the different source files are related to each other. Because CMake is a meta-build system, it can be used cross-platform. For instance, when using CMake on a Linux distribution, CMake generates Makefiles. When Cmake is used on Windows, it generates a Visual Studio solution.

**Git version control** Most projects use a source version control system to enable better collaboration between team members and keep code reliability high. In this project, the Git service of a

self-hosted Gitlab instance from the RAM group is used. In this way, the development of the code is tracked, and any changes in the code are logged.

**Visual studio code remote development** Development directly on the Jetson Nano by connecting a screen and keyboard is not recommended because of the device's speed. A better way to develop on the Jetson Nano is by using remote development. The Visual Studio Code IDE has an excellent extension allowing remote development over SSH with the ability of remote debugging as well. The remote development is made easily accessible by the implementation of CMake.

**GitLab wiki** Within the repository of this project, a wiki is created to guide new users with the installation and usage of the prototype system. The wiki contains links and instructions for installing libraries and instructions on how to operate the system.

# 6 Experiments & Results

This chapter discusses the experiments and results that are performed to evaluate the designed system. Each of the sections below describes an experiment or component of the project.

## 6.1 Software Performance

Starting with the first experiment, testing the performance of the embedded software on the handheld scanner. The handheld scanner runs mainly on three CPU threads; one thread per camera module (thus, two threads in total), handling the data acquisition, and another thread for processing the acquired data. In Figure 18 the results of profiling both programs are shown. On the left side, the total duration of a single iteration of the image acquisition is given. The average duration is taken from 1000 samples. In the right plot, the computation times per component of the data processing are shown. Because of the relatively higher throughput times of the data processing, the average is taken over 100 samples instead of 1000. This reduces the time it takes to complete the experiment while still having a large sample pool. In both plots, the x-axis represents the results for multiple camera resolutions, while on the y-axis, the duration is shown. At the top of every bar, the red indicator mark indicates the standard deviation. In the two paragraphs below the figures, the two results are explained per sub-program.

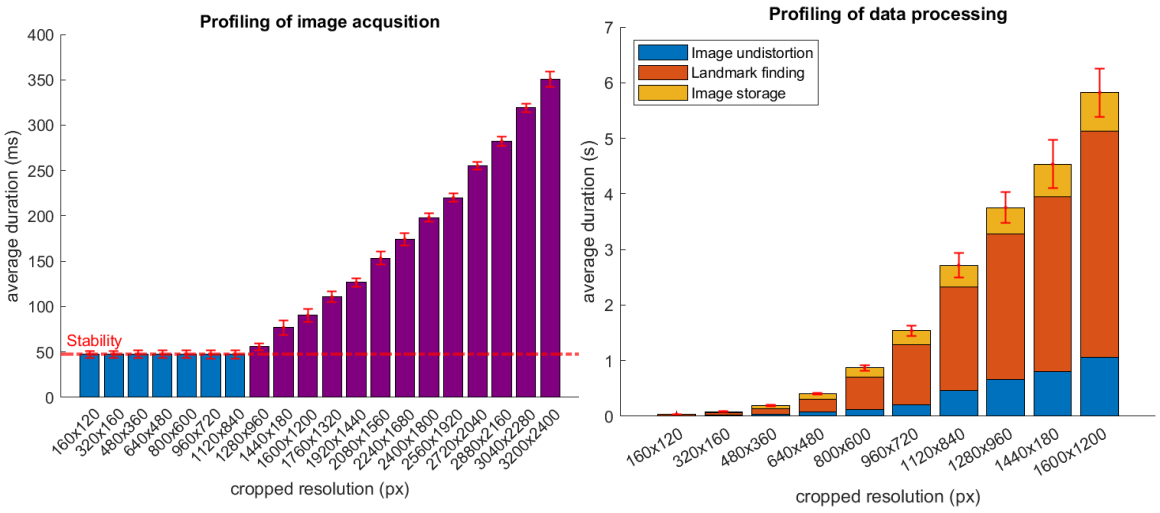



Figure 18: Profiling results of the handheld device embedded software

**Image acquisition** The image acquisition software is responsible for *grabbing* the image data from the image buffer on the GPU, converting images to usable data, and storing the data in a circular buffer. The results of profiling this software are shown in the left plot in Figure 18. The image grab function used to load in an image from the GPU is a blocking function; this means that the function will wait until an image is available. Therefore, it is interesting to see that for image resolution up to and including 1120 px × 840 px the throughput time is the same. These throughput times are not arbitrarily but are the same as the camera frame rate: 1000/21=47.63 ms. Therefore, the actual period of the function will be lower than indicated in the plot. The red dashed line around 48 ms shows the threshold for a stable system (given that the cameras run at 21 FPS). When the average throughput times are higher than the stability threshold, the FIFO will fill up en eventually result in a buffer overflow. Therefore it is not recommended to use a higher resolution than 1120 px × 840 px in combination with the Jetson Nano controller. Using a better version Jetson board may improve the throughput times, thereby allowing larger resolution images.

**Data Processing** Parallel to the data acquisition software, the acquired data is processed. The data processing flow exists of; loading in the received data, undistortion of the images, landmark

detection, data transmission, and optionally storage of the data. In the right plot of Figure 18, the computation times of three of the sub-systems are shown for different image resolutions. As one can see, the throughput times ramp up fast, with a throughput time of close to 3 s for an image set of 1120 px × 840 px. The function that takes the most time to complete is the landmark finder. There has been experimenting with using even larger image sizes, but this takes too much time. Also, processing such amounts of data does heat up the Jetson Nano to the point that active cooling is needed, or the system will go into thermal throttling (down-clocking the clock speeds).

## 6.2 Prototype Setup

In Section 5.6 the design of the 3D printed prototype is discussed. The images below (Figure 19) shown the final result of the 3D printed prototype system with all the components mounted to it. In the left image, the handheld device is in its stationary position. With the help of the adjustable feet, the distance from the camera to an object can be precisely configured. This position is perfect for scanning stationary objects. Note that moving an object underneath the device will not give inertial data as the scanner itself is motionless. This can cause issues when the SLAM algorithm uses the inertial data from the handheld device. The right image shows the handheld scanner mounted into the rotary test setup. A stationary object can be placed in the middle with this setup, while the scanner can rotate around it with a fixed radius. With this method image, data can be acquired while simultaneously measuring the orientation of the scanner.

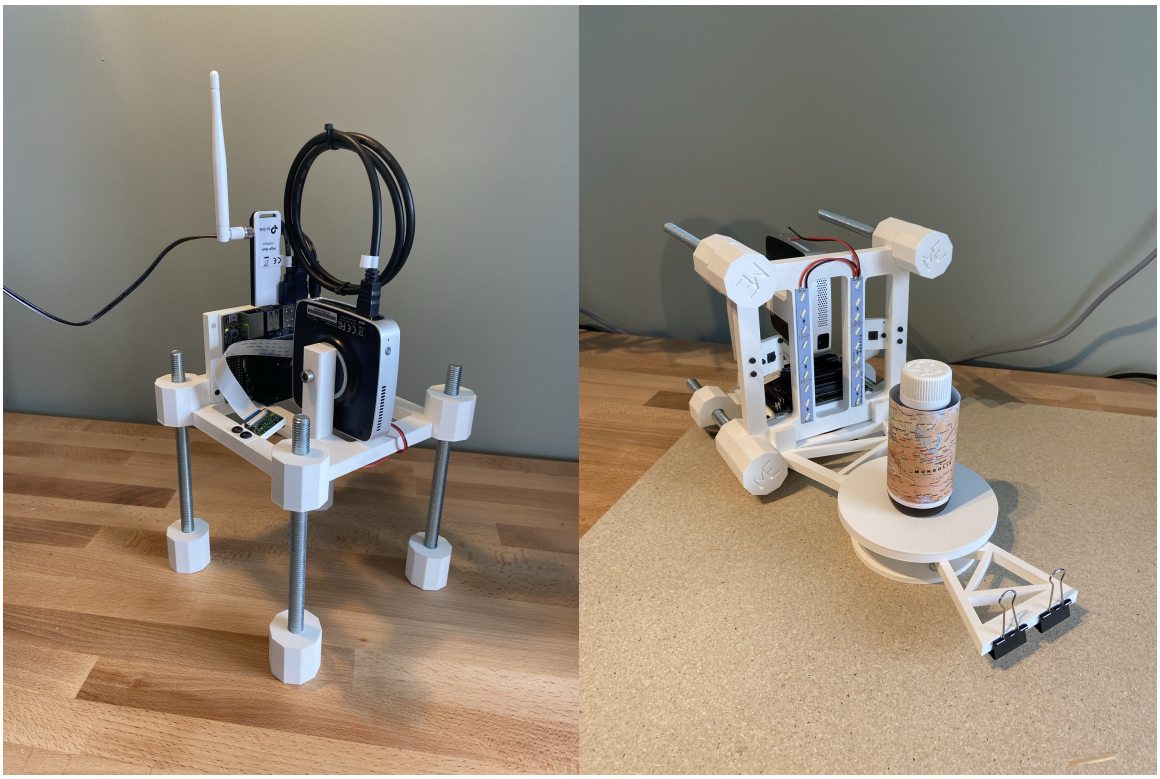


Figure 19: Photos of the prototype handheld device and the rotary test setup

## 6.3 Cameras

**Stereo camera lag** As discussed in previous sections, the two cameras used in the stereo camera setup are not synchronized by hardware. This is not by choice, but the camera modules do not have an out-of-the-box solution for hardware synchronization. Synchronization of the images is needed to minimize the disparity error between the images. Therefore, synchronization is accomplished soft-

waremathically in this prototype. The way the synchronization software works results in a delay/lag between cameras that can theoretically go up to half the frame rate of the cameras. Thus, in this case, the maximum theoretical camera lag is 23.81 ms. When evaluating the sync delay in reality, it shows that the delay can get higher than the theoretical limit. This can be caused by the cameras not outputting frames at exactly 21 frames per second. In Figure 20 a box plot of the camera lag is shown with a variety of resolutions. A total of 1000 samples is used per resolution to give a realistic result. The results show that the camera resolutions up to and including 960 px × 720 px are rather low, for example at 640 px × 480 px the sync delay is around 15 ms, while resolutions above the 960 px × 720 px have an significant increase in lag. The camera lag is measured by calculating the difference in the timestamps of the matched frames. As discussed in the design section, the two cameras do not share the same clock but are started at roughly the same time. Therefore, there may be an unknown offset between the camera clocks, and thus the measured camera lag might differ some milliseconds.

Keeping the stereo camera lag low is essential, as an increased lag introduces more disparity error. The disparity error is the error in the measurements for the point triangulation. This is caused by using two images that are not taken simultaneously and therefore may be shifted (e.g., the object moved a little between the two image captures). Therefore, the best practice to prevent disparity errors is to keep the scanner still while capturing images.

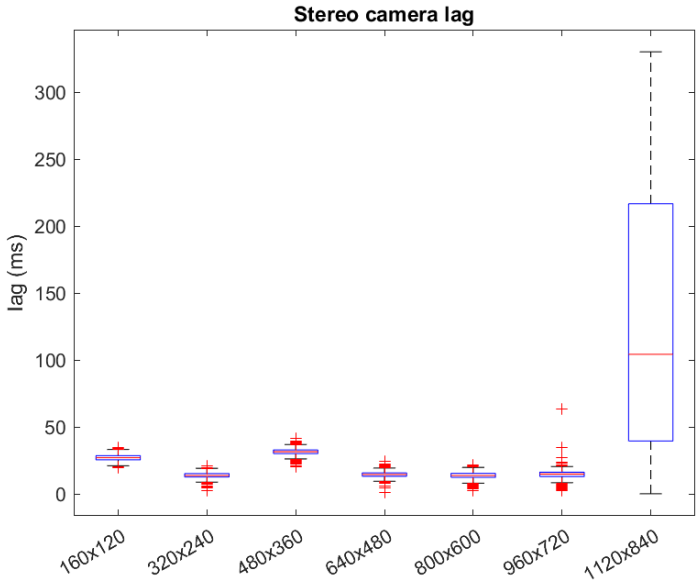



Figure 20: Box plot of stereo camera lag

## 6.4 SLAM

To evaluate the whole system's functionality, the rotary test setup is used to check the entire system pipeline, from the stereo cameras to the external SLAM sever. For the experiment, the system is set to use cropped images of 640 px × 480 px, the images are undistorted, keypoints are acquired, the resulting keypoint matches are sent wireless to the SLAM server, and finally processing the measurements through the SLAM algorithm. In the meantime, all gathered data, like images and inertial data, is saved locally on the Jetson. A cylindrical-shaped object with an image wrapped around it is used as a test object. This test object is used to have a constant round shape with a very distinguishable surface. Moreover, the test object gives a reliable ground truth because the used test object has a consistent diameter. In this experiment, an object with a diameter of 45 mm is used. Larger objects can also be used, but because of the size of the rotary test setup, the object size is limited.

The results are plotted in a 3D plot using a Matlab script. The front and side view of this 3D

plot is shown in Figure 21. The two green circles indicate the ground truth of the test object and the path of the stereo camera setup. The red dots are the mapped points of the test object, and the blue points indicate the estimated path of the camera setup. The right side of Figure 21 (side view of SLAM result) shows movement in the z-axis in the path of the camera. This movement should be almost non-existing as the rotary test setup makes sure of this. Therefore, only vibrations should be visible. With this experiment, the gathered inertial data of the handheld scanner is not embedded in the SLAM algorithm. The SLAM output is, therefore, purely based on the stereo camera images (vSLAM). In the next chapter, the results from this experiment will be discussed.

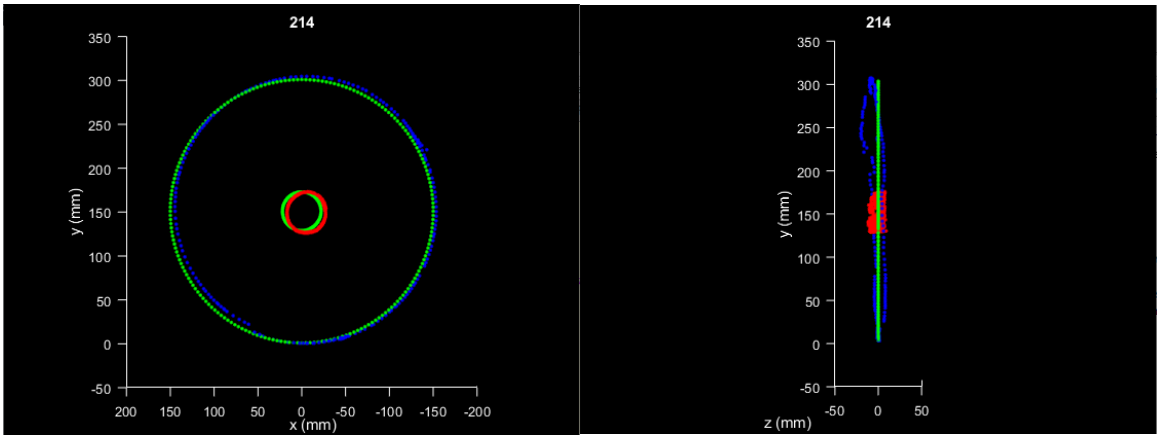



Figure 21: A font and sideways view of the SLAM results 3D plot

## 6.5 Landmark Acquisition

The landmark acquisition software finds and matches keypoints on the left and right images. In Figure 22 an example of one key point finding and matching result can be seen with the experiment described in the previous section. The system is configured to use the 200 strongest keypoint matches. When the resulting key point matches are analyzed frame by frame, it can be seen that the key points are mostly found on a vertical line around the horizontal center of the two images. This effect may come from the significant distortion at the sides of the images caused by using a small cylindrical object. The small object in combination with the configuration of baseline and working distance also gives that there is little overlap in the stereo images.

The experiments show that the landmark acquisition is prone to error and is not robust against arbitrarily chosen surfaces. Before the system can function the way shown in the experimental results, the keypoint rejection parameters of the landmark acquisition need to be tweaked. The adjustment of this rejection parameter is precise and is depended on the surface. Too much rejection will result in too few keypoint matches which makes which reduces the robustness and accuracy of the SLAM algorithm. On the other hand, too many matches, including mismatches, can break the algorithm. Another finding of experimenting with the prototype setup is that the throughput times depend on the type of surface/object is used. After some tests, this phenomenon can be linked to the landmark acquisition. When an input image with lots of different shapes, forms, or patterns is used, the landmark acquisition finds lots of keypoints in both images, which results in a longer computational time for the keypoint matching of all those keypoints.

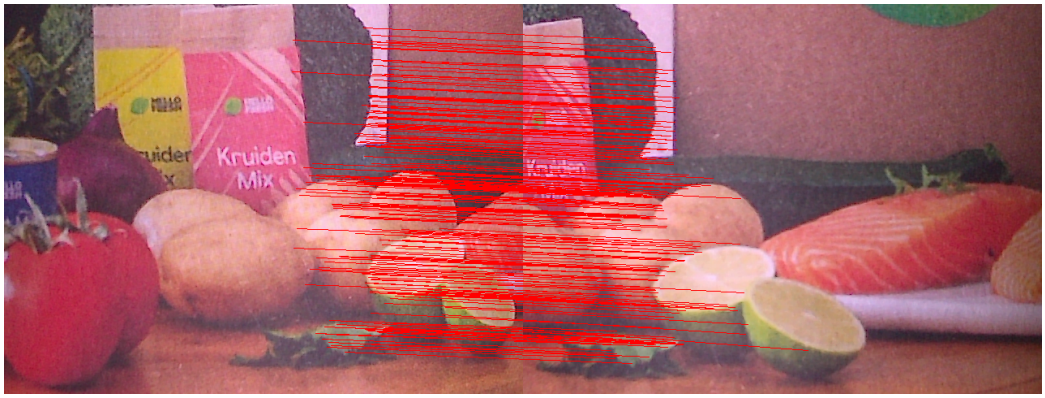



Figure 22: Example of keypoint finding result

## 6.6 Inertial Measurement Unit

The Xsens IMU outputs rotational data of the handheld scanner with a sample rate of 100 Hz. The rotational data is already the integrated angular rate and is also compensated and calibrated for temperature and magnetic interference and thus the current orientation of the handheld scanner. In Figure 23 the data from the Xsens sensor (blue) and the SLAM estimated orientations (orange) are plotted. In the left plot, the quaternions are shown, and the right plot shows the same data but then in Euler angles. The data is as expected. When looking at the Euler angles, the sensor makes a full 360-degree rotation, and the SLAM estimate follows this curve. The other two rotations, the roll and pitch, should be stable as the rotary test setup used in the experiment only allow yaw rotations. There are small rotations around the roll and pitch axes of around $\pm 0.4\,^{\circ}$ in the IMU data. These rotations are most likely vibrations caused by the plastic wheels of the rotary test setup and the clearance on some of the parts. However, the SLAM estimated orientations around the roll and pith show much more displacement than the IMU. When accepting the IMU as ground truth data, it can be seen that there is some error on the estimated data of the SLAM algorithm.

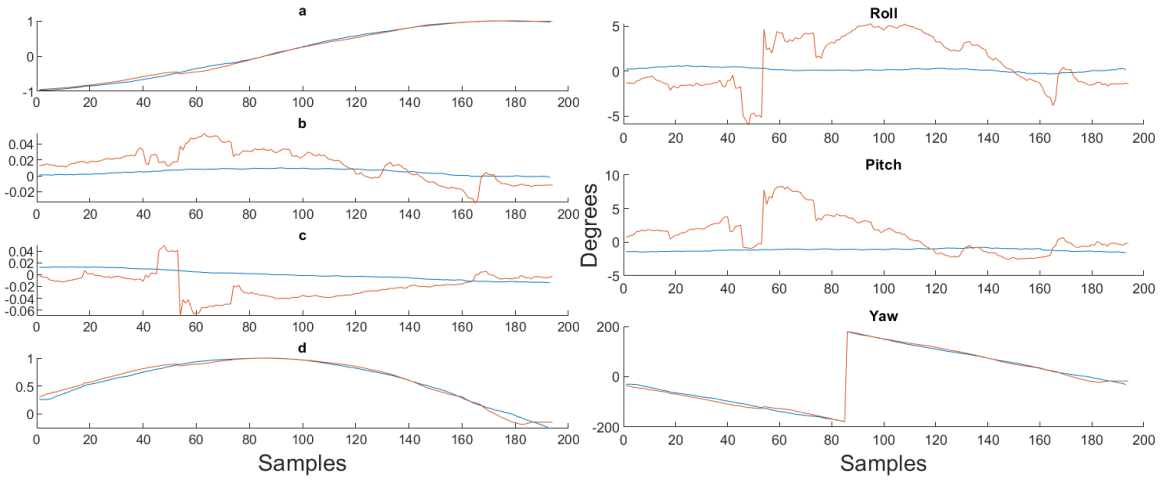



Figure 23: IMU data results at experiment (blue: IMU orientation - orange: SLAM estimated orientation)

# 7 Discussion

In this second to last chapter of the report, the results of the experiments and the evaluation of the design objectives are discussed.

## 7.1 Data Acquisition

The stereo camera setup uses two 8 Mpx cameras with an baseline of 100 mm and a optimal working distance of 150 mm. Within the embedded software, the images from the camera sensors can be cropped to reduce the amount of data. The current setup with the Jetson Nano as embedded controller allows for the processing of images with a maximum resolution of 980 px × 620 px. With this resolution, the image acquisition and the camera synchronization are stable. With a 980 px × 620 px cropped resolution the throughput time is around the 1 s, and the resulting field of view is 54 mm × 34 mm. The embedded software allows the use of different resolutions. Choosing a lower resolution improves the throughput time of the pre-processing, but this will reduce the field of view. With the current setup, the field of view cannot be expanded as increasing the image crop size results in an unstable system. Also, the current theoretical depth resolution is 0.0829 mm with the given camera parameters. This gives almost no space to down-scaling the resolution to enlarge the field of view, as this would lower the depth resolution.

## 7.2 Pre-processing

The pre-processing done on the handheld scanner includes the synchronization of the data, undistortion of the images, landmark acquisition, the transmission of the data, and optionally storing the results. With the experiments, it has been seen that the landmark acquisition software seems to find only matches around a vertical line located near the middle of both images. This phenomenon might be caused by the distortion imposed by the roundness of the test object. For example, when looked at the limes of both images in Figure 22, a distortion of the shape of the limes can be observed in the left image. This can explain why only keypoint matches are found in one specific area. This theory was tested by using a larger object without any curvature. This experiment showed that keypoints were found all over the overlapping areas of both images.

Also, when experimented with different types of surface patterns, the execution time of the landmark acquisition seems to vary. After some experiments and monitoring the results, it is clear that these fluctuations result from the number of keypoints detected within the images. A larger pool of keypoints results in a more extensive search for keypoint matches.

## 7.3 SLAM results

The SLAM results (Figure 21) show that the prototype system works, as the path/location of the camera and the points on the object look correct. However, there is some noticeable error in the z-axis of the estimated camera path. This error has most likely caused by the landmark acquisition. The experiments observed that there were some mismatches in the keypoint matching of the landmark acquisition algorithm. Mismatches may result in strange behavior when processed through the SLAM algorithm. Another explanation for the movement of the camera path in the z-axis is that the handheld scanner is indeed just moving/vibrating within the rotary test setup. When taking a look at the output of the SLAM estimated orientation in comparison to the IMU data (Figure 23), it can be seen that the SLAM algorithm estimated orientations around the roll and pitch axes have some error. Therefore, it seems most likely that the error is the result of the landmark acquisition.

Besides the slight error in the z-direction, the prototype system shows promising results as the object's surface is indeed reconstructed. The experiments show that the pipeline from data acquisition to the SLAM server works but that the system needs to be further optimized to increase throughput times and thereby increase the overall usage. The orientation data from the Xsens sensor is not yet implemented into the SLAM algorithm. Therefore, the results do not include this data. A rough performance calculation can be made to give an impression of the usability of the system.

With the maximum image crop size of 980 px × 620 px the system can scan 54 mm per iteration. One iteration takes an average of 1 s. Given that the SLAM algorithm needs to find enough known landmarks between the two images, it is estimated that a 50% overlap is needed between two image sets. This gives a scan speed of 27 mm/s.

## 7.4 Design Goals Evaluation

At the start of the project, a list of design goals and requirements is set up in the project plan and literature research. For this project, a total of twelve design goals are derived that all contribute to the project goal "design and develop a prototype 3D surface reconstruction camera system for monitoring skin diseases".

**Skin surface data acquisition** Starting at the first design goal, "skin surface data acquisition" (Section 3.1), the handheld scanner is equipped with two cameras that can shoot images with a maximum resolution of 3264 px × 2464 px and a frame rate of 21 FPS. The stereo camera setup can measure depths with a (depth)resolution of 0.0829 mm (Equation 6) and has a pixel density of 324 px/mm$^2$, given the used working distance of 150 mm. The cameras output ram 3x10-bit color information, but this is down-scaled to 3x8-bit in the image pre-processing stage. All the mentioned specifications comply with the set requirements of the first design goal.

**Inertial measurements** The second design goal (Section 3.2) is with respect to the implementation of an inertial sensor meant for sensor fusion with the visual SLAM algorithm. In the prototype handheld scanner, an IMU from Xsens is embedded. This sensor can give angular rates, acceleration, and current rotations in quaternions and Euler angles. The Xsens sensor can also provide data on the temperature and magnetic fields influencing the sensor. The current data synchronization method lets the sensor outputting measurements at full speed (100Hz) and storing the measurements in a circular buffer. Then, when the software requests a new stereo camera image pair, a matching inertial measurement is pulled from the buffer. The requirements for this design goal are not strict, as the idea of the implementation of an inertial sensor to improve the SLAM algorithm is still a hypothesis, and no specific requirements for the inertial data are available.

**Surface illumination** For the "surface illumination" design goal (Section 3.3), the same projector is used as was implemented in the project of Grimm (2020). The illumination of the surface and pattern projects did not have a high priority in this project. Therefore, only the basics for controlling the illumination unit have been implemented. Next to the projector also two high-power LED strips were added to the prototype for stable illumination of the test object at experiments. One of the requirements for this design goal is that the camera exposure times could be lower than 10 ms, at the experiments performed in the previous chapter, a shutter time of 34 $\mu$s is used. This fast shutter time is accomplished by using the two high-power LED strips.

**Camera sensor control** The "camera sensor control" design goal (Section 3.4) has a requirement that the focus, white balance, exposure time, gain, and other camera parameters of the camera modules can be controlled. The control of these parameters must be consistent, even when the system is powered off. In the design & implementation chapter in the section about the camera sensors (Section 5.2) the features of the camera sensors are described. The feature set of the camera sensors comply with the set requirements of this design goal.

**Visual inertial SLAM** One of the design goals is to implement the acquired inertial data into the currently used visual SLAM algorithm (Section 3.5). Within the limited time of the project and the limitations imposed by the SARS-CoV-2 pandemic that was ongoing during this project, not all design goals were reached. The implementation of the inertial data into the visual SLAM is because of these limitations not completed. The current status of the implementation is that the orientation data can be sent to the SLAM server, but the SLAM algorithm itself does not use this orientation data yet.

**Embedded software pipeline** The "embedded software pipeline" design objective (Section 3.6) is an overarching objective including multiple implementations of different systems. To meet this design objective, a system architecture design is made that makes it possible to implement the various components, like the GPU optimized landmark acquisition algorithm designed by Wolters (2021). In Section 7.3 a summarization of the whole system pipeline is given by using the experiments as an example. This shows that the prototype system works and that a pipeline has been constructed from data acquisition to the SLAM algorithm running on an external device. Within this system, the landmark acquisition and SLAM optimizations from Wolters are implemented into the system. This is made possible by the component selection and system architecture. Therefore, this design objective is completed.

**Cost effective design** The "cost-effective" design goal (Section 3.7) is a straight forward one. The requirement for this objective is that the costs of the system should stay around €1500.00. From the bill of materials found in Appendix D, it can be seen that the total cost of the project is €1280.87 and thus lower than the set (soft)limit, thereby meeting the design objective.

**Noise pattern projection** The "noise pattern projection" design goal (Section 3.8) aimed at the addition of introducing noise pattern projection on the to-be-scanned surface. This to reduce the adverse effects of specular reflection on the surface. The pattern projection can be implemented using the illumination unit available in the prototype, but the noise pattern projection technique is not yet implemented within this project.

**Fast imaging** The design objective for "fast imaging" (Section 3.9) has a requirement that the throughput time should be under 3 s. With the test results shown in Figure 18 it can be seen that the throughput times are heavily dependant on the input size of the images. For example; when an image size of 640 px × 480 px throughput times of around 400 ms can be seen. Following the results, it can be concluded that this requirement is met. However, it is advised to re-evaluate this design goal in the future, as 3 s is fine for a prototype but may be too slow for actual testing on human skin.

**Handheld System** For the second to last design objective, "handheld system" (Section 3.10), the goal is to design a lightweight system that could be handheld. From research is known that the standard weight limits for a handheld device are between the 400 g and 2.3 kg depending on the needed accuracy when handling the device. From some self-tests, a requirement of 600 g was picked. While in the design phase, the weight of the components was taken into account, this was not the main focus. Therefore, the weight of the prototype handheld scanner is currently 1377 g, which is still lower than the upper limit of 2.3 kg but higher than the wanted 600 g.

**Wireless system** The last design objective is the "wireless system" objective (Section 3.11) has a goal that the handheld scanner can be used wirelessly. A TCP/IP protocol is used for the communication using a Wi-Fi USB adapter in the experiments. In the future, this USB adapter could be replaced by an M.2 compatible Wi-Fi adapter giving a smaller form factor of the adapter. A wired connection is used for the power supply, as no time was available to experiment with battery-powered solutions. Therefore, this design objective is partly completed.

# 8 Conclusions & recommendations

This last chapter summarizes the designs and experiment results of the prototype handheld 3D surface reconstruction design. After the discussion of the conclusions, a list of recommendations for future research is given.

## 8.1 Conclusion

This project aimed to design and develop a prototype handheld 3D surface reconstruction system designed for monitoring skin diseases. As discussed in the introduction of this thesis, the goal is to make a prototype system that can reconstruct arbitrary test objects, still with the system's requirements bound to the monitorization of skin diseases.

The system's design is based on the outcome of a design space exploration that explored multiple solutions for designing problems at hand. The system architecture of the prototype has two main systems; the handheld scanner and the external SLAM server. The handheld scanner is designed to capture stereo images, measure inertial data, undistort images, find landmarks, and sends the resulting data wirelessly to the SLAM server. The SLAM server runs a GPU-optimized ES-EKF-SLAM algorithm that is used to reconstruct 3D surfaces. The "pre-processing" of the data on the Jetson Nano (embedded controller) has as advantage that not the stereo images have to be sent to the SLAM sever, but only the processed images, thus the landmark measurements and orientation data. This method of performing the landmark acquisition reduces the stress on the wireless communication.

To evaluate the designed system, a 3D printable test setup is designed and realized. This test setup is designed to hold the handheld scanner and rotate the whole scanner around a stationary platform with a fixed radius. An experiment was performed by utilizing the test setup in combination with a specifically picked test object. The selected test object has a cylindrical shape with a diameter of 45 mm with an image wrapped around it. The wrapped image was chosen such that the landmark acquisition algorithm would have no problem detecting strong keypoint matches. The result of the SLAM algorithm shows that the shape of the test object is correctly reconstructed and the camera's location. Next to evaluating prototypes function, the performance of the software of the handheld scanner is evaluated. The evaluation of the image acquisition shows that the system is capable of handling input image sizes up to 980 px $\times$ 620 px. Evaluating the processing of the data indicates that the system performance is dependent on the used input size. For example, The average throughput time for a resolution of 980 px $\times$ 620 px is around 1 s, this includes; loading the data, finding the landmarks, and sending the landmarks to the external SLAM server.

Concluding the results. The prototype can scan objects, process them, and reconstruct the surface of that object. Within the performed experiments, a specifically selected test object is selected, and the parameters of the landmark acquisition are tuned to the test object to get to the results. This prototype shows that the system works but that the algorithms' robustness must be improved for the system to be usable on a multitude of surfaces. In this project, the prototype is not tested on human skin or people with skin diseases as it was not the project's goal. However, the system should be suitable for the 3D reconstruction of skin as the system can theoretically acquire the needed data with the required accuracy to monitor skin diseases.

## 8.2 Recommendations

While working on this project, more information about the system is gathered, combined with some unfinished work of this project from a list of recommendations for future work. In the paragraphs below, each recommendation is separately discussed.

**Jetson Improvement** As discussed multiple times, the imitating factor for the throughput time is the processing power of the Jetson Nano. The image acquisition and keypoint acquisition software can significantly benefit from an improvement in processing power. A better, more expensive version of the Jetson, for example, the Jetson Javier NX offers more GPU cores and a better CPU.

**More Processing on GPU** Processing data can significantly benefit from concurrent programming, as most image processing computations are applied per pixel or row of pixels. This is the reason why the landmark acquisition runs on the GPU. Currently, the images are captured and cropped on the GPU. Then send to the CPU for synchronization and undistortion. After that, the images are sent back to the GPU for the landmark acquisition. Finally, the resulting landmark matches are sent back to the CPU to transmit the measurements to the SLAM server. This pipeline can be optimized by performing the synchronization and undistortion on the GPU as well. This prevents unnecessary data hand-over between the CPU and GPU.

**IMU-SLAM Sensor Fusion** Because of the limited time of the project, the acquired inertial data is not implemented in the SLAM algorithm. Thus, no experiment was performed to see if the addition of an inertial sensor would reduce the rotation/translation ambiguity problem. Within the performed experiments, it can be seen that the SLAM algorithm indeed has a problem with small distinguishing small rotations and translations. The IMU data gathered during the experiments shows promising results, as the orientation seems far more accurate than the orientation results of the SLAM algorithm. Therefore, it is recommended to implement the inertial data in the SLAM algorithm in a future project. Expected is that this would significantly improve the stability and accuracy of the system.

**Single Camera SLAM** While working on this project, an idea was suggested that it may be possible to remove one of the cameras when the IMU was implemented correctly. Removing one camera from the system has several advantages; less weight, lower costs, fewer data to process, and smaller in size. It may be interesting to examine this hypothesis when the IMU data is correctly implemented in the SLAM algorithm and tested.

**Hardware Synchronization** As discussed multiple times within this report, the synchronization of all components is done softwarematically. This method works, but the lag/sync delay between the different sensors is within the order of milliseconds. The lag between the sensor could be significantly reduced when using hardware synchronization. The Xsens IMU has an option for hardware synchronization by using one of its GPIO pins. The camera modules do not have an out-of-the-box option for synchronization, but solutions exist for this. There are synchronizer boards available that make sure that the shutters close and open at the same time. Therefore, the recommendation is to investigate the sensor synchronization further to get better-synchronized data and thereby less disparity error.

**Image undistortion** The captured images are undistorted on the Jetson Nano before being pushed into the landmark finder algorithm. The computational intensity of the undistorting algorithm is high. To increase the throughput time of the system, it is desirable to reduce or remove this undistortion function. The cameras used in this project do not have a significant radial distortion in the lenses. The distortion in the images is therefore not that prominent. The low radial distortion and the fact that the images are cropped from the center out gives that there is almost no need for undistorting the images. There is an option to turn the undistortion function on or off within the embedded software to improve the throughput time. If the distortion in the images is still too significant and gives problems, this can be solved by implementing a point-undistorting function on the (much faster) SLAM server.

# Acknowledgements

At the end of this thesis and thus the end of my project, I look back at an interesting, instructive, challenging, and maybe a little bit strange period due to the COVID-19 pandemic. I am thankful for many people, whom without, I would not have been in the position I am today.

I want to start with expressing my gratitude to my daily supervisor, dr.ir. Ferdi van der Heijden. Thank you for giving me the opportunity to work on this project. Thank you for your guidance and knowledge throughout this project that helped me keep my focus and reach my goals. I also want to thank dr.ir. Jan Broenink for the valuable input and expertise in this project.

I also want to thank the guys from the project group led by Ferdi. It was exciting and helpful to have the bi-weekly update meetings. I want to express my special thanks to Stijn Wolters for helping me with my project and for the sometimes much-needed coffee breaks.

Next, I want to thank the people from the BMPI group. Thank you for the interesting meetings. Even though it maybe was a short time that I joined your project group, it was interesting to be in such a project group and see what you were working on.

Next, I want to thank my friends and former roommates from our beautiful student house *Club Huyzch*. It was a blessing living with you; I had so much fun while living there. You guys gave me a great time as a student in Enschede. I also want to thank my friends who had the needed beers ready for me and provided the sometimes much-needed little breaks from working on this project.

I want to thank my family who made this all possible. You always supported, motivated, and inspired me throughout my life. You gave me the confidence and the resources to develop myself and to work on my career path.

Lastly, I want to thank my girlfriend Amber. Thank you for being there for me throughout all the years we have been together. You always inspire me, make me laugh, and support me when I needed it. Also, thank you for hearing out all my (maybe too technical) stories and rants about problems and bugs throughout my project. It is great to have someone that is genuinely interested in your work while you may not always know what I was talking about. Without you, I am not sure if I could make it to the point I am today, so once again, thank you.

# Appendices

## A PASI Score Images

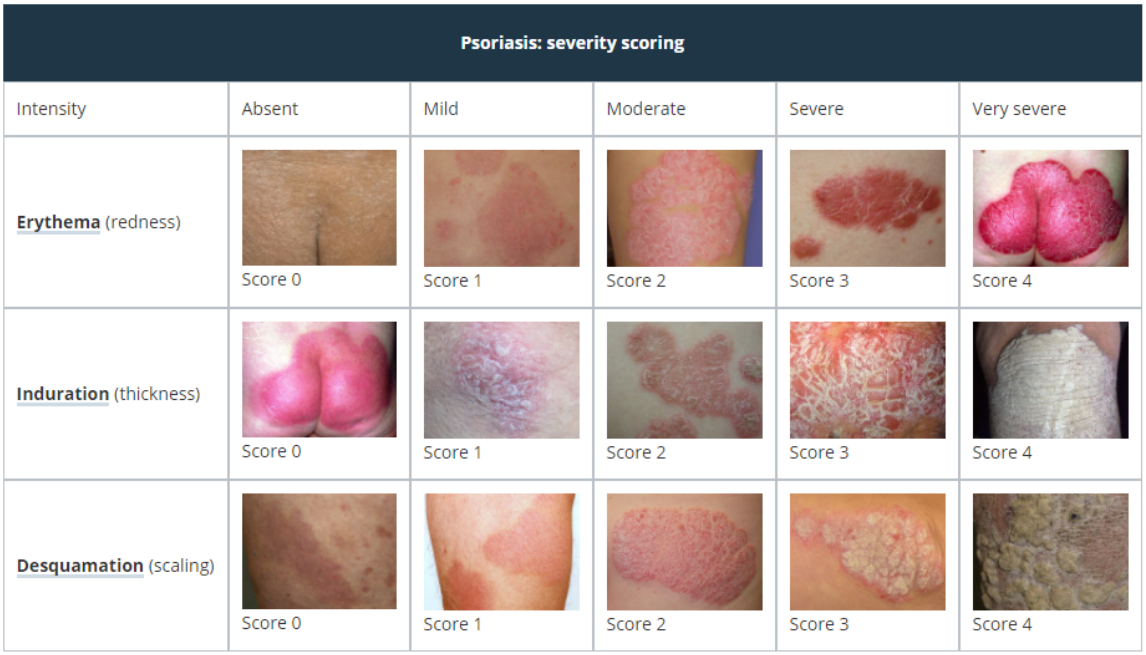



Figure 24: PASI score images (Oakley, 2009)

# B IMU Terminology

- **Inertial system**: A system that measures linear and/or angular motion by processing the outputs of one or more inertial sensors to perform a given task.
- **Inertial measurement unit (IMU)**: An inertial system that measures linear and angular motion in three dimensions without external reference.
  - The sensors used are usually accelerometers and gyroscopes.
  - An IMU can be either strapdown or gimbaled.
  - The outputs of an IMU are either incremental angles or angular velocities and either incremental velocities or linear accelerations in the ISA frame. If the IMU includes gimbals, the gimbal angles are also output.
- **Inertial navigation system (INS)**: An inertial system that estimates the vehicle's position, attitude, and velocity as a function of time in the specified navigation frame using the outputs of an IMU, a reference clock, a model of the gravitational field. An INS requires initialization.
  - The INS outputs are frequently time tagged to allow synchronization of the data with other systems.
  - Inertial navigation errors grow with time, but can be bounded by aiding.
  - An INS can be either strapdown or gimbaled.
- **Inertial reference unit (IRU)**: An inertial system that measures inertial angular motion in three dimensions without external reference.
  - The sensors used are usually gyroscopes.
- **Inertial sensor assembly (ISA)**: A structure incorporating multiple inertial sensors in fixed orientations relative to each other.
  - This is sometimes referred to as the instrument cluster or, in certain types of gimbaled IMU, as the stable element or stable platform.
- **Inertial navigation computations**: Either the gyro outputs are used to compute the orientation of the ISA with respect to the navigation frame, or are used to mechanically stabilize the ISA using gimbals or other means. The IMU accelerometer outputs are compensated for size and lever arm effects and transformed to the navigation frame. In a mechanically stabilized system the ISA may be maintained in a navigation frame and as such the transformation is not needed. The navigation frame accelerometer outputs are compensated for the inertial rotation of the navigation frame. The modeled gravitational acceleration is added to the compensated navigation frame accelerometer outputs. The result is numerically integrated to obtain the vehicle motion.
  - In a strapdown INS, the computation rate must be high enough (typically on the order of hundreds or thousands of hertz) to reduce the effects of coning and sculling.
- **Inertial stabilization**: Maintenance of the ISA in a fixed inertial orientation by servoing the gimbals to null the gyro outputs.
  - Gyroscope drift and control loop limit cycles result in deviations from inertial orientation.
- **MEMS**: MEMS is a acronym for Microelectromechanical systems. Gyroscopes, accelerometers, and magnetometers can be build as a MEMS device.

# C SLAM Server Power Consumption

Table 17: Workstation Power Consumption Calculation

| Property | Value |
|---|---|
| Motherboard | Mini-ITX |
| Socket | Socket AM4 |
| CPU | 1 x AMD Ryzen 5 5600X |
| CPU Speed | 3700MHz |
| CPU Vcore | 1.1V |
| CPU Utilization | 90% |
| Memory | 2 x 8GB DDR4 Module |
| Video Card Set 1 | 1 x NVIDIA GeForce RTX 2080 Ti |
| Core Clock | 1350MHz |
| Memory Clock | 1750MHz |
| Storage | 1 x M.2 NVMe SSD |
| Keyboard | 1 x Standard Keyboard |
| Mouse | 1 x Standard Mouse |
| Fan | 2 x 140mm |
| Computer Utilization | 16 hours per day |
| Gaming / Heavy 3D Application Time | 2 hours per day |
| | |
| Load Wattage | 379W |
| Recommended Wattage | **429W** |
| Amperage | +3.3V: 8.8A, +5V: 6.3A, +12V: 30.7A |
| Recommended UPS Rating | 750VA |
| OuterVision PSU Calculator part list https://outervision.com/b/ZrMhqb | |

# D Bill of Materials

Table 18: Bill of Materials

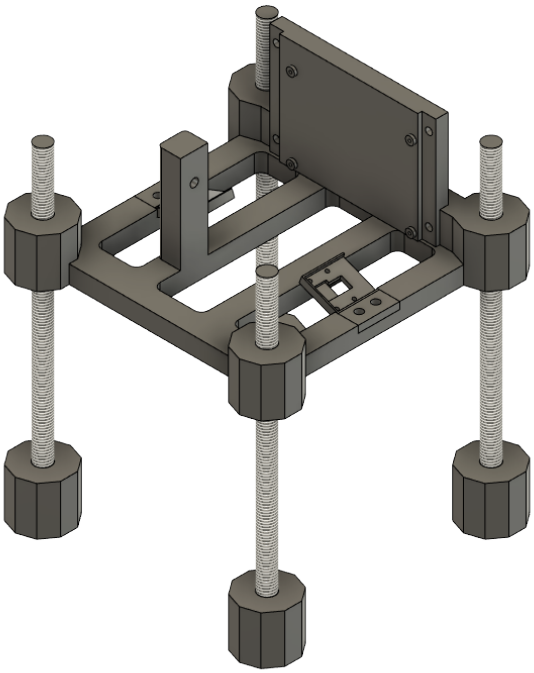
| Sub-system | Component | Product | Price |
|---|---|---|---|
| SLAM Server | Processor | AMD Ryzen 5 5600X | €369.95 |
| | Graphics Card | Gigabyte GeForce RTX 2080 Ti Gaming OC | €- |
| | Random Access Memory | Corsair Vengeance LPX 2x8GB 3200MHz | €94.99 |
| | Motherboard | Gigabyte B550M DS3H | €85.00 |
| | Solid Sate Drive | Kingston A2000 500 GB | €69.99 |
| | Power Supply Unit | Corsair TX-M Series TX650M V2 | €87.95 |
| | Case | be quiet! Pure Base 500 Black | €74.99 |
| Embedded Controller | SBC with GPU | Nvidia Jetson Xavier NX | €506.99 |
| | SBC with GPU | Nvidia Jetson Nano | €- |
| Stereo Camera Setup | Camera 1 | Raspberry Pi Camera Module V2 | €28.06 |
| | Camera 2 | Raspberry Pi Camera Module V2 | €28.06 |
| Motion Tracking | AHRS system | Xsens MTi-3 Dev Kit | €441.88 |
| | | **Total with new Jetson Xavier NX** | **€1,787.86** |
| | | **Total with current Jetson Nano** | **€1,280.87** |

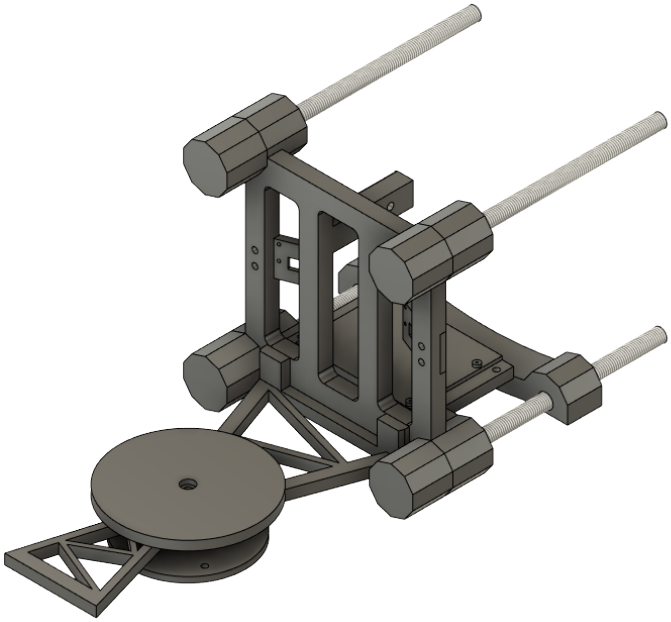
# E Camera Control Options

An OpenCV VideoCapture instance can be used to connect to the GStreamer pipeline. This method gives the option to control different parameters of the pipeline when initiating it. The options are listed below.

- Crop size (crop is centered)
- Exposure time range
- Gain range
- ISP digital range
- White-balance mode:
    0. off
    1. auto
    2. incandescent
    3. fluorescent
    4. warm-fluorescent
    5. daylight
    6. cloudy-daylight
    7. twilight
    8. shade
    9. manual
- Flip mode:
    0. none - Identity (no rotation)
    1. counterclockwise - Rotate counter-clockwise 90 degrees
    2. rotate-180 - Rotate 180 degrees
    3. clockwise - Rotate clockwise 90 degrees
    4. horizontal-flip - Flip horizontally
    5. upper-right-diagonal - Flip across upper right/lower left diagonal
    6. vertical-flip - Flip vertically
    7. upper-left-diagonal - Flip across upper left/low

# F 3D Designs





# G Software Usage

The embedded software on the handheld scanner is programmed on C++ for fast and optimized performance. The software is written in a object orientated fashion to keep the code clear and understandable. The usage of the software is designed to be easy. In the list below the program options are shown. These program options can be used within the command line.

| Option | Default Value | Description |
|---|---|---|
| help | - | produce help message |
| sensorWidth | 3264 | set width of image sensor (px) |
| sensorHeight | 2464 | set hight of image sensor (px) |
| imgWidth, w | 640 | set desired width of image (px) |
| imgHeight,h | 480 | set desired hight of image (px) |
| matcher | "AGPU" | set landmark matcher type |
| calibrationFile | "calibration3264_2464.yml" | set path to calibration file |
| ipAddress" | "192.168.10.139" | set IP address of the SLAM server |
| undistort | false | set to undistort images |
| saveImages | false | set to save images |
| saveInertialData | false | set to save inertial data |
| saveMeasurements | false | set to save measurements |
| transmit | false | set to data transmission |

Table 19: Program options embedded software

# References


Bai, J. et al. (2019). "A Novel Feedback Mechanism-Based Stereo Visual-Inertial SLAM". In: *IEEE Access* 7, pp. 147721–147731. DOI: 10.1109/ACCESS.2019.2946352.

Basra, Mohammad KA and Muhammad Shahrukh (2009). "Burden of skin diseases". In: *Expert Review of Pharmacoeconomics & Outcomes Research* 9.3, pp. 271–283. DOI: 10.1586/erp.09.23. URL: https://doi.org/10.1586/erp.09.23.

Chiu, Dr Keith et al. (2008). "Psoriasis". In: *InnovAiT* 1.7, pp. 481–486. DOI: https://10.1093/innovait/inn089.

Fadzil, M. H. Ahmad, Hurriyatul Fitriyah, et al. (Mar. 2010). "Objective assessment of psoriasis lesion thickness for PASI scoring using 3D digital imaging". In: *World Academy of Science, Engineering and Technology, International Journal of Medical, Health, Biomedical, Bioengineering and Pharmaceutical Engineering* 63, pp. 109–115.

Fadzil, M. H. Ahmad, Dani Ihtatho, et al. (2009). "Objective assessment of psoriasis erythema for PASI scoring". In: *Journal of Medical Engineering & Technology* 33.7, pp. 516–524. DOI: 10.1080/07434610902744074.

Fink, Christine, Tobias Fuchs andAlexander Enk, and Holger A. Haenssle (Nov. 2018). "Design of an Algorithm for Automated, Computer-Guided PASI Measurements by Digital Image Analysis". In: *Journal of Medical Systems* 42. DOI: 10.1007/s10916-018-1110-7.

Grewal, Mohinder S., Lawrence R. Weill, and Angus P. Andrews (2001). *Global Positioning Systems, Inertial Navigation and Integration*. 1st ed. 605 Third Avenue, New York, NY, USA: Wiley-Interscience.

Grimm, Sander (2020). "Design of a 3D imaging system for psoriasis assessment". MA thesis. University of Twente - RAM.

Heijden, Ferdi van der (2016). *Stereo: from pixels to 3D surface mesh*. Tech. rep. P.O. Box 217 - 7500 AE Enschede - The Netherlands.

— (2019). *Camera models*. Tech. rep. P.O. Box 217 - 7500 AE Enschede - The Netherlands.

— (2020a). *EKF-SLAM*. Tech. rep. P.O. Box 217 - 7500 AE Enschede - The Netherlands.

— (2020b). *Visual Navigation*. Tech. rep. P.O. Box 217 - 7500 AE Enschede - The Netherlands.

*Human Factors criteria for hand held devices* (2004). Tech. rep.

IEEE (2009). "IEEE Standard for Inertial Systems Terminology". In: *IEEE Std 1559-2009*, pp. c1–30. DOI: 10.1109/IEEESTD.2009.5226540.

— (2019). "IEEE Standard for Inertial Sensor Terminology". In: *IEEE Std 528-2019*, pp. 1–35. DOI: 10.1109/IEEESTD.2019.8863799.

Jyothi, S.L. et al. (2021). "Drug delivery systems for the treatment of psoriasis: Current status and prospects". In: *Journal of Drug Delivery Science and Technology* 62, p. 102364. ISSN: 1773-2247. DOI: https://doi.org/10.1016/j.jddst.2021.102364. URL: https://www.sciencedirect.com/science/article/pii/S1773224721000447.

Kytö, Mikko, Mikko Nuutinen, and Pirkko Oittinen (Jan. 2011). "Method for measuring stereo camera depth accuracy based on stereoscopic vision". In: vol. 7864, pp. 78640I–1. DOI: 10.1117/12.872015.

Li, Y. and S. Lang (2019). "A Stereo-Based Visual-Inertial Odometry for SLAM". In: *2019 Chinese Automation Congress (CAC)*, pp. 594–598. DOI: 10.1109/CAC48633.2019.8997432.

Liu, Y. et al. (2018). "Stereo Visual-Inertial SLAM With Points and Lines". In: *IEEE Access* 6, pp. 69381–69392. DOI: 10.1109/ACCESS.2018.2880689.

*Medical Definition of Desquamation* (2018). https://www.medicinenet.com/desquamation/definition.htm [Accessed: Feb 18th 2021].

*Medical Definition of Erythema* (2018). https://www.medicinenet.com/erythema/definition.htm [Accessed: Feb 18th 2021].

*Medical Definition of Induration* (2018). https://www.medicinenet.com/induration/definition.htm [Accessed: Feb 18th 2021].

Nvidia (2018). Jetson Camera Architecture Stack https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/jetson_xavier_camera_soft_archi.html#wwpID0E0LC0HA.

Nvidia (2021). Nvidia Libargus Camera API https://docs.nvidia.com/jetson/l4t-multimedia/group__LibargusAPI.html.

Oakley, Dr. Amanda (2009). DermNet NZ PASI score https://dermnetnz.org/topics/pasi-score/.

Okun, Martin M. (2008). "Psoriasis Area and Severity Index: Nuts and bolts of measuring disease severity in psoriasis". In: *Clinics in Dermatology* 26.6, pp. 653–656. DOI: https://doi.org/10.1016/j.clindermatol.2008.08.002.

Rouse, Margaret (2020). *MoSCoW method*. https://searchsoftwarequality.techtarget.com/definition/MoSCoW-method [Accessed: Dec 12th 2020].

Shah, Nimish (2020). "3D stereovision for quantification of skin diseases". MA thesis. University of Twente - RAM.

Solá, Joan (2014). *Simulataneous localization and mapping with the extended Kalman Filter*. Tech. rep.

Steamworks (n.d.). SteamVR Tracking System https://partner.steamgames.com/vrlicensing.

Tizek, L. et al. (2019). "Skin diseases are more common than we think: screening results of an unreferred population at the Munich Oktoberfest". In: *Journal of the European Academy of Dermatology and Venereology* 33.7, pp. 1421–1428. DOI: https://doi.org/10.1111/jdv.15494. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jdv.15494.

Vydhyanathan, Arun and Giovanni Bellusci (2018). "The Next Generation Xsens Motion Trackers for Industrial Applications". In:

Wolf-Henning Boehncke, Michael P Schön (2015). "Psoriasis". In: *The Lancet* 386.9997, pp. 983–994. DOI: https://doi.org/10.1016/S0140-6736(14)61909-7..

Wolters, Stijn (2021). "Implementation of 3D visual ekf slam for handheld perfusion imaging". MA thesis. University of Twente - RAM.

Xsens (2019). *MTi 1-series Datasheet*. English. Version 2019.A. 40 pp.