TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Department of Electrical Power Engineering and Mechatronics

MOBILE HYPERSPECTRAL ACQUISITION SYSTEM

MOBIILNE HÜSTERSPEKTRAAL-PILDISALVESTUSE SÜSTEEM

MASTER THESIS

Student: Andrei Palshin

Student code: 187523MAHM

Supervisor: Dhanushka Chamara Liyanage

Co-supervisor: Mart Tamre, Professor

Tallinn, 2019

AUTHOR'S DECLARATION

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

"....." ..... 201.....

Author: .....
/signature /

Thesis is in accordance with terms and requirements

"....." ..... 201....

Supervisor: .....
/signature/

Accepted for defence

"....."..... 201... .

Chairman of theses defence commission: .....
/name and signature/

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student:	Andrei Palshin, 187523MAHM
Study programme, main speciality:	MAHM02/13 - Mechatronics
Supervisor(s):	Dhanushka Chamara Liyanage
Co-supervisor:	Professor Mart Tamre, +372 5120982
Consultant:	Svetlana Perepelkina, PhD, Associate Professor of Mechatronics Department of ITMO University

Thesis topic:

(in English) Mobile Hyperspectral Image Acquisition System

(in Estonian) Mobiilne hüsterspektraal-pildisalvestuse süsteem

Thesis main objectives:

1. Use hyperspectral technology
2. Develop a hyperspectral image acquisition method for NVIDIA Jetson TX2
3. Use NVIDIA Jetson TX2 and Ximea xiQ hyperspectral camera to capture hyperspectral images
4. Modifying and creating the artificial neural network (ANN) for classification purpose

Thesis tasks and time schedule:

№	Task description	Deadline
1.	Review of previous work	12.03.2019
2.	Study of hyperspectral imaging techniques, principles and features of the camera and single board computer	07.04.2019
3.	Development of hyperspectral image acquisition method	02.05.2019
4.	Neural network training	12.05.2019
5.	Testing of the system	15.05.2019

Language: English **Deadline for submission of thesis**: "21" May 2019

Student: Andrei Palshin "……."……2019

Supervisor: Dhanushka Chamara Liyanage "……."……2019

Consultant: "……."……2019

CONTENTS

PREFACE

Computer vision has an innumerable number of applications, and I am very glad that I had the opportunity to work on such an interesting direction as hyperspectral imaging.

This thesis is provided by the Department of Electrical and Power Engineering and Mechatronics of Tallinn University of Technology.

I am deeply grateful to my supervisor, Dhanushka Liyanage who throughout my studies at the Tallinn University of Technology, helped me find answers to all my questions and cope with the emerging difficulties. Under his guidance, I successfully completed the task and gained a lot of new knowledge.

I would like to thank my co-supervisor Prof. Mart Tamre for all the thesis-related suggestions and help with research paper.

Finally, I would like to thank Ali Zahavi, with whom we worked on the thesis-related research paper and successfully presented it at the conference REM 2019

Keywords: Hyperspectral imaging, XIMEA camera, NVIDIA Jetson TX2, Neural networks, Tensorflow

List of abbreviations and symbols

API - Application programming interface

BMP - Bitmap Picture

CMOS - Complementary metal–oxide–semiconductor

GPU - Graphics processing unit

NIR - Near-infrared spectrum

NN - Neural Network

RAW – RAW image format

RGB - Red Green Blue (color model)

UAV - Unmanned Aerial Vehicles

VIS - Visible spectrum

1. INTRODUCTION

1.1 General overview

The hyperspectral camera is capable of acquiring data well beyond the spectral range of the human eye, which is limited by a maximum wavelength of about 700 nm.
Hyperspectral imaging combines the power of digital imaging and spectroscopy. This method can be used in applications such as identifying different materials, classifying objects based on spectral signature obtained by processing hyperspectral data cubes. [1]
Examination of data obtained from hyperspectral imaging allows us to detect objects by their physicochemical composition, identify the species composition and state of the vegetation cover, determine the geological structure, identify the chemical composition of water, and much more. Different materials have different spectral signatures by which they can be identified.

Recent advances in sensor design and processing speed have cleared the path for a wide range of applications employing hyperspectral imaging, ranging from satellite-based/airborne remote sensing and military target detection to industrial quality control and lab applications in medicine and biophysics. Due to the rich information content in hyperspectral images, they are uniquely well suited for automated image processing, whether it is for online industrial monitoring or for remote sensing.

A few years ago, the company NVIDIA has made great progress in the creation of small, but very powerful embedded computers for machine learning. The latest generation of this embedded computer is NVIDIA Jetson TX2. By placing such a computer on board an unmanned aircraft, we can quickly and accurately classify objects and determine their composition from the air.

This thesis aims to develop a neural network image recognition system based on NVIDIA Jetson TX2 platform and Ximea xiQ hyperspectral camera, that will be part of an aerial hyperspectral data acquisition and processing system built on top of the unmanned aerial vehicle (UAV).

1.2 Motivation

I had the opportunity to engage in hyperspectral imaging during my internship at the University. Hyperspectral imaging is a powerful and versatile tool that can be used in a wide variety of practical applications. For example, we can analyse various animals and plants in their habitat, without interacting with them directly, take complex, multi-layered images of the surface of the earth from space or diagnose the state of sown areas.

We have done great and interesting work, the result of which was a speech at the 20th International Conference on Research and Education in Mechatronics (REM 2019) and publication of the work in IEEE Xplore Digital Library.

At the same time, I took the course Intelligent Control System, where I studied methods of building and training neural networks. This course gave me a great interest in neural networks and I wanted to apply them in practice. Hyperspectral imaging is a complex multidimensional structure that is difficult to analyse automatically. That is why the use of deep learning to recognize hyperspectral images shows very high efficiency and allows you to achieve unprecedented accuracy.

My growing interest in those fields and successfully defended the research paper gave me inspiration for this thesis work.

1.3 Research paper

Before starting work on the thesis, the author together with Ali Zahavi worked on writing a research paper devoted to the study of the influence of different light sources on the results of hyperspectral photography. The research article is called "Influence of Illumination Sources on Hyperspectral Imaging" [2], was successfully presented on REM 2019 and will be published in IEEE Xplore Digital Library.

1.4 Task definition

The aim of this work is to develop a hyperspectral image acquisition method using a GPU based single board computer (NVIDIA Jetson TX2) with Ximea xiQ hyperspectral imaging camera.
The system must be energy efficient, run on NVIDIA Jetson TX2 and designed in such a way that it can be integrated with other systems that will be installed on the same platform.

1.5 Description of tasks involved

1. Review of the previous research
2. Setup and configure equipment for HSI
3. Flash, configure and connect NVIDIA Jetson TX2 to the equipment for HSI
4. Study the principle of operation of the hyperspectral camera
5. Develop a method for acquisition hyperspectral images
6. Check the effectiveness of the developed method and make adjustments based on the results
4. Take a series of hyperspectral images.
5. Create datasets based on captured hyperspectral images
6. Train the neural network to classify images by the presence of real and artificial plants
7. Test the entire system

1.6 Structure of the thesis

In the introduction chapter, the main idea is introduced, the task definition and motivation are clarified, the structure of the thesis described, and the goal is specified as well.
The second chapter contains the theoretical basis of hyperspectral imaging, discusses basic concepts in deep learning and neural networks, materials used, and describes the hardware and software parts of the system.
In the description of the hardware platform, the scheme of the whole system and its individual components, such as NVIDIA Jetson TX2, Ximea xiQ, IMEC sensor and Thorlabs LTS300/M.

In the description of the software platform, a description of the entire system, the justification for the choice of programming language and libraries used, as well as additional third-party software. At the end of the Chapter, describes the method of obtaining images, their processing and stitching.

The third section contains a consistent description of the development of the system from setting up the environment and assembling the various components together, to the development of the method and training of the neural network based on the hyperspectral images.
At the beginning of the chapter, the assembly and connection of the system, the performance check, the system firmware and the installation of the necessary software are discussed. Then, the developed method and the program code that implements it are described in detail.
At the end of the section, based on the data obtained by the developed method, trains a neural network and describes the results of the entire system.

The fourth section contains the analysis of experimental results, conclusions and proposals for further development of the system.
In the fifth section lists references, and in the sixth Appendix.

2. DESCRIPTIONS, THEORETICAL BASIS, OVERVIEWS

2.1 Hyperspectral imaging

2.1.1 Overview

A hyperspectral image is a three-dimensional data array (data cube) that includes spatial information (2D) about an object, supplemented with spectral information (1D) for each spatial coordinate. In simple words, a hyperspectral image is a three-dimensional image, where the third dimension is spectral information in a limited range.

Below, Figure 1 shows the main differences between various visualization methods. Monochrome, RGB, Spectroscopy, multispectral and hyperspectral imaging. The monochrome image contains data on the light interacting with the sample at one wavelength, the RGB image at three different wavelengths (red, green and blue), spectroscopy allows collecting data over the entire available range, but only at one point of sample, while multispectral or hyperspectral image contains data on all wavelengths and information about their location on the sample.

However, multispectral imaging has a spectral resolution of more than 10 nm, and the number of spectral bands is limited to three to ten spaced bands, while commercially available HSI systems have a higher resolution (<5 nm) with spectral bands ranging from tens to hundreds in a continuous range [3]

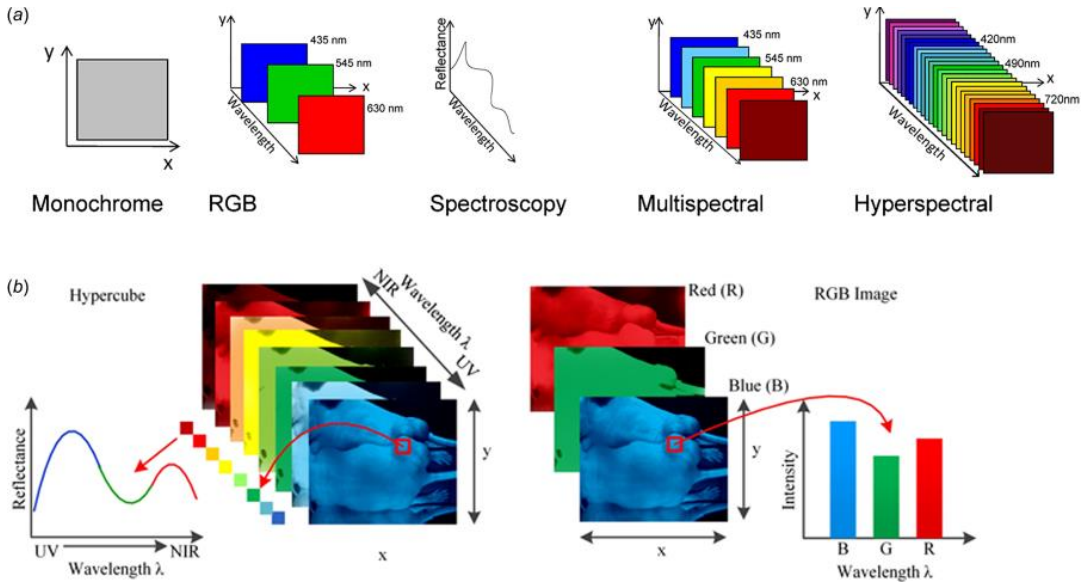

Figure 1 - Hyperspectral image [4]

There are three main methods used in hyperspectral image processing technology [5]:

- spectral scanning of the image, with sequential capture of full spatial information;
- spatial image scanning with sequential capture of common spectral data;
- snapshot method, simultaneous capture of all spectral and spatial information.

The first method works on the basis of optical bandpass filters (either tunable or with a fixed bandwidth), where the spectrum is scanned step by step.
In the second method, the resulting image is built along lines (push broom scan), and the use of moving optical elements is necessary.

The third type of device for obtaining hyperspectral images receives a spectrum using the so-called Fourier transform spectroscopy (FT) technique; In this technique, interferograms obtained using a scanning interferometer are used to determine the spectral composition of the luminous flux entering the interferometer.

First two methods lead to qualitative results but are inefficient in collecting and processing the entire volume of data.
In linear scanning systems where spatial measurement is done by movement or scanning (Figure 2), good vibration isolation or accurate beam direction information is required to reconstruct the image. In the system, scanning the wavelength, there is a spectral blurring, if movement occurs in the scan area.

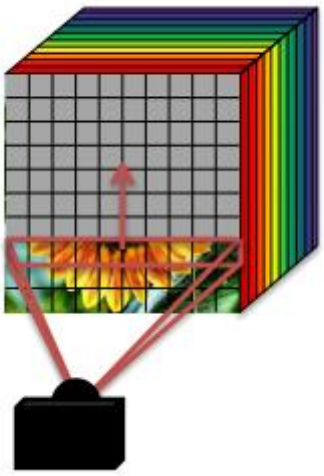

Figure 2 - Pushbroom [5]

Unlike the previous two methods, the third method takes the entire image at one point in time (Figure 3), rather than piecing it together. It even allows capturing hyperspectral videos.

Figure 3 - Snapshot [5]

In this thesis, the equipment that performs image acquisition uses the pushbroom method (more about the hyperspectral camera in chapter 1.3.3).

The result of the hyperspectral imaging is a three-dimensional array called data cube (Figure 4). Each pixel of this data cube contains information about the reflection of light over the entire range of the camera. According to the data obtained from the data cube, the composition of the object can be analysed, and various dependencies and areas of interest highlighted.

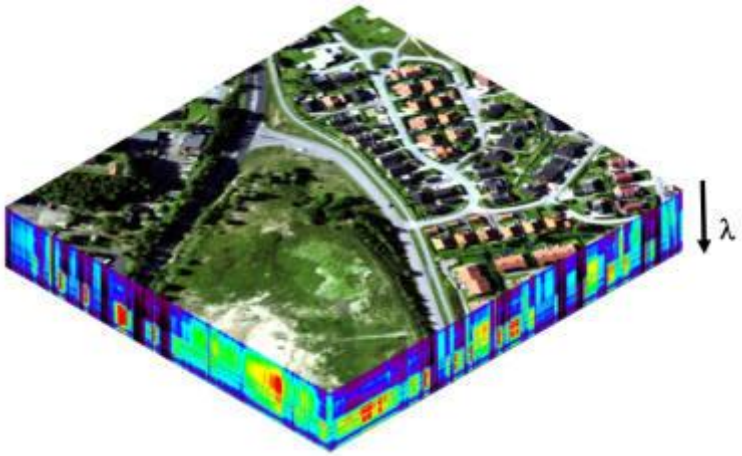

Figure 4 – Data cube [6]

2.1.2 Hyperspectral imaging applications

Hyperspectral cameras are successfully used in various industries and fields of activity [7]. The analysis of the images obtained allows for a detailed classification by the selected parameters and to solve many problems:

- search for oppressed trees prone to diseases or infected with bark beetles, etc.;
- monitoring of agricultural vegetation (in the field);
- research in cell biology;
- diagnosis of eye diseases;
- monitoring of wound healing;
- quality control of medicines in pharmacology.
- agricultural and soil mapping;
- geological mapping;
- sorting of food products on the conveyor belt and determining their quality;
- sorting of municipal solid waste by type;
- classification of materials of buildings and structures for topographic maps and plans;
- laboratory analysis of minerals and their classification;
- determination of the affected areas of the brain in neurosurgical operations;
- determination of cancer foci of internal organs;

2.1.3 Influence of Illumination Sources on Hyperspectral Imaging

In order to find the best illumination source for hyperspectral image acquisition, sun, fluorescent, light-emitting diodes (LED) and incandescent light sources have been used. The hyperspectral images were acquired using two hyperspectral cameras, Resonon Pika II in the range of 400-1000 nm and Resonon Near Infrared camera in the range of 1000-1700 nm. Then, the results have been compared with the literature.

This publication was written before the writing of this thesis and reveals the features of hyperspectral imaging.

The first chapter of the paper suggests the incandescent lights as the best illumination source in HSI. The second part of the paper presents light absorption spectra of different materials including Alumina Oxide (Al2O3), Aluminium Oxide plus Cubic Boron Nitride (cBN), Wood and unknown Metal Disk by using two hyperspectral cameras, Resonon Pika II and Resonon Near Infrared by the presence of incandescent lights as the illumination source. All experiments have been done in Mechatronics and Autonomous Systems Centre of Tallinn University of Technology. [2]

Graphs of light absorption of all these materials show that the use of incandescent lamps for hyperspectral imaging is optimal from the point of view of covering a wide spectral range of light. This light source adds no significant peaks throughout the operating spectrum. This makes it easy to neutralize its influence in the analysis of the results of hyperspectral imaging and further calculations.

2.2 Hardware platform

2.2.1 System description

The main components of the system are the NVIDIA Jetson TX2 board [8], the Ximea xiQ hyperspectral camera [9] and THORLABS LTS300/M [10] motion stage (Figure 5). All basic calculations are carried out on the Nvidia board, while all other devices get engaged in executing commands received from it.

The hyperspectral camera is placed on the motion base, which moves relative to the subject. The motion stage can move the motion base with a camera along its entire length to an accuracy of 5 μm.

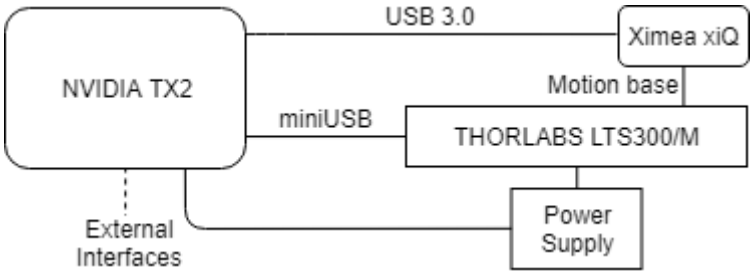

Figure 5– System schematic

When the program inside the NVIDIA Jetson TX2 decides to start shooting, a signal is sent to the motion stage, which sets it in motion. Motion base moves with the camera to the home position from which it is ready to start shooting.

After receiving a signal that the motion base is in the home position, the program synchronously sends a signal to move the motion base and to shoot one column of the image. The motion base passes a predetermined number of motion steps, while the camera takes the same number of shots.

Currently, the management of the motion stage is carried out not from the developed program itself, but with the help of third-party software. For more information on how the decision was made, see the software development section below.

Once the shooting is complete, the captured data is transferred to NVIDIA via high-speed USB 3.0, where it is stored in a buffer, waiting for further processing.

The system operates from two power supplies of 12V 2A and 19V 4.7A, which for convenience are combined into a single element in the schematic above.

Figure 6 shows an image of the assembled system in working condition.

The following elements are marked on the image:

1. Ximea xiQ camera
2. THORLABS LTS300/M motion stage
3. NVIDIA Jetson TX2
4. Light absorbing fabric
5. Artificial plant

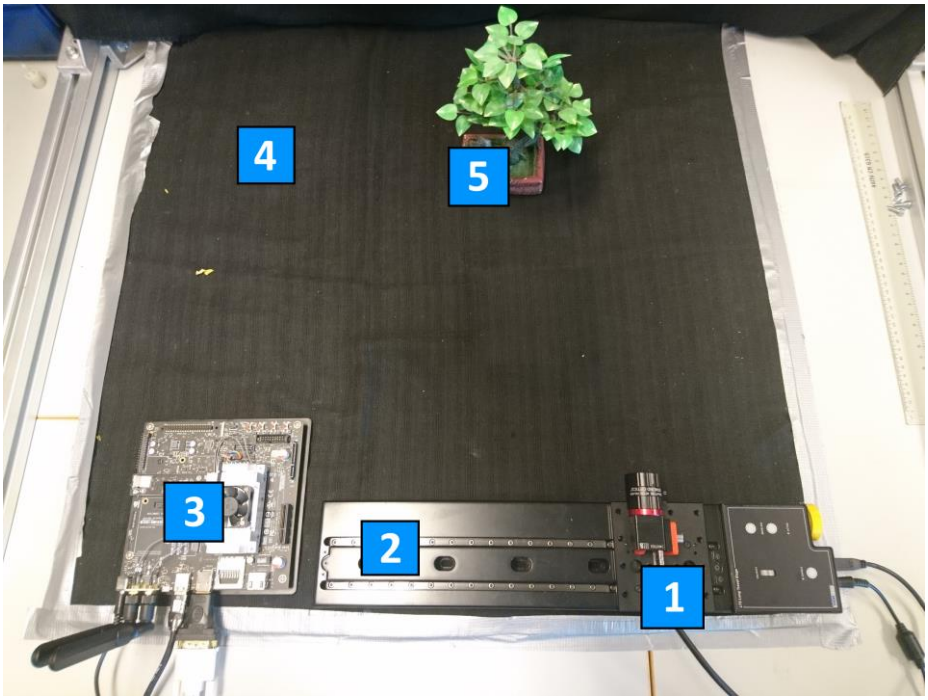

Figure 6 - Assembled system

2.2.2 NVIDIA TX2

Jetson TX2 is one of the fastest, most power-efficient embedded AI computing device at this moment (Figure 7). With a compute performance of more than 1 Teraflops, the Jetson TX2 is the ideal solution for deploying artificial intelligence in remote areas with poorly developed or expensive infrastructure for connecting to the Internet. Jetson TX2 provides minimal application response time and high responsiveness in near real-time, which is a key factor for smart machines that require a high level of autonomy.

The use of this platform offers great opportunities for working with hyperspectral images.

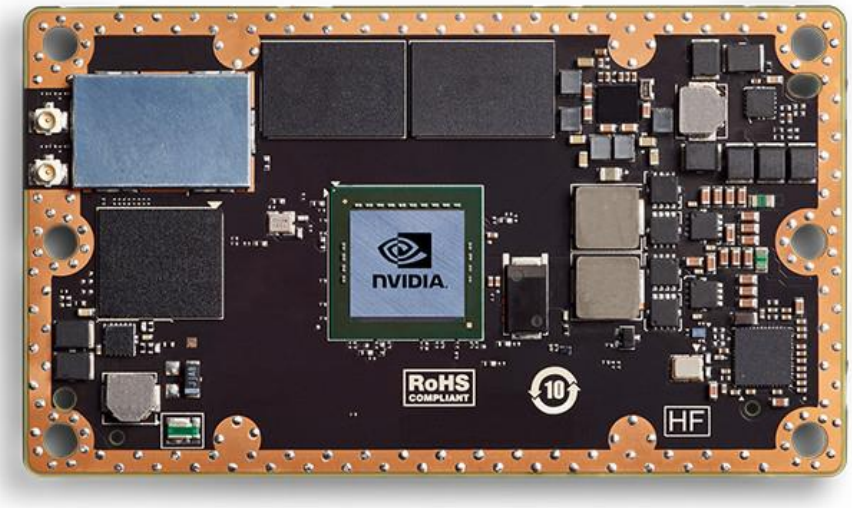

Figure 7 - NVIDIA Jetson TX2 [8]

NVIDIA Jetson TX2 supports high-resolution sensors, can simultaneously process information from multiple sensors and run several neural networks simultaneously. It also supports many popular AI frameworks, which allows developers to integrate their favourite models and frameworks.

The configuration of the single-chip system used in the Jetson TX2 [11] includes two 64-bit CPU cores Denver 2, four CPU cores ARM Cortex-A57 and 256 CUDA cores on the Pascal architecture. The system supports encoding and decoding of 4K x 2K video with a frame rate of 60 f/s, connecting up to six cameras over 12 lines of CSI with a capacity of 2,5 Gb/s each (Figure 8). The memory subsystem includes 8 GB LPDDR4 memory with a bandwidth of 58.4 GB/s and a 32 GB eMMC module. There is a Gigabit Ethernet port and Wi-Fi 802.11ac and Bluetooth wireless connectivity. The components of the module are assembled on board with dimensions of 50 x 87 mm [12].

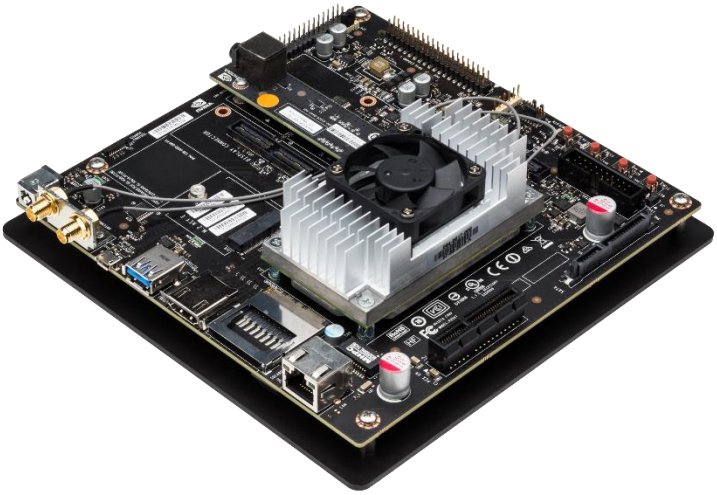

Figure 8 – NVIDIA Jetson TX2 Developer Kit [11]

Of all the variety of opportunities provided, for the implementation of a neural network based on this platform, the most important thing is system performance in order to process large amounts of data received from a hyperspectral camera.

2.2.3 Ximea xiQ

Ximea xiQ MQ022HG-IM-LS150 VISNIR [13] is an ultra-compact USB 3.0 industrial hyperspectral near-infrared camera with a tiny footprint, 2,2 megapixel sensor that can shoot at 90 frames per second, spectral range 470-900 nm, low thermal dissipation and it's all combined on a small single cube with a lens (Figure 9).

Figure 9 - Ximea xiQ MQ022HG-IM-LS150 VISNIR [13]

Due to the fact that the camera consumes less than 2 watts, it does not need a separate power supply and it is powered directly from the TX2 board.

An EDMUND OPTICS 35mm C Series VIS-NIR [14] lens is used with the camera (Figure 10). It has a focal length of 35mm and an aperture of F1,65.

Figure 10 - EDMUND OPTICS 35mm C Series VIS-NIR lens [14]

2.2.4 IMEC sensor

The camera has an HSI sensor from IMEC [15]. It's a linear sensor. The sensor provides data on 150 spectral bands from 470 to 900 nm (Figure 11).

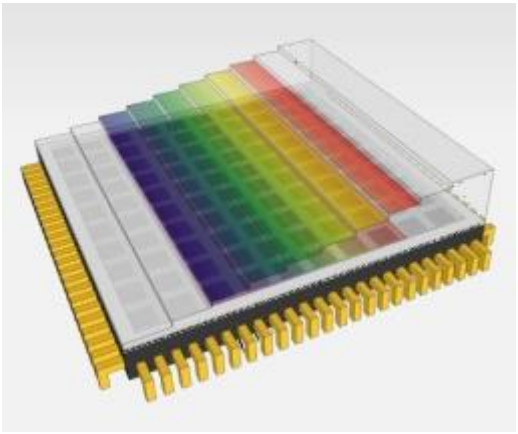

Figure 11 - Linear HSI imaging sensor [15]

The sensor technology used in HSI-camera is based on standard CMOS area sensors, with a native resolution of 2048*1088 pixels. Hyperspectral filters are added at wafer-level on top of the pixel structure of the sensor.

The sensor is designed to work in a specific spectral range (active range). The sensor behaviour is not defined outside of this range. Some of the filters added to the sensor surface with their sensitivity peak at the edges of the active range do have two sensitivity peaks (first and second order response).

The sensor works in such a way that the RAW image is obtained from a set of bands with a width of 5 pixels. Each of these bands is responsible for a certain spectral range. The general structure of the RAW image is shown in Figure 12.

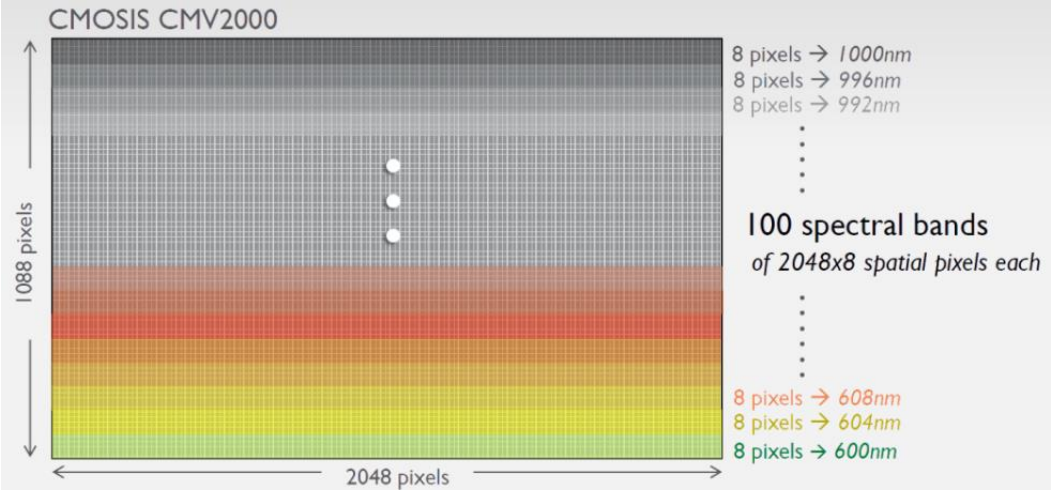

Figure 12 - RAW data from the sensor [15]

The image above shows 150+ bands between 470 and 900 nm in steps of 4 nm, each with a size of 2048x8 pixels. The next Chapter will consider the development of a program that will process such data. An example of the image obtained from the hyperspectral camera is shown below in Figure 13.

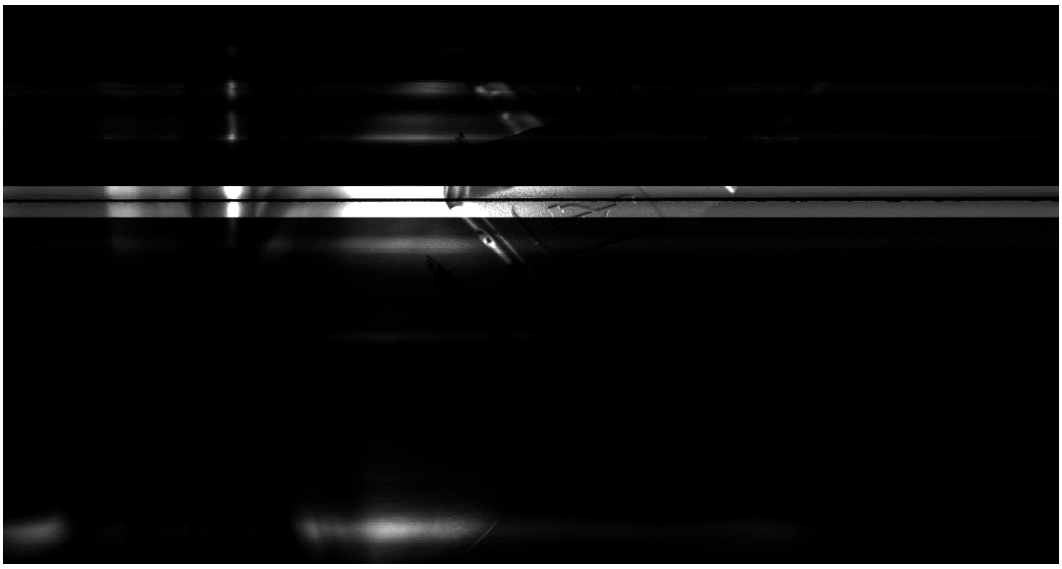

Figure 13 – RAW Hyperspectral image

2.2.5 Thorlabs LTS300/M

In order to capture the entire object, the hyperspectral camera must move from one edge of the image to the other. Translation Stage with Stepper Motor is great for this task.
THORLABS company produces the translation stage with a stepper motor and integrated controller LTS300 / M (Figure 14).

Figure 14 - THORLABS LTS300/M [10]

It provides 300 mm of linear travel for loads as great as 15 kg when mounted horizontally and 4 kg (8,8 lbs) when mounted vertically. Each stage features an on-axis accuracy of 5.0 µm (Min) [6].
This translation stage contains a controller inside of itself, therefore for its control, it is only necessary to transmit commands via USB according to a special protocol.

2.3 Software platform

2.3.1 Linux and Jetpack

Jetson platform is adapted to work with Linux. Nvidia has created a wrapper for its platform, which is installed on top of the operating system. They called it JetPack [16].
Jetson TX2 is supported by a single software stack, which allows companies to deploy systems anywhere. The Jetson platform is equipped with a Jetpack developer toolkit that contains the platform support package, Linux OS, NVIDIA CUDA (R), and is compatible with other platforms. DeepStream allows developers to quickly develop and install Jetson-based video analysis systems.

The JetPack development toolkit initially includes libraries for deep learning, computer vision, accelerated computing and multimedia, and also supports drivers for a variety of sensors. In

addition, NVIDIA is developing API with graphical acceleration for such important tasks of autonomous machines as determining the depth, building a route and recognizing objects.

Included Jetpack NVIDIA offers a wide range of tools for developers that speed up the work and contain detailed information about the application, performance and power system that allows to quickly optimize and customize the code.

Below is a schematic representation (Figure 15) of all the components that make up the jetpack.

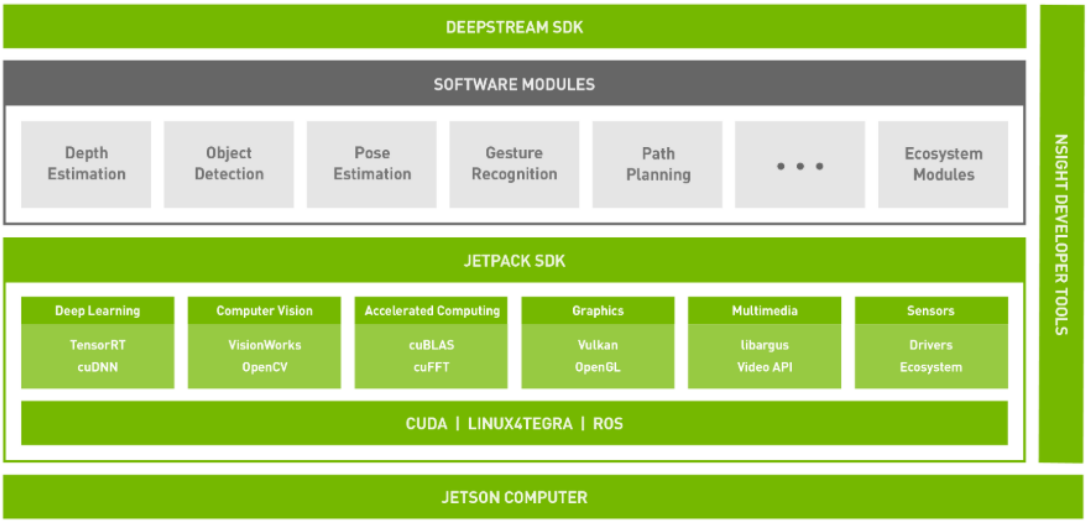

Figure 15 - JetPack schematic [16]

2.3.2 Python

Python is an extremely powerful programming language that is used to work with scientific data in a variety of ways. The main reason for which the Python was chosen is its high efficiency when working with huge multi-dimensional data sets and a large selection of tools for their processing [17]. Python and used libraries are multiplatform, which allows using the developed application not only on ARM64 and Linux but also on x86_64 and Windows systems.

In the developed application, hyperspectral data is stored in the numpy array format [18]. Using the built-in numpy functions to process this data shows a high speed, which is enough for processing real-time data.

2.3.3 TensorFlow

TensorFlow is an open source library created by Google that is used in the development of systems that use machine learning technologies. This library includes the implementation of many powerful algorithms designed to solve common machine learning problems, including pattern recognition and decision-making [19].

Keras is an open neural network library written in Python [20]. It is an add-on to the TensorFlow framework. It is aimed at operational work with deep learning networks, while designed to be compact, modular and expandable. This library contains numerous implementations of widely used neural network building blocks such as layers, target and transfer functions, optimizers, and a host of tools to simplify working with images and text.

2.3.4 XimeaCameraCapture

XimeaCameraCapture is a program developed by Dhanushka Chamara Liyanage [21], PhD student of the Department of Electrical Power Engineering and Mechatronics.

This program provides a great Toolkit to work with - Ximea xiQ MQ022HG-IM-LS150 VISNIR camera and THORLABS LTS300/M translation stage.

This program was used to test the operation of the camera, check the performance of individual parts of the system and analyse the resulting hyperspectral images. The interface of XimeaCameraCapture showed below in Figure 16.

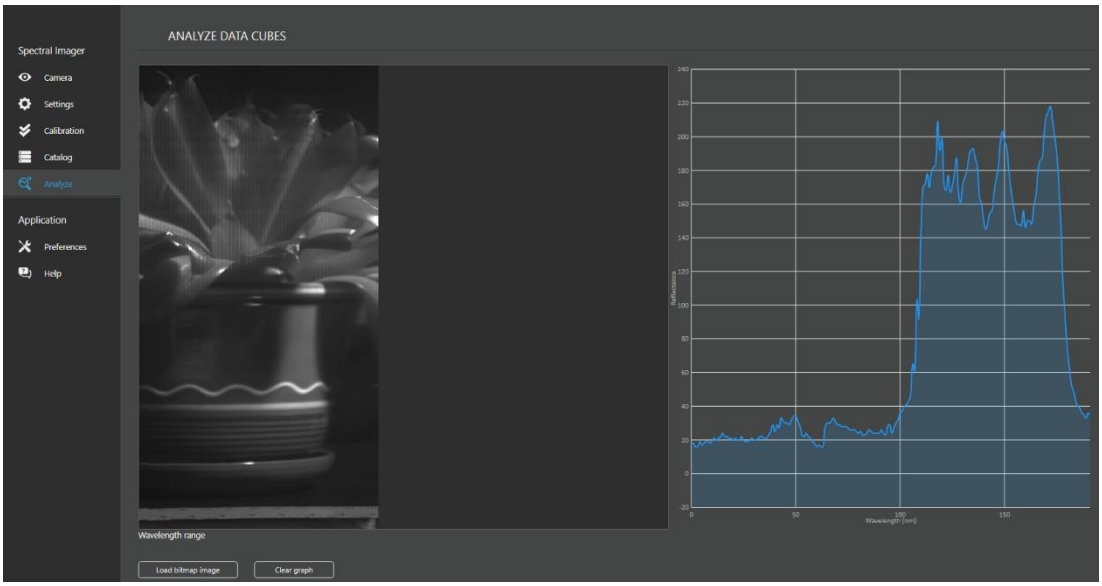

Figure 16 – XimeaCameraCapture interface

2.4 Image processing

The Image data that is delivered by the sensor is processed by multiple components of the data pipeline. The first component is typically the Sensor Data Processor - FPGA. Data is then transported over the communication link (USB, PCIe, Thunderbolt, FireWire) to the computer. In the computer, the image data might be processed by xiApi Image Processing [22] and after that RAW8 data is transferred to the application (Figure 17).

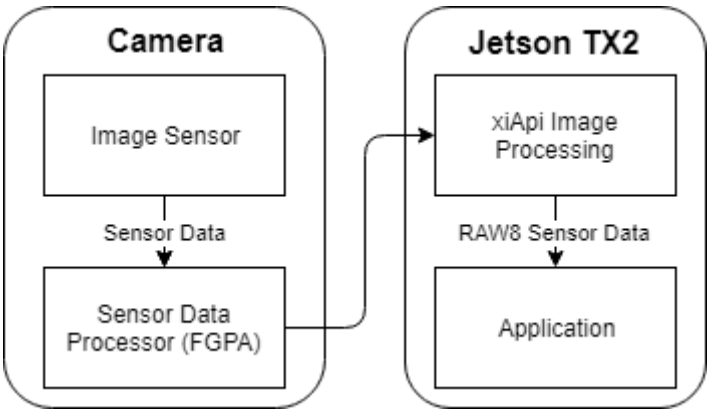

Figure 17 - Image Data Flow

As mentioned earlier, to obtain hyperspectral information, the sensor is covered with special filters, each of which is responsible for a certain light range. However, the filters do not cover the entire surface of the sensor. The area on the sensor covered with filters is called the active area. The positioning of the active area on the sensor is illustrated below in Figure 18.

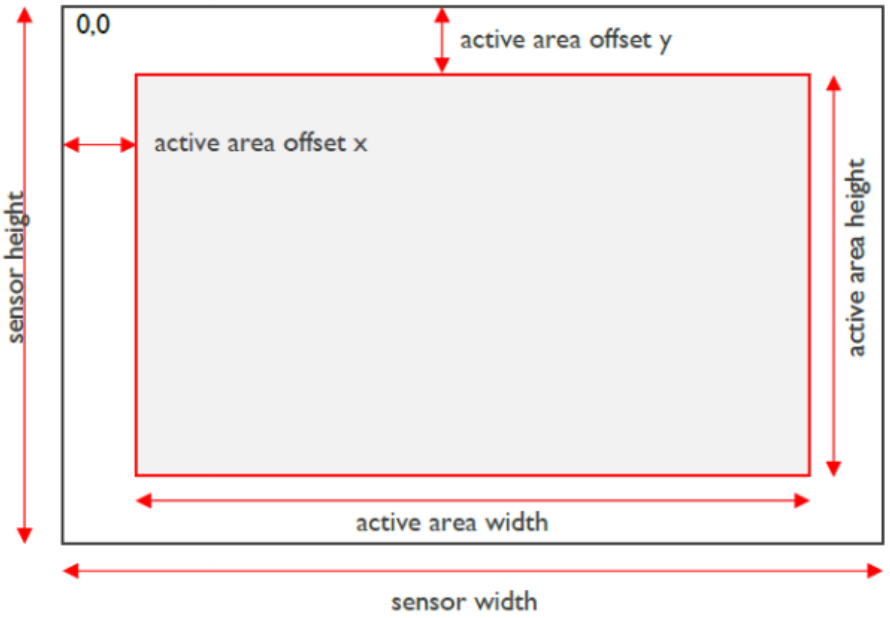

Figure 18 – Active area of a sensor [23]

Another feature of the sensor used is that it is composed of two light-sensitive sensors designed for different ranges. Those filters are distributed over two separate active areas. Within each active area, the filters are organized in a wedge pattern in which each band covers 5 rows.

The entire filter area consists of 320 visible range, 120 interfaces, 640 VIS, and NIR lines.

The active areas are separated from each other by an empty interface zone of 120 rows.

The scheme of the filter is shown in Figure 19, where each line is five pixels high [23].

0	VIS
1	VIS
	...
62	VIS
63	VIS
	empty interface zone
64	NIR
65	NIR
	...
190	NIR
191	NIR

Figure 19 – Empty interface zone [23]

Vertically, the image consists of 1088-pixel bands. By removing the offset of 4 pixels on each side, a 1080-pixel length is obtained. Then, an empty interface zone is removed, located between 63 and 64 spectral lines. The remaining 960 bands are divided into blocks of 5 pixels each These blocks are the 192 active zones from which information is read in a certain light range. Image processing is shown in Figure 20. Red indicates the areas to be removed.

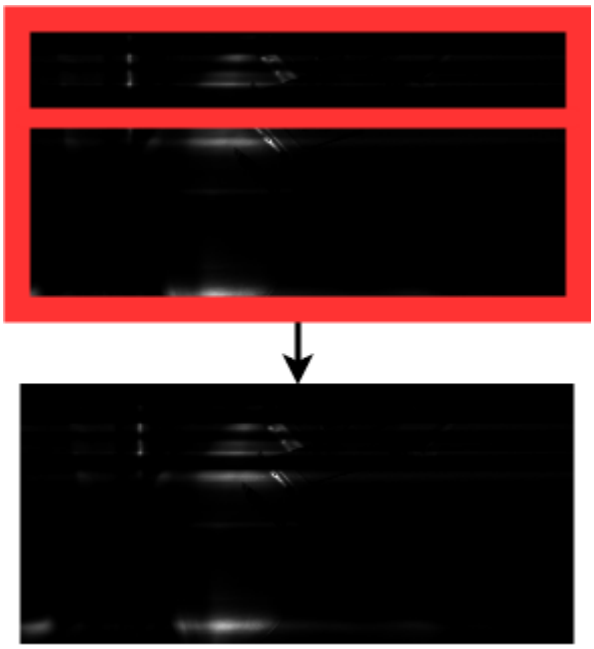

Figure 20 – Image cropping

After removing the offset and empty interface zone, an image was obtained with a resolution of 2040 * 960. After the operations, the image is ready for stitching.

2.5 Image stitching

The line-scan sensor that is used to acquire a hyperspectral representation of an object consists of a conventional sensor with specialized filters processed on top. Each filter only transmits a small portion of the complete spectrum reflected by the scene. In other words, by taking a single image of an object, we get information about the light in only one light range, a width of five pixels.
In order to obtain complete information about the reflection of light on the surface of the entire object, it is necessary to take a series of images and stitch them together. To capture the spectra of every point of an object, every point of an object should pass over each individual band. This can be done by moving the object under the sensor, perpendicular to the orientation of the wedge bands (Figure 21).

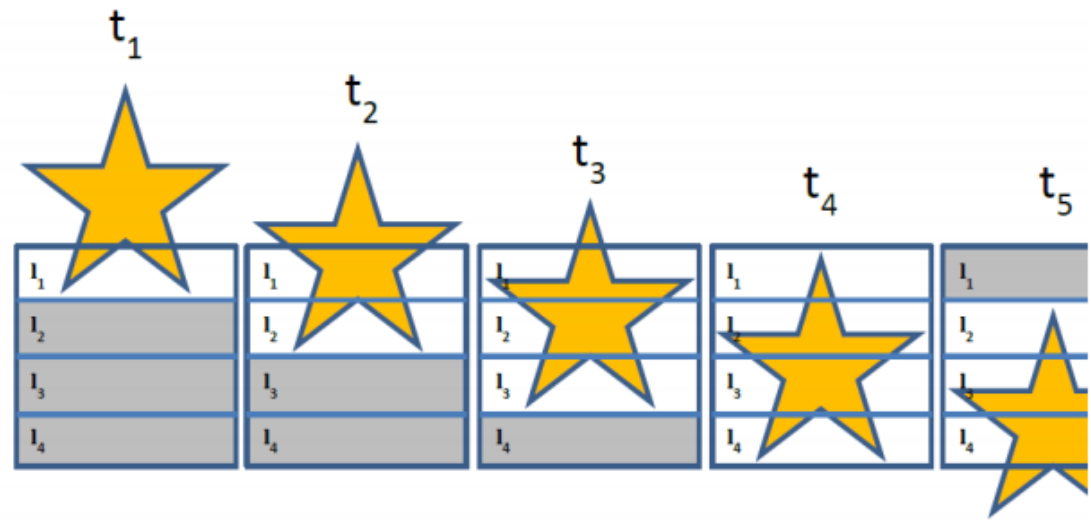

Figure 21 – Object movement [23]

A full scan of an object consists of a start-up phase, a steady state phase, and a shutdown phase. During the start-up and shutdown phase (t1 / t2 / t3 and t5 / t6 / t7, respectively), not all captured data will be used to build the hypercube. This unused data is shown in grey in the Figure above. During steady state (see t4), all captured data is used in the hypercube construction. The number of frames in the steady state depends on the length of the object.

The construction of a hypercube from these wedge images can be performed by reorganizing the individual bands so that all the bands of a certain wavelength, taken in consecutive frames, are stitched together, and this is for each wavelength (Figure 22).

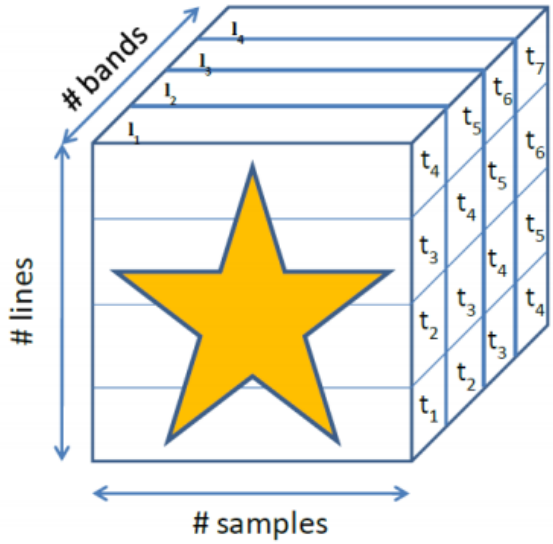

Figure 22 – Hyperspectral cube [23]

In order to effectively stitch the individual ranges of consecutive wedge frames, it is necessary to move the camera perpendicular to the orientation of the wedge bands.

2.6 Neural networks

In recent years, neural networks and machine learning have become firmly entrenched in the information space. Day after day, many researchers and engineers find new ways to apply these methods and achieve unprecedented results. In the task of identifying such complex substances as various oils, they fit perfectly.

Using machine learning, a programmer is not required to write instructions covering all possible problems and containing all solutions. Instead, the computer (or a separate program) lays the algorithm for finding solutions by integrated use of statistical data, from which regularities are derived and based on which predictions are made.

Machine learning technology based on data analysis begins in 1950 [24] when they began to develop the first programs for the game of checkers. Over the past decades, the general principle has not changed. But thanks to the explosive growth of computing power of computers repeatedly complicated patterns and forecasts created by them, and the range of problems and tasks solved using machine learning has expanded.

To start the machine learning process firstly needs to load a Dataset (a certain amount of input data) into a computer, on which the algorithm will learn how to process requests. For example, there may be photos of dogs and cats that already have tags indicating to whom they belong. After the learning process, the program itself will be able to recognize dogs and cats in new images without tags. The learning process continues even after the predictions issued, the more data we have analysed by the program, the more accurately it recognizes the desired images. Using machine learning, computers learn to recognize not only faces in photographs and pictures, but also landscapes, objects, text, and numbers.

One of the categories of machine learning is neural networks. The artificial neural network (ANN) is a mathematical model, as well as its software or hardware implementation, built on the principle of organization and functioning of biological neural networks - networks of nerve cells of a living organism. The neural network allows for efficient clustering, in which the sets of objects will be distributed into predefined categories. Thus, we will be able to classify the images by the substances found on them.

2.7 Investigated sample

During the search for suitable material, two kinds of silicone oil, three kinds of machine oil, sunflower oil, PLS and ABS plastics, metal disc, wood chips, houseplants were investigated. The best results were shown by the test for absorbing chlorophyll in plants. The study of chlorophyll content and was chosen to test the operation of the entire system.

The Figure below (Figure 23) shows all the substances for the detection of which it is reasonable to use hyperspectral imaging.

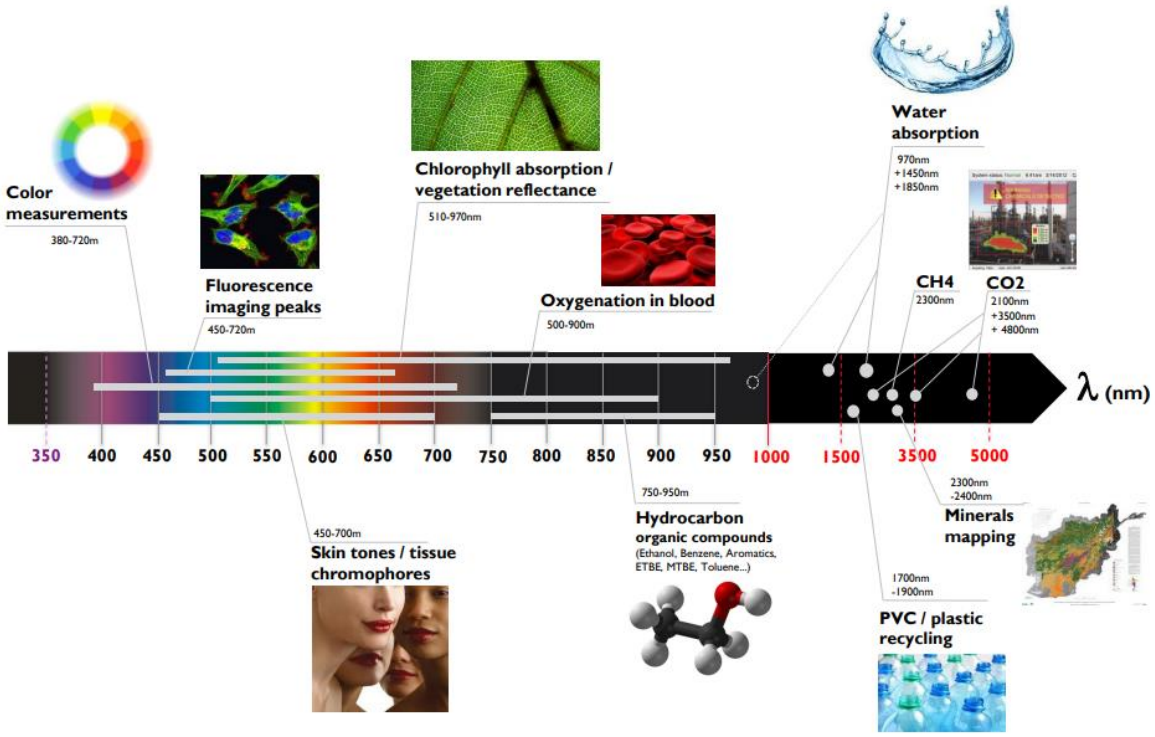

Figure 23 – Substances visible on hyperspectral images [25]

3. DEVELOPMENT OF A SYSTEM

3.1 Jetson TX2 setup

As the very first step, the NVIDIA Jetson TX2 platform should be prepared for further work with it. Jetson TX2 comes without any firmware. At the factory, it is flashed only bootloader to give developers the opportunity to flash firmware themselves. For flashing, TX2-trained Jetson needs a host Linux machine. For this step, a regular computer was used, on which Ubuntu x86_64 Linux version 18.04 was installed. For further installation, the system must meet certain requirements [16]. After installing the operating system on the host computer, jetpack should be downloaded from the NVIDIA website.

Jetpack, which was available at the beginning of the work, supported only the version of Ubuntu 14.04. Since my host machine was installed the version of Linux 18.04, I wrote the following patches to ensure the stability of the Jetpack installer. The code snippet is shown in Figure 24 below.

```
$ Sudo wget -q -O /tmp/libpng12.deb http://mirrors.kernel.org/
$ ubuntu/pool/main/libp/libpng/libpng12-0_1.2.54-ubuntu1_amd64.deb\
$ sudo dpkg -i /tmp/libpng12.deb
$ sudo rm /tmp/libpng12.deb
```

Figure 24 – Patch for Jetpack

The code snippet presented above resolves the libpng12 version of the conflict. Because of this conflict, it was impossible to install Jetpack on Ubuntu 18.04. This code is used in the Linux terminal. JetPack runs on the host Ubuntu x86_64 machine and sets up a development environment and Jetson Development Kit target via remote access[26].

The following components are required to install the jetpack:

1. NVIDIA Jetson TX2
2. Host PC with Linux Ubuntu
3. Display for Jetson TX2
4. HDMI cable
5. mini USB to USB cable
6. USB Keyboard
7. Power supplies

The diagram below (Figure 25) describes a complete cycle of assembly and firmware Jetson TX2, including all the necessary components and dependencies.

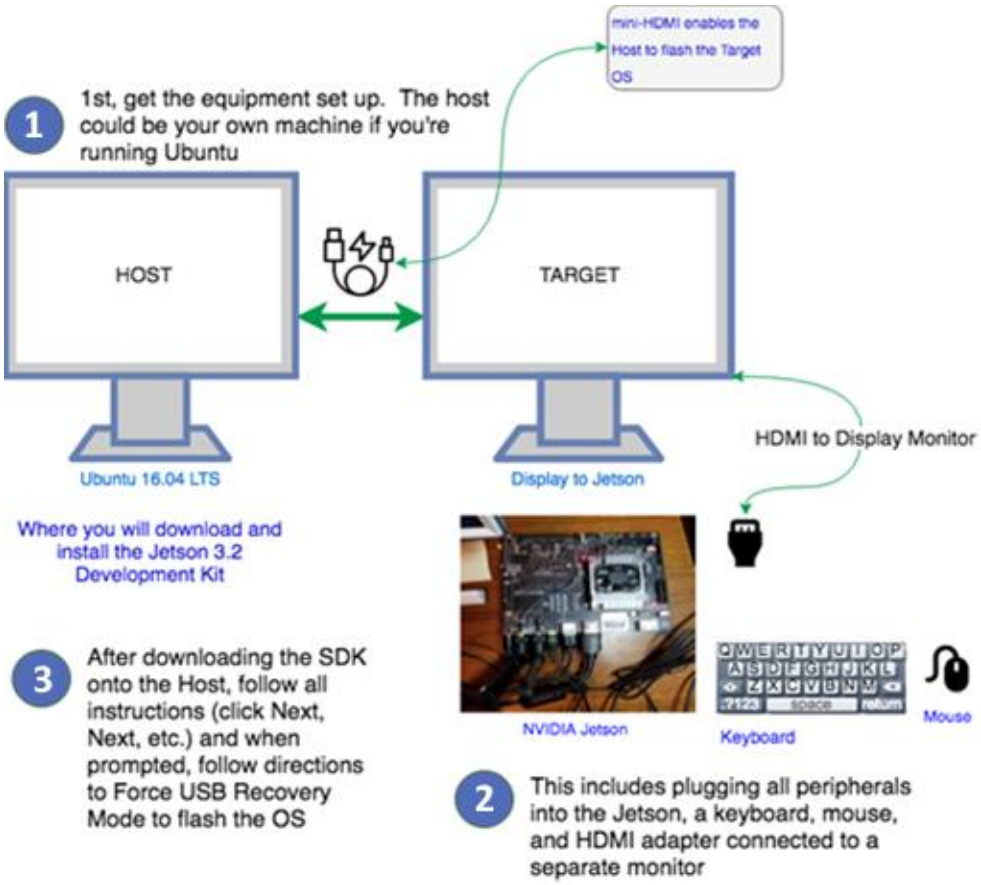

Figure 25 - Jetson TX2 flashing and installation [16]

After connecting all the parts together and completing all the preparations, the terminal should be used to go to the directory where jetpack is downloaded and run the following code (Figure 26):

```
$ chmod +x JetPack-${VERSION}.run
```

Figure 26 - Jetpack install command

This command will start the automatic installation of all required software components. After some time, the device will reboot several times. After that, it will start automatically and be ready to configure the software environment.

3.2 Software environment

3.2.1 Dependences, packages, IDE and fixes

The first step in setting up the software environment was to install the necessary dependencies, which include various libraries, APIs, IDE and configuration files.
As an IDE, ARM package version of code-oss [27] has been installed. The main development language is Python, so its latest, third version (python3) and the package installer (python3-pip) for it have been installed. Git was used as a version control system. HEX editor nomacs [28] is necessary for analysing data received from the camera.

The next step is related to the feature of USB3 in Linux. XIMEA xiQ camera uses USB3 as the primary data channel. Due to this, it turns out very quickly to transfer the images to Jetson and shoot with a high frame rate.
General support for USB 3.0 on Linux is still in a relatively early phase and future releases of Linux kernel and libusb should improve this situation. However, at the moment, for the correct data transfer from the camera to the device memory, it is necessary to increase the size of the USB buffer. The following command was used to increase the buffer size (Figure 27).

```
$ tee /sys/module/usbcore/parameters/usbfs_memory_mb >/dev/null <<<0
```

Figure 27 – Command to increase the buffer size

The default size of USB buffer in Linux is 16 MB, the command above removes this limitation.
The last step is to install the XIMEA Linux Software Package. This package contains USB3.0 drivers for the xiQ camera, xiAPI, interactive viewer CamTool and adapted libraries for Python.

3.3 Hyperspectral imaging

3.3.1 HSI imaging setup

The result of hyperspectral imaging can be greatly distorted by an external light source. The influence of different light sources was discussed in more detail in the previous research work [2]. Based on the experimental data, it was decided to perform hyperspectral imaging in a dark room with a single Incandescent light source. The object is placed on a coating of light-absorbing fabric to minimize the impact of reflected light. Next to the object is a plate with a white surface, which is

used as a reference white to build a colour palette. Before the object is a measuring ruler, which is removed during shooting. The ruler is necessary for measuring the shooting parameters.
A photo of the shooting workspace is shown in Figure 28.

Figure 28 – Object on the light-absorbing surface

A camera with a lens is attached to the base of the stage using a mount printed on a 3D printer (Figure 29). Translation Stage with Stepper Motor THORLABS LTS300/M moves the platform with the camera in a horizontal plane, which allows you to shoot frame by frame with very high accuracy.

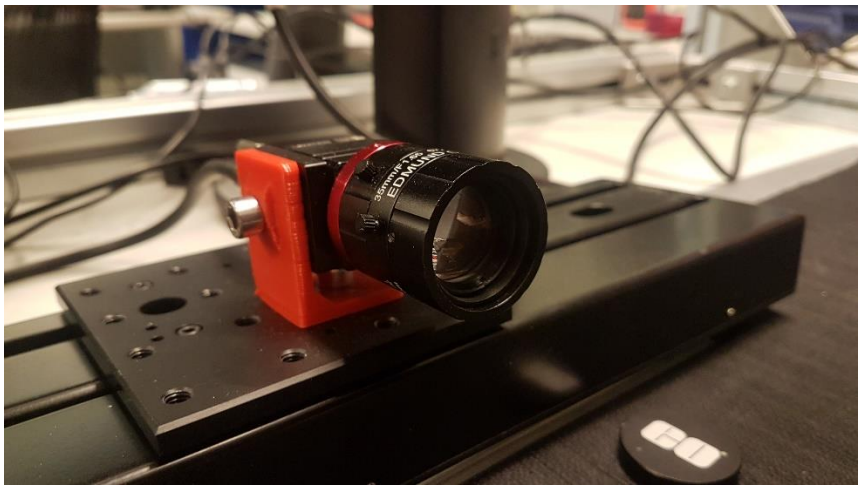

Figure 29 – Camera on the platform

The camera placed in front of the object at a distance of at least 340 mm and moves from right to left relative to the subject. The Incandescent lamp is placed above the surface at a distance of one and a half meters.

The NVIDIA Jetson is connected to the USB hub, Internet cable, HDMI cable, keyboard, mouse, power cable and two antennas (Figure 30).

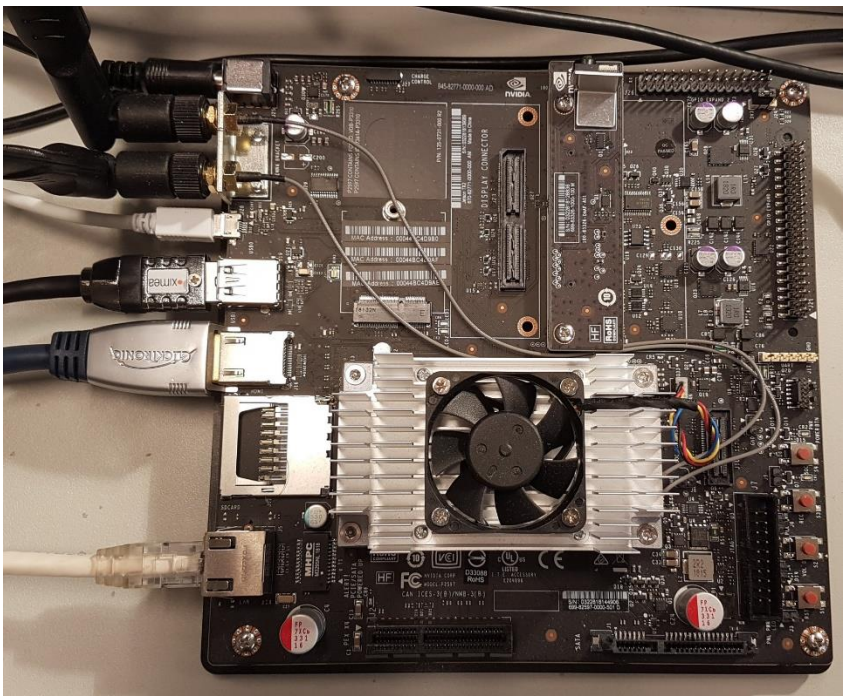

Figure 30 – NVIDIA Jetson TX2 Top View

A fully functional system consists of the following parts:

- NVIDIA Jetson TX2
- XIMEA xiQ MQ022HG with EDMUND OPTICS 35mm C Series VIS-NIR lens
- THORLABS LTS300/M translation stage
- Laptop
- USB hub
- Keyboard
- Mouse
- HDMI Display
- USB – mini USB cable
- USB3.0 cable
- Incandescent lamp
- Mounts and frame
- Power supply

3.3.2 Image acquisition

To create the ideal laboratory conditions for shooting, the following steps can be performed. The room should be isolated from strong outside light sources. The background and surface under the object must be of light-absorbing fabric. The camera should be located in such a way as to capture the entire object when moving from right to left and the distance to the object should be more than 340 mm. Reaching these conditions, it is much easier to get high-quality images.

Hyperspectral camera, which is used in the shooting, has a number of distinctive features that require post-processing of the images. Because of the matrix of the linear type, to create a full hyperspectral image, it is necessary to move the camera step by step relative to the object. Since each image consists of 5-pixel stripes that display different light ranges, all the resulting images must be disassembled into components and stitched together to get 192 images into 192 ranges. Hyperspectral cube is a three-dimensional array created from 192 two-dimensional images obtained by stitching all images of the object. From this cube, you can get a graph of light intensity, shown in Figure 31 (chlorophyll as an example).

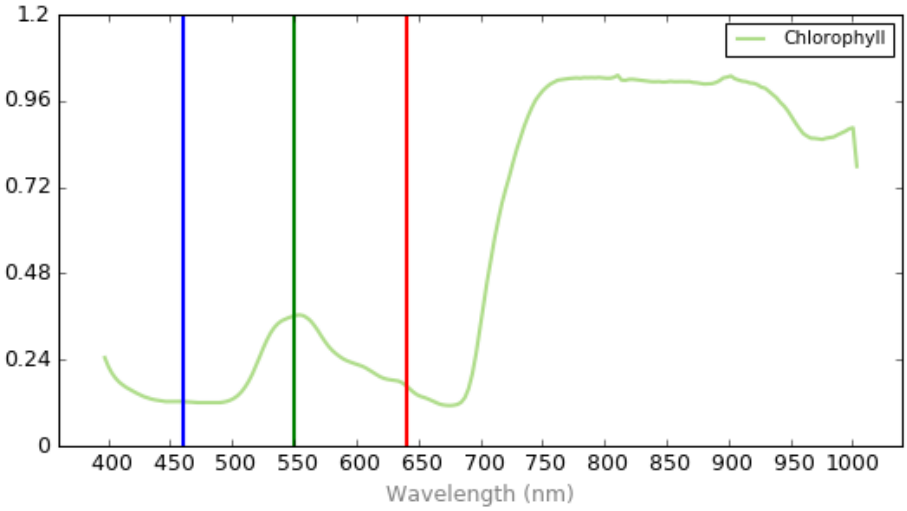

Figure 31 - Chlorophyll

As an alternative to a live plant, below is a graph of an artificial flower made of plastic (Figure 32).

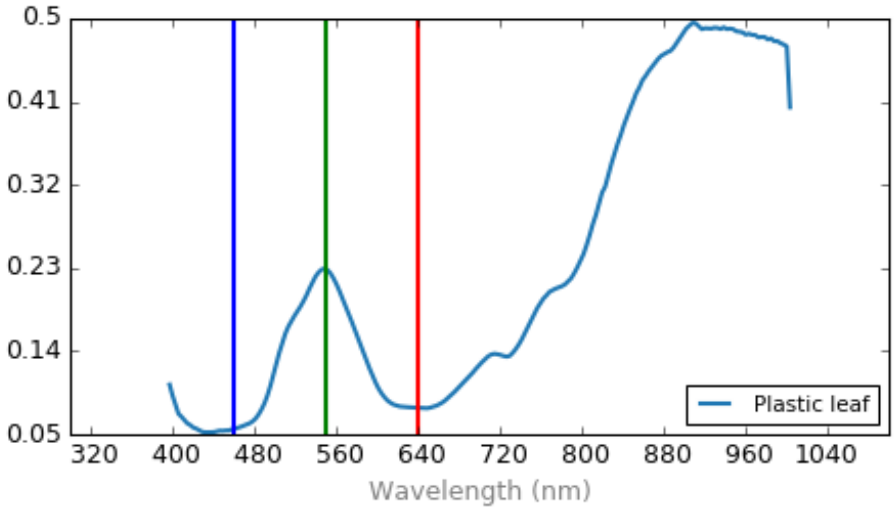

Figure 32 – Plastic leaf

3.4 Capturing software

The developed software works according to the flowchart, which is shown in Figure 33.
After starting the program, a connection to the camera is established. The camera loads the settings specified by the user and starts capturing. Capturing is done in a loop until all images are captured. In the first step of the loop, a snapshot is taken, then the image is processed and loaded into the buffer. Once all images are captured, the connection to the camera is closed. The next step is to create a folder in which all layers of the hyperspectral image will be stored. Then the last stage begins, where the images are saved to the computer. Each image is converted from a one-dimensional to a two-dimensional array, then rotated 90 degrees and stored in the previously created directory. When all images are saved, the program shuts down.

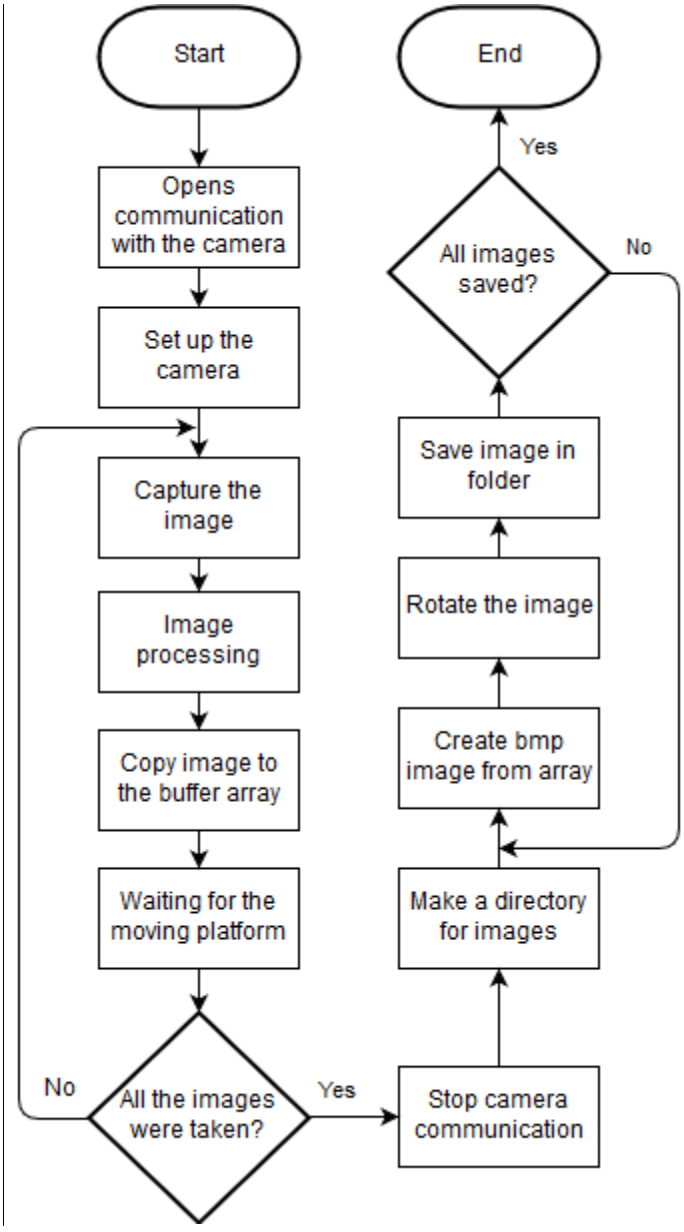

Figure 33 – Software flowchart

3.4.1 Libraries and dependencies

The developed software uses the following libraries (Figure 34):

```
4 from ximea import xiapi
5 from os import mkdir
6 from os.path import expanduser
7 from time import sleep, gmtime, strftime
8 import PIL.Image
9 import numpy as np
```

Figure 34 – Libraries and dependencies

XIAPI is an API provided by XIMEA to interact with the camera used. Using API functions, you can use all available functions of the camera at your discretion. Depending on the level of abstraction, you can both take snapshots, without betraying the value of the data transfer method and file generation, or prescribe the algorithm for processing data obtained directly from the sensor.

The os module provides many functions to work with the operating system, and their behaviour, as a rule, does not depend on the operation system, so the programs remain portable. Mkdir is needed to create a directory, and expanduser to get the path to the home directory.

Time is a module for working with time in Python. The sleep function is used to pause the program for a certain time. Gmtime and strftime are used to get the current system time and convert it to a string.

PIL.Image is a Python library for working with raster graphics. It provides a wide range of tools for working with images, converting, editing, converting, changing parameters and much more.

The NumPy library provides implementations of computational algorithms (in the form of functions and operators) that are optimized to work with multidimensional arrays. As a result, an algorithm that can be expressed as a sequence of operations on arrays (matrices) and implemented using NumPy works as fast as the equivalent code executed in MATLAB.

3.4.2 Camera configuration

The first step in working with the camera is to detect it among the connected devices and establish a direct connection. To do this, you create a cam object of type xiapi. Then the method of this object open_device() is called, which searches for and establishes a connection with the camera (Figure 35).

```
#create instance for first connected camera
cam = xiapi.Camera()
#start communication
cam.open_device()
```

Figure 35 – Camera communication

After establishing a connection with the camera, the following listing appears in the terminal (Figure 36).

```
xiAPI: ---- xiOpenDevice API:V4.17.35.00 started ----
xiAPI: XIMEA Camera API V4.17.35.00
xiAPI: Adding camera context: dwID=51780152  ptr=9D7B7000 processID=00000D4E
xiAPI: Create handles 1 Process 00000D4E
xiAPI: Enable sensor
xiAPI: OK retrains 0
```

Figure 36– Terminal listing for camera opening

The next step is setting the image capturing parameters (Figure 37).

The imgdataformat parameter sets the format of the data sent to the program. The RAW8 data format is a two-dimensional array of size 2048 * 1088. Each element of this array has a value from 0 to 255, reflecting the value of illumination on the corresponding pixel of the matrix.

Buffer_policy is responsible for the way to transfer data from the camera. It can be safe, the data will be copied to the user/application buffer or unsafe, the user will receive an internally allocated buffer without copying the data.

The following two parameters set the gain and shutter speed.

```
#camera settings
cam.set_imgdataformat('XI_RAW8')
cam.set_buffer_policy('XI_BP_SAFE')
cam.set_gain(4)
cam.set_exposure(100000)
```

Figure 37 – Camera settings

If the parameters are successfully written, XIMEA API returns the following answer (Figure 38)

```
xiAPI: AutoSetBandwidth measurement
xiAPI: CalculateResources : Context 9D7B7000 ID 51780152 m_maxBytes=1024 m_maxBufferSize=1048576
xiAPI: Failed to change thread scheduler, check user limit for realtime priority.
xiAPI: AutoSetBandwidth measured 3567Mbps. Safe margin 10% will be used.
xiAPI: Current bandwidth limit auto-set to 3210 Mbps (min:80Mbps,max:3567Mbps)
xiAPI: Sensor SetExposure freq=48MHz exp=16us regexp=x1
xiAPI: Sensor SetExposure freq=48MHz exp=16us regexp=x1
```

Figure 38 – Terminal listing for setting parameters

3.4.3 Parameters

For the correct processing of raw images, it is necessary to determine several basic values in advance. The get_height and get_width functions return the height and width of the image obtained from the matrix.

Variable pixel determines the width of the filter, which is located on the matrix and divides it into zones responsible for different light intensity.

Offset is a variable that adjusts the width of the indent from the edge of the matrix. In the area of this offset, the matrix returns noise instead of specific light values.

The variables described above are represented in Figure 39.

```
22 #basic parameters
23 height  = cam.get_height()
24 width   = cam.get_width()
25 pixel   = 5
26 offset  = 4
```

Figure 39 – Basic parameters

As described earlier, the camera sensor consists of two smaller sensors connected to each other. The first part of the sensor with 64 filters active in the visual range (470-600 nm) and the second part with 128 filters active in the near infrared (600-900 nm). The active areas are separated from each other by an empty interface zone of 120 rows. The empty_interface_zone_start and empty_interface_zone_end variables are responsible for the size of this empty zone (Figure 40).

```
27 empty_interface_zone_start  = 323
28 empty_interface_zone_end    = 443
```

Figure 40 – Empty interface zone

The following two parameters are used to generate dynamic names when storing hyperspectral imaging data (Figure 41).

The variable path is assigned the value of the expanduser function. This function is from the os library, which returns the absolute path to the home directory of the active user. As an argument to the function, a link to the directory with the user's documents is passed. The function uses the obtained argument in such a way that as a result, it returns the absolute path to the documents of the current user.

The second variable is a string containing the current date and time. The value "%d%m%y_%H%M%S" passed to the strftime function, tells it to return a string containing the current day, month, year, hour, minutes, and seconds. Gmtime is a function that returns a structure containing all data about the current time and date.

```
path        = expanduser("~/Documents")
dir_name    = strftime("%d%m%y_%H%M%S", gmtime())
```
Figure 41 - Paths

3.4.4 Calculations

The lines variable is calculated based on the height of the image from which the offsets and the empty interface area are subtracted. In our case, the final image will consist of 960 lines. These 960-pixel lines include 320 lines in the visible range and 640 lines in the near infrared range.

The number of light ranges contains the bands variable. Each light range consists of five lines and has a width of five pixels. The value of the variable is obtained by dividing the total number of pixel lines by the width of the light range. These variables are described in Figure 42 below

```
lines   = height – 2 * offset – (empty_interface_zone_end -
empty_interface_zone_start) #960
bands   = int(lines / pixel) # 192
```
Figure 42 – Lines and bands

The object_width and working_distance parameters are set manually. The first variable stores the value of the shooting object width, and the second distance to it (Figure 43).

The line_height variable shows the line width of five pixels in millimetres. Calculated as the height of the sensor divided by the width of a pixel and multiplied by the number of pixels in one spectral range.

The last variable is the focal length of the lens.

```
object_width        = 50
working_distance    = 400 #Distance to object
line_height         = 0.0275 # (Sensor height / ROW pixel) * pixel per
line
focal_length        = 35 # Focal length in mm
```
Figure 43 – Object parameters

For the correct shooting of the object, its resolution must be calculated. It is calculated based on the parameters defined above and is the distance to the object multiplied by the line height and divided by the focal length (Figure 44).

```
object_resolution    = (working_distance * line_height) / focal_length
```
Figure 44 – Object resolution calculation

Variable frames (Figure 45) - shows how many images should be taken to get a hyperspectral image of the object of the declared width.

image_buffer is a numpy array that stores all captured images.

This array is three-dimensional (Figure 45), where the first level consists of 192 elements reflecting the total number of light ranges, the second level contains the number of images (5-pixel bands) taken in this light range, and the last level has a length of 2044 characters and contains the data of these images. The array is of type uint8, which means that each character contains a number from 0 to 255.

```
46 frames = int((object_width / object_resolution  + ((bands - 1) * 2)))
47 image_buffer = np.zeros((bands, frames * pixel, width - offset * 2),
dtype = np.uint8)
```

Figure 45 – Frames and image buffer

The last step before capturing is to create an instance of Image to store image data and metadata and then set camera into acquisition mode (Figure 46). Img is used as such variable, to which the built-in type xiapi.Image used. After the declaring of all variables, the camera receives a command to switch to acquisition mode.

```
50 img = xiapi.Image()
51 cam.start_acquisition()
```

Figure 46 – instance for image data and acquisition start

3.4.5 The hyperspectral image capturing and stitching

All the functions described below are executed in a loop, the number of iterations of which is equal to the number of frames to be captured.

Shooting starts from the starting position, which is determined by the configuration of the travel stage.

The first step of each iteration is to get the image from the camera (Figure 47). The Cam object calls the get_image() method, which overwrites the img variable, assigning it the value of a new array with imaging data. This array is overwritten every iteration of the loop.

```
54 for current_frame in range(frames):
55     #stage in start position
56     print(current_frame + 1, "step of", frames)
57     cam.get_image(img) #Image capture
```

Figure 47 – Main loop and image capturing

Once the picture is taken from the camera, it will be processed. The get_image_data_numpy function converts the received data to a numpy array. It is a fast built-in xiapi function that allows to easily convert an array from raw data to a numpy array (Figure 48).

```
60 data = img.get_image_data_numpy() #Get RAW8 data as numpy array
```

Figure 48 – Get numpy array

After conversion, the array data is of type numpy, which allows to convert it differently. At this step, using the mask, the offsets will be removed. On each side of the image, 4 pixels will be removed. As a result, the image will change its resolution from 2048*1088 to 2040*1080 (Figure 49).

```
61 data = data [offset : height - offset, offset : width - offset] #Cut the edges
```

Figure 49 – Cutting the edges

After removal of the indentation, it is necessary to remove the noisy region within the image. The empty interface area is 120 pixels wide and is removed using the numpy's built-in delete function (Figure 50).

```
62 data = np.delete(data, np.s_[empty_interface_zone_start - offset : empty_interface_zone_end - offset], 0 ) #Cut the sensor connection (empty interface zone)
```

Figure 50 – Cut the empty interface zone

After image pre-processing is completed, a set of 192 spectral ranges is saved to the buffer. However, it is not enough just to consistently save images to the buffer. As described earlier, each subsequent image is shot with a shift of 5 pixels relative to the previous one. In accordance with this, at each iteration of the cycle, it is necessary to place a range of 5 pixels in accordance with its real location in the hypercube (Figure 51).

```
64      #transfer bands into the buffer
65     for i in range(bands):
66          for j in range(pixel):
67              image_buffer[i][current_frame * 5 + pixel - 1] = data[i * pixel + j][:]
```

Figure 51 – Buffer filling

The documentation and the translation stage itself were examined, and it was found that it did not have Linux support [10]. Based on the fact that the device does not support the operating system used and will not be used in further experiments, it was decided to manage travel stage using the official Windows application from the laptop. Shooting and moving the camera are synchronized at time intervals. For this reason, instead of directly accessing the travel stage API, the program pauses, waiting for the platform to move with the camera initiated from outside (Figure 52).

```
71     time.sleep(.65)
```

Figure 52 – Standby function

After the timer ends, the terminal displays information about the completed stage and the cycle moves to the next iteration.

3.4.6 Camera connection close and Image saving

After all the necessary frames have been shot in the loop, a command is sent to the camera to finish reading the data from the matrix and close the data channel (Figure 53).

```
73 #stop communication
74 cam.stop_acquisition()
75 cam.close_device()
```

Figure 53 – Stop acquisition and close the device

To save the received images, a new directory is created in the user's documents. The name of this directory consists of adding the path and dir_name lines, which, in turn, is the current date and time (Figure 54).

```
81 mkdir('{}/{}'.format(path, dir_name))
```

Figure 54 – Directory creation

In a cycle of 192 iterations, images are saved, composed of all captured frames.

The fromarray function generates an image that is assigned to the img variable. The first argument of the function is the array from which the image will be built. The second argument of the function defines the type and depth of a pixel in the image ('L' means (8-bit pixels, black and white).

Since the camera is rotated 90 degrees relative to the subject, the resulting image must be expanded by the rotate function.

The save function belongs to the PIL class.Image and allows to save the image. The argument is a string that contains the path to the file to be created. The path to the file is taken from the path variable and added to it the name of the directory, the file name from 0 to 191 and the extension ".bmp" (Figure 55).

```
84 for i in range(bands):
85     img = PIL.Image.fromarray(image_buffer[i, :, :(frames - bands)], 'L')
86     img.rotate(90)
87     img.save(path + '/{}/{}.bmp'.format(dir_name,i))
```

Figure 55 – Creation of 192 BMP files

Once all the images are saved, we get a new directory in the document folder that contains 192 images with the BMP extension.

3.5 Neural network

3.5.1 Dataset

The first step in learning a neural network is to prepare data for training. As a task for neural network training, image classification by chlorophyll content was chosen.
The task for the neural network will be to classify plant photos into real and artificial ones based on chlorophyll data obtained from the hyperspectral image.

Hyperspectral images for the dataset were obtained under different environmental conditions. In laboratory conditions with one light source, with two or more artificial light sources and with natural light. Also, the values of photosensitivity and exposure of the sensor changed constantly.

As a result, a set of 200 hyperspectral images was obtained, in which artificial and natural plants are illuminated from different sides and different sources. The data set turned out to be quite diverse so that the trained neural network was ready to work with a large set of initial data (different position, lighting, distance, and so on).

150 of these images will be used for training and 50 for testing the accuracy of the neural network. The sample is not too large, but acceptable accuracy can be achieved with this dataset. Also, the purpose of training of this neural network is to demonstrate the capabilities of the developed algorithm and the operation of the entire system from the moment of image acquisition to decision-making based on the processed data.
Because chlorophyll is actively reflecting light only in certain light bands (Figure 56), prior to the preparation of dataset of the images, they must be pre-sorted.

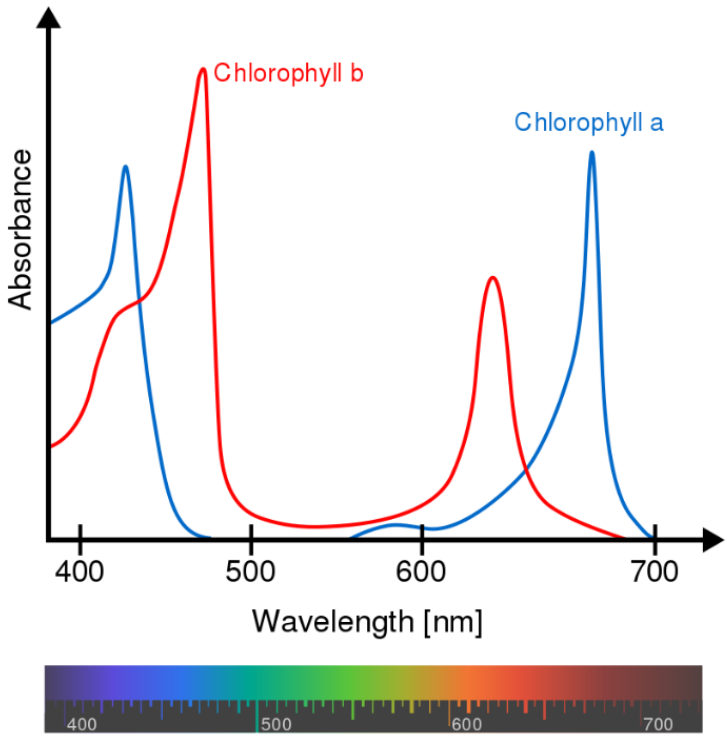

Figure 56 – Absorption spectra of chlorophylls a and b [29]

According to the specification of the camera used, the luminous flux is distributed over 150 light ranges in the manner shown in Table 1 in the appendix section.

According to the spectral distribution of chlorophyll and the method of positioning the spectral ranges of the image in the range from 500 to 600 nm and after 700 nm can be excluded from the dataset. On Figure 57 sample images from the dataset are shown. On the left side of the image of the flower, and the right artificial.

Figure 57 - Image samples

3.5.2 Dataset generation

To create the dataset, the pillow and numpy libraries were used, which were described in the previous section. To generate a dataset, all layers of the hyperspectral image are combined into a single image, which will be transmitted to the neural network input. Figure 58 shows the code fragment that is responsible for stitching two images. The function measures the height and width of images and on their basis creates a new image. Then, two source images are sequentially copied onto the blank image. The result is the first and second images, connected horizontally. The function returns the stitched image.

```
9   def image_stitching(base_image, connected_image):
10
11    (width, height) = base_image.size
12    (width_con, height_con) = connected_image.size
13
14    result_width = width + width_con
15    result_height = max(height, height_con)
16
17    result = Image.new('L', (result_width, result_height))
18    result.paste(im=base_image, box=(0, 0))
19    result.paste(im=connected_image, box=(width, 0))
20    return result
```

Figure 58 – Image stitching function

The function described above will be used for each image of which the hyperspectral image consists. The next step is to stitch the images and form a set of different images. In order to properly compose a set of images, a set of variables must be defined.

The bands variable has a value of 192 and specifies the maximum number of spectral levels that the hyperspectral image will consist of.

As mentioned earlier, to determine chlorophyll and reduce the volume of the original images, the dataset will be trained on the cropped spectral images.

In order to take data only for certain spectral ranges, the variable first_segment contains the length of the first range (starting from zero), and the variables second_sigment_start and second_sigment_end contain the coordinates of the beginning and end of the second range.

The sample variable contains the number of hyperspectral images from which the dataset will be composed. The images list stores the stitched image, and script_dir contains the absolute address to the folder from which the script is executed.

The variables described above are shown in Figure 59.

```
22 bands   = 192
23 first_segment   = 15 # 0 - 15
24 second_segment_start = 35
25 second_segment_end   = 75
26 second_segment_length = second_segment_end - second_segment_start #35->75
27 samples = 200
28 images = [samples]
29 script_dir = os.path.dirname(__file__)
30 current_image = Image.open(script_dir + "\\" + str(samples) + "\\" +
str(0) + ".bmp")
```

Figure 59 - Variables

The next step is to stitch the images. Figure 60 shows the cycle in which hyperspectral images are loaded and processed. For each image, data from previously defined spectral ranges is taken and stitched into a single image, which is stored in the images variable. Then, when all parts of the image are stitched, it is converted to a one-dimensional array and stored in the image array.

```
32 for im in range(samples):
33     for i in range(1,first_segment):
34         band = Image.open(script_dir + "\\" + str(im) + "\\" + str(i) +
".bmp")
35         current_image  = image_stitching(current_image, band)
36
37     for i in range(second_segment_start, second_segment_end):
38         band = Image.open(script_dir + "\\" + str(im) + "\\" + str(i) +
".bmp")
39        current_image  = image_stitching(current_image, band)
40
```

```
41    current_image = np.array(current_image) #Make array
42    current_image = current_image.ravel() #Make array flat
43    images.append(current_image)
```

Figure 60 – Image stitching cycle

As a result of the operations, we get a set of processed images. These images should then be marked. Since the output of the neural network has only two classes of images (with or without chlorophyll), each image is assigned a value of 0 or 1. Where 0 is the absence of chlorophyll and 1 is its presence. Since the dataset consists of only 200 images, their marking was carried out manually. As a result of this work, there are 200 processed images and an array of zeros and ones, classifying these images.

3.5.3 Neural Network training and tests

The trained Neural network must perform the task of binary classification. To work with a neural network, Tensorflow library and keras module will be used. The first step is to prepare two input vector arrays. The first vector is called images and contains vectorized images. The second vector is called targets and contains ones and zeros that indicate that the image belongs to a certain class.
These arrays should be divided into data that will be used in the training process and data that will be used to test the learning outcomes. Figure 61 shows the division of the dataset into training and testing data.

```
48 train_data      = images[150:]
49 train_targets   = targets[150:]
50 test_data       = images[:150]
51 test_targets    = targets[:150]
```

Figure 61 – Dataset dividing

The next step is to configure the NN. The basic Keras data structure is a model, a way of organizing layers. The main type of model is a sequential model, a linear stack of layers.
Input, output, and hidden layers for NN are defined below in Figure 62. As an activation function, the ReLu function is used [30]. The Dense function is used to connect layers to each other, and the Dropout function prevents overtraining [31].

```
53 model = models.Sequential()
54 # Input level
55 model.add(layers.Dense(25, activation = "relu",
input_shape=(width*height, )))
56 # Hidden - Layers
57 model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
58 model.add(layers.Dense(25, activation = "relu"))
59 model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
60 model.add(layers.Dense(25, activation = "relu"))
61 # Output- Layer
62 model.add(layers.Dense(1, activation = "sigmoid"))
```

Figure 62 – NN inner structure

After defining the neural network model, it must be compiled. The adam optimizer was used for compilation, and binary cross-entropy is used as a loss function. Figure 63 shows compilation of the model.

```
model.compile(
 optimizer = "adam",
 loss = "binary_crossentropy",
metrics = ["accuracy"]
)
```

Figure 63 – Model compile

When a neural network model is defined and compiled, it can be trained using a prepared dataset. The model is trained with batch sizes in 32 and two epochs. During the training of the model, it was found that the accuracy of the model decreases when establishing more epochs. Source code for model training is showed below in Figure 64.

```
results = model.fit(
 train_data, train_targets,
 epochs= 2,
 batch_size = 32,
 validation_data = (test_data, test_targets)
)
```

Figure 64 – Model training

After the model is successfully trained, it is necessary to evaluate its effectiveness. For this purpose, a function is used that shows the accuracy with which the NN was determined by the images from the test part of the dataset (Figure 65).

```
print(np.mean(results.history["val_acc"]))
```

Figure 65 – Accuracy evaluation

As a result of this work, the neural network was trained, which showed an efficiency of 96%. This high accuracy is due to the presence of easily distinguishable signatures of plants that have been analyzed, as well as the fact that the dataset was prepared in the laboratory.

4. SUMMARY

This thesis demonstrated in detail the development of a method for acquisition of a hyperspectral image using a single NVIDIA Jetson TX2 Board computer. To test the developed method, the neural network was trained. As initial data for neural network training, the images obtained as a result of the developed method were used. The neural network has been trained to classify real and artificial plants on hyperspectral images.

Based on the obtained results and developed method efficiency, it was proven that all components of the system were correctly configured and assembled together, and the program code correctly performs the task.

4.1 Future work

In this paper, the developed method has been tested in the laboratory. The next step is to test it in tasks that have practical application. The camera, together with a single Board computer, can be placed on a stand-alone complex (robot or UAV) and tested in real conditions.

The second direction for the development will be the integration of another moving platform for the camera. The platform on which the development was carried out has two serious limitations. The first is that it is too bulky to be placed on an aircraft, and the second is that it does not support Linux and its use will require the development of additional software components.

The trained neural network implements the binary classification of images. Neural networks have great potential in the analysis and classification of complex multilayer hyperspectral images. The developed method of obtaining hyperspectral images can be used to create a more diverse dataset from a larger number of examples and on its basis to train the neural network to identify substances in the images or to classify them.

5. LIST OF REFERENCES

[1] H. Grahn, P. Geladi (2007), “Techniques and applications of hyperspectral image analysis”. John Wiley & Sons.

[2] A. Palshin, A. Zahavi, D. Chamara, M. Tamre. “Influence of Illumination Sources on Hyperspectral Imaging”. 20th International Conference on Research and Education in Mechatronics (REM 2019). This paper will be published in June.

[3] ‘Hyperspectral Imaging technology’. ScienceDirect, [Online]. Available: https://www.sciencedirect.com/topics/medicine-and-dentistry/hyperspectral-imaging

[4] ‘Medical hyperspectral imaging: a review’. SPIE Digital Library, [Online]. Available: https://www.spiedigitallibrary.org/journals/Journal-of-Biomedical-Optics/volume-19/issue-01/010901/Medical-hyperspectral-imaging-a-review/10.1117/1.JBO.19.1.010901.full?SSO=1

[5] ‘Hyperspectral Imaging’. XIMEA official website, [Online]. Available: https://www.ximea.com/support/attachments/4675/XIMEA_imec_HSI_technology-Part-V1.1.pdf

[6] ‘Hyperspectral Imaging’. HySpex company website, [Online]. Available: https://www.hyspex.no/hyperspectral_imaging/

[7] ‘Advantages for Applications using opportunities of Hyperspectral Imaging’. XIMEA official website, [Online]. Available: https://www.ximea.com/usb3-vision-camera/hyperspectral-usb3-cameras

[8] ‘Jetson TX2 Module’. NVIDIA official website, [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx2

[9] ‘xiQ - USB3 Vision Cameras’. XIMEA official website, [Online]. Available: https://www.ximea.com/usb3-vision-camera

[10] ‘300 mm Linear Translation Stage with Integrated Controller, Stepper Motor’. THORLABS company website, [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=7652&pn=LTS300

[11] ‘Harness AI at the Edge with the Jetson TX2 Developer Kit’. NVIDIA official website, [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx2-devkit

[12] ‘Jetson TX2 Series Data Sheet’. Jetson Download Centre, [Online]. Available: http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-modules-data-sheet

[13] ‘MQ022HG-IM-LS150-VISNIR’. XIMEA datasheet and technical information, [Online]. Available: https://www.ximea.com/en/products/hyperspectral-cameras-based-on-usb3-xispec/mq022hg-im-ls150-visnir

[14] ‘35mm C Series VIS-NIR Fixed Focal Length Lens’. Edmund optics website, [Online]. Available: https://www.edmundoptics.com/p/35mm-c-series-vis-nir-fixed-focal-length-lens/22384/

[15] ‘Hyperspectral imaging, data correction and standardization, mobile applications’. XIMEA GmbH, Jürgen Hillmann, [Online]. Available: https://www.ximea.com/support/attachments/5981/SpectroNet-2016-03-Ximea-V02.pdf

[16] ‘Jetson Development Pack for Linux for Tegra (JetPack L4T)’. NVIDIA documentation website, [Online]. Available: https://docs.nvidia.com/jetpack-l4t/2_1/content/developertools/mobile/jetpack/jetpack_l4t/2.0/jetpack_l4t_main.htm#System_Requirements

[17] Annala L., Eskelinen M. A., Hämäläinen J., Riihinen A., Pölönen I. (2018) ‘Practical Approach for Hyperspectral Image Processing in Python’. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-3, 2018.

[18] Numpy official website, [Online]. Available: https://www.numpy.org/

[19] ‘TensorFlow official site’. [Online]. Available: https://www.tensorflow.org/

[20] ‘Keras Documentation’. [Online]. Available: https://keras.io/.

[21] ‘Dhanushka Chamara Liyanage personal page’. [Online]. Available: www.dhanushkaliyanage.com

[22] ‘xiAPI Image Data Flow’. [Online]. Available: https://www.ximea.com/support/wiki/apis/

[23] ‘xiSpec linescan cameras sensor info and data processing’. [Online]. Available: https://www.ximea.com/files/public/xiSpec-linescan-cameras-V01.pdf

[24] A. L. Samuel (1959) ‘Some studies in machine learning using the game of checkers’. IBM Journal of Research and Development, Volume: 44, Issue: 1.2.

[25] Bruno Aiazzi, Luciano Alparone, Stefano Baronti, Cinzia Lastri. (2012) ‘Spectral Distortion in Lossy Compression of Hyperspectral Data’. Journal of Electrical and Computer Engineering 2012, Volume 5.

[26] ‘Download and Install JetPack L4T’. Jetson Download Centre, [Online]. Available: https://docs.nvidia.com/jetpack-l4t/2_1/content/developertools/mobile/jetpack/jetpack_l4t/2.0/jetpack_l4t_install.htm

[27] ‘Community builds of Visual Studio Code for ARM’. [Online]. Available: https://code.headmelted.com

[28] ‘Nomacs home page’. [Online]. Available: https://nomacs.org/

[29] ‘Light and photosynthetic pigments’. Khan Academy. [Online]. Available: https://www.khanacademy.org/science/biology/photosynthesis-in-plants/the-light-dependent-reactions-of-photosynthesis/a/light-and-photosynthetic-pigments

[30] ‘Keras Activations. Usage of activations’. [Online]. Available: https://keras.io/activations/

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’. J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014.

[32] ‘Adam — latest trends in deep learning optimization’. [Online]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-

6. APPENDICES

Table 1 spectral bands distribution

№ of band	nm
1	470.00
2	472.87
3	475.73
4	478.60
5	481.47
6	484.33
7	487.20
8	490.07
9	492.93
10	495.80
11	498.67
12	501.53
13	504.40
14	507.27
15	510.13
16	513.00
17	515.87
18	518.73
19	521.60
20	524.47
21	527.33
22	530.20
23	533.07
24	535.93
25	538.80
26	541.67
27	544.53
28	547.40
29	550.27
30	553.13
31	556.00
32	558.87
33	561.73
34	564.60
35	567.47
36	570.33
37	573.20
38	576.07

39	578.93
40	581.80
41	584.67
42	587.53
43	590.40
44	593.27
45	596.13
46	599.00
47	601.87
48	604.73
49	607.60
50	610.47
51	613.34
52	616.20
53	619.07
54	621.94
55	624.80
56	627.67
57	630.54
58	633.40
59	636.27
60	639.14
61	642.00
62	644.87
63	647.74
64	650.60
65	653.47
66	656.34
67	659.20
68	662.07
69	664.94
70	667.80
71	670.67
72	673.54
73	676.40
74	679.27
75	682.14
76	685.00
77	687.87
78	690.74
79	693.60
80	696.47
81	699.34
82	702.20
83	705.07

84	707.94
85	710.80
86	713.67
87	716.54
88	719.40
89	722.27
90	725.14
91	728.00
92	730.87
93	733.74
94	736.60
95	739.47
96	742.34
97	745.20
98	748.07
99	750.94
100	753.80
101	756.67
102	759.54
103	762.40
104	765.27
105	768.14
106	771.00
107	773.87
108	776.74
109	779.60
110	782.47
111	785.34
112	788.20
113	791.07
114	793.94
115	796.80
116	799.67
117	802.54
118	805.40
119	808.27
120	811.14
121	814.00
122	816.87
123	819.74
124	822.60
125	825.47
126	828.34
127	831.20
128	834.07

129	836.94
130	839.80
131	842.67
132	845.54
133	848.40
134	851.27
135	854.14
136	857.00
137	859.87
138	862.74
139	865.60
140	868.47
141	871.34
142	874.20
143	877.07
144	879.94
145	882.80
146	885.67
147	888.54
148	891.40
149	894.27
150	897.14
	900.00