Automatic tree detection and species identification by fusion of laser scanner data and camera images

Jimmy Forsman (jifo0004@student.umu.se)
Viktor Wiberg (viwi0019@student.umu.se)
Carl Holmberg (caho0060@student.umu.se)
Karl-Johan Lundström (kalu0009@student.umu.se)
Sofi Backman (soba0030@student.umu.se)
Ludvig Renström (lure0007@student.umu.se)

January 25, 2018

Design-Build-Test, 15 credits
Tutors: Johan Holmgren,
Mattias Nyström,
Kenneth Olofsson &
Jonas Bohlin

Abstract

A recent master thesis project conducted on behalf of Sveriges lantbruksuniversitet (SLU) developed a backpack-based mobile laser scanning (MLS) system. This system was capable of generating georeferenced 3D-point clouds of the surroundings with promising accuracy.

This project concerns a method which extends the MLS system by integrating a camera. The aim was to determine the species of each georeferenced tree through bark images. For this task we use a machine learning approach of convolutional neural networks (CNN).

The method is to project laser data of each individual tree to the camera images depicting the trees. Then, extract the tree from the picture and classify its species. We require that the camera images are synchronized with the rest of the system in both space and time. The time synchronization uses the camera hot shoe to read off when a picture was taken in order to time stamp that image. The position and orientation of the camera for each image is found through its spatial relation to the navigation system at the time stamp.

Three tree species can be detected by the neural network: Norway spruce, Scots Pine and birch. Testing our re-trained CNN on manually taken photos resulted in correctly classifying 95% of the images.

When testing the MLS with the camera it showed that in most cases, the laser data mapped to the pictures showed sufficient accuracy as to extracting the part of the images containing trees.

The code for automatic extraction of tree stems from images, which is then partitioned into bark segments, is not yet completely developed. What then remains is to run experimentally obtained bark image segments through our modified CNN and examine the classification results. Completing these tasks one could obtain an end-to-end tree georeferencing and species identification tool.

Contents

1 Introduction 1

1.1 Purpose and goal . . . 1

2 Conceptual description of the system design 3

2.1 Pre-existing system setup . . . 3

2.1.1 Light detection and ranging . . . 3

2.1.2 Global navigation satellite system . . . 3

2.1.3 Inertial navigation system . . . 4

2.1.4 Operating computer . . . 4

2.2 Conceptual setup and method of the system extension . . . 5

2.2.1 Spatial calibration of a camera image and 3D data points . 5

2.2.2 Automation of the image taking process . . . 5

2.2.3 Time synchronization of the camera and the rest of the system 5

2.2.4 Software for tree classification . . . 5

3 Method 7

3.1 Instruments and system mount . . . 7

3.2 Time synchronization of camera and INS . . . 9

3.2.1 Camera triggering . . . 10

3.3 Extrinsic calibration . . . 10

3.3.1 Object 3D-points in camera frame . . . 11

3.3.2 Object 3D-points in the LiDAR frame . . . 13

3.3.3 Rigid body transformation . . . 14

3.4 Tree classification using convolutional neural networks . . . 15

3.4.1 Transfer learning . . . 16

3.4.2 Data acquisition and two classification problems . . . 16

3.5 Post processing . . . 17

3.5.1 Dynamic calibration tools . . . 18

3.5.2 Mapping laser data to pictures . . . 18

4 Results 19

4.1 Results from time synchronization . . . 19

4.2 Results from extrinsic calibration . . . 20

4.3 Survey results . . . 21

4.4 Results from CNN tree classification . . . 23

5 Discussion 24

5.1 Ricoh GR camera . . . 24

5.2 Extrinsic calibration . . . 24

5.3 CNN tree classification 25
5.4 Post processing 25

6 Conclusions and outlook 26

7 References 27

A Extrinsic calibration parameters A.i

B User guide for the MLS system B.i

1 Introduction

Until recently, the use of laser scanning for forestry has been limited to airborne (ALS) and terrestrial laser scanning (TLS) systems. The last few years, however, the methods of ground-based mobile laser scanning (MLS) systems have started to reach the robustness and accuracy required for forestry purposes. The reason why MLS for forestry has been slow in coming are the problem of georeferencing 3D point clouds to world coordinates in environments where positioning using only a global navigational satellite system (GNSS) is insufficient.

This paper covers the extension of an existing backpack-based MLS system, developed by Mattias Tjernqvist as part of his master thesis project [1]. The pre-existing system showed promising results in georeferencing trees in a 3D point cloud. It also performed well as regards to the number of trees registered compared to a control measurement by a TLS system, and estimated the diameter at breast height of the registered trees close to that of a manual control measurement.

One can easily imagine an MLS system of this sort as part of a system with the ability to collect data for a complete forest inventory. On that note, this project aims to extend the MLS system by integrating a camera and through corresponding camera images determine the species of the registered trees in a 3D point cloud.

The project was conducted on behalf of the Swedish University of Agricultural Sciences (SLU) and supervised by Johan Holmgren at SLU.

1.1 Purpose and goal

The purpose of this project is to further develop the backpack-based system by integrating a camera, enabling it to classify trees mapped out in a generated 3D point cloud.

The final goal is to have a camera fully integrated with the initial system. This means that when at use, the system should automatically and continuously collect camera images. Further, the goal is to develop software for combined post processing of images and laser scanner data as to conduct tree species classification.

This breaks down into the following objectives:

- *Mount a camera on the backpack and integrate it to the initial system:* During a survey, the camera should be able to take images automatically and continuously without any need for the user to intervene.

- *Time synchronization of the camera with other parts of the system:* The camera images needs to be timestamped in order to relate the position they were captured to the corresponding locations in the georeferenced point cloud.
- *Spatial synchronization of the camera with other sensors:* In order to match spatially relevant point cloud data to a camera image, a spatial synchronization needs to be conducted.
- *Software development for automatic tree species classification:* This step consists of two parts: segmentation of an image so that only tree stems are considered, followed by a tree species classification.

2 Conceptual description of the system design

The following section intends to give the reader a conceptual understanding of how the system works. Section 2.1 describes the system setup as it was when the project took off. And with Section 2.2, we hope to give the reader an overall view of the extended system's design, so that the in-depth description of each separate part of the extension, presented in Section 3, can be understood in its context.

2.1 Pre-existing system setup

The original MLS system consisted of three separate sensors: a light detection and ranging (LiDAR) device, a global navigation satellite system (GNSS) and an inertial navigation system (INS). The sensors were interconnected and mounted on a backpack. The fully fledged system was operated by a computer, preferably at the hands of the backpack carrier.

2.1.1 Light detection and ranging

Light detection and ranging, or LiDAR, is a common method for modeling an environment in 3D. It measures the properties of reflected light to find, for example, the distance to objects. In this case the environment is mapped with the help of a laser scanner. The 3D model one obtains from such LiDAR are often called point clouds.

The existing laser scanner mounted on the MLS is a Velodyne VLP-16. It uses 16 laser/detector pairs mounted in a compact housing to scan the environment by spinning the housing rapidly, with a speed of 5-20 Hz (it is adjustable). This results in the laser firing up to 300 000 points per second. The Velodyne has a horizontal field of view at 360° and a vertical field of view of ±15°. It has a measurement range of up to 100 m [14].

2.1.2 Global navigation satellite system

A global navigation satellite system (GNSS) receiver is a navigation tool that recieves satellite signals from either the American global positioning system (GPS), or the Russian global navigation satellite system (GLONASS). For any given time, the positions of the satellites are known in a global coordinate system, and by measuring the distance between several satellites and a reciever, the global position of the reciever can be determined.

An unobstructed signal transmission between satellite and reciever is of crucial importance in order to rely on a GNSS. Since the forest canopy might obstruct the signal, additional sensors are needed for positioning in such environments.

The particular GNSS reciever mounted on the backpack is a NovAtel GPS-702-GG antenna. For technical specifications, see [9].

2.1.3 Inertial navigation system

As the LiDAR is firmly mounted on a backpack on move, its measurements will be subject to continuous effects from roll, pitch and yaw, relative to the world coordinate system. This is taken to account by mounting an inertial navigation system (INS), that measures those three features.

The INS also keeps track of its relative position, so by an initial pre-survey synchronization with a reliant GNSS position, it can act as a back-up positioning system that kicks in when the GNSS signal becomes unreliant.

The mounted INS is a NovAtel SPAN-IGM-S1. For technical specification, see [10].

2.1.4 Operating computer

The operating computer controls the initialization, i.e. the pre-survey synchronization of the INS and the GNSS as to obtain a reliant positoning system. This initialization is managed using the INS manufacturer's own software, NovAtel Connect.

After initialization but before conducting a survey, a Python script is run as to collect INS data. The script opens up a serial communication lane between the computer and the INS, where data from the INS is requested and logged at 125 Hz.

Although the LiDAR has a build-in clock of its own, it is known to drift with time. As a remedy, a corrective pulse per second signal was sent from the INS to the LiDAR. The LiDAR stores its measurement data as user datagram protocol (UDP) packets. Now, having accurately time-stamped packets, they are sent to the computer at a frequency of 10 Hz and logged by Wireshark, a software for network protocol analyses.

2.2 Conceptual setup and method of the system extension

After a survey, the data is post processed as to generate a 3D point cloud. The conceptual idea was to map 3D points of trees down on to a 2D camera image showing the exact same trees. This 3D-to-2D mapping should match so that the projected 3D-points fill up any tree stems showing up in an image. Over each such tree stem section, the image data should be processed as to classify which tree species it represents.

2.2.1 Spatial calibration of a camera image and 3D data points

The 3D-to-2D mapping provides a rigid geometric relation between the camera and LiDAR device. After such a physical system extension, the camera and LiDAR needs to undergo an extrinsic calibration process. This process is outlined in 3.3.

2.2.2 Automation of the image taking process

Microcontrollers are commonly used in embedded systems as to control certain functions using electrical in- and output signals. In our case, such a microcontroller is connected to the operating computer and the camera. The operating computer is used to power and initiate the microcontroller so that it issues a regularly recurring signal to the camera. This yields images taken on a regular basis.

2.2.3 Time synchronization of the camera and the rest of the system

The time instances when the images were taken needs to be known in order to connect it to relevant LiDAR data. The conceptual approach to this was to collect the output signal from the camera's hot shoe. As soon as the microcontroller picks up this signal it sends a message to the operating computer. The time instances when the operating computer receives these messages will then be the timestamps for the images. Since the time for which both the LiDAR and the computer collects data are GPS synchronized, the images' timestamps will be synchronized with the LiDAR data.

2.2.4 Software for tree classification

In machine learning, a Convolutional neural network (CNN), is a class of feedforward neural networks that lies in the forefront in the field of image classification.

The CNN field is under intense development and rapidly growing. Big companies has released open-source packages for anyone to use. This project has benefited from a CNN toolbox of Matlab.

3 Method

3.1 Instruments and system mount

The instruments used in this project can be seen in Fig. 3.1.

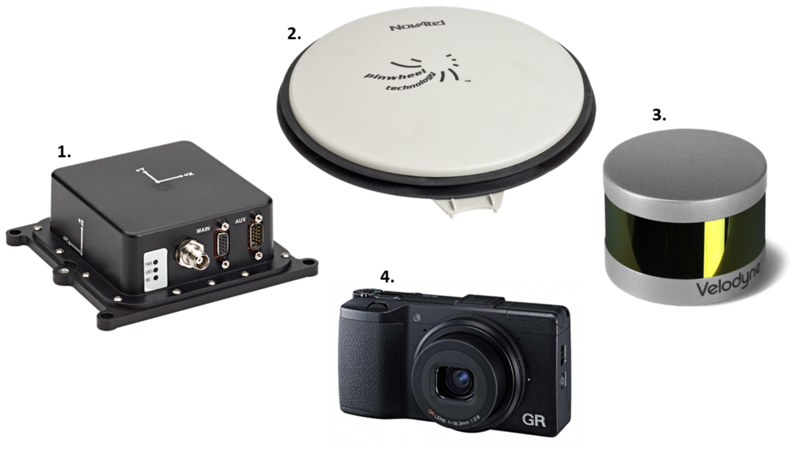


Figure 3.1 – 1. NovAtel SPAN-IGM-S1 (INS) [10]. 2. NovAtel GPS-702-GG (antenna) [9]. 3. Velodyne VLP-16 (laser scanner) [11]. 4. Ricoh GR (camera) [12].

The component added to the existing MLS system consists of the Ricoh GR camera. It is a compact digital camera equipped with an image sensor of type CMOS allowing for a resolution of approximately 16.2 megapixels [12]. This type of sensor is commonly used in so called rolling shutter mode, meaning, the image is captured one row of pixels at the time resulting in a time delay between the start of exposure for each row of pixels [13].

Figure 3.2 shows a sketch of how the different components of the MLS are mounted with respect to each other. The position of the camera is only a first measurement and is not completely exact. In Figure 3.3 the components can be seen on the actual MLS. Finally, for detailed instructions regarding how to carry out a survey using the extended MLS system, consider Appendix B.

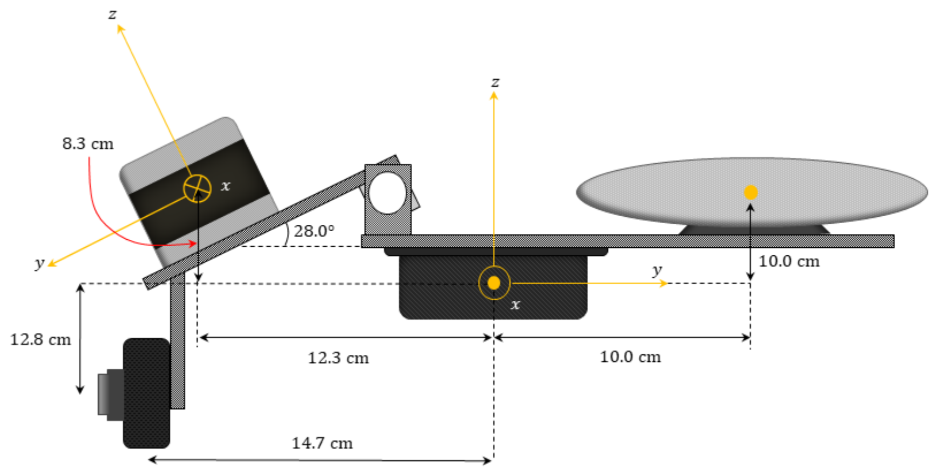


Figure 3.2 – Sketch of how the different components on the MLS are mounted with respect to each other.

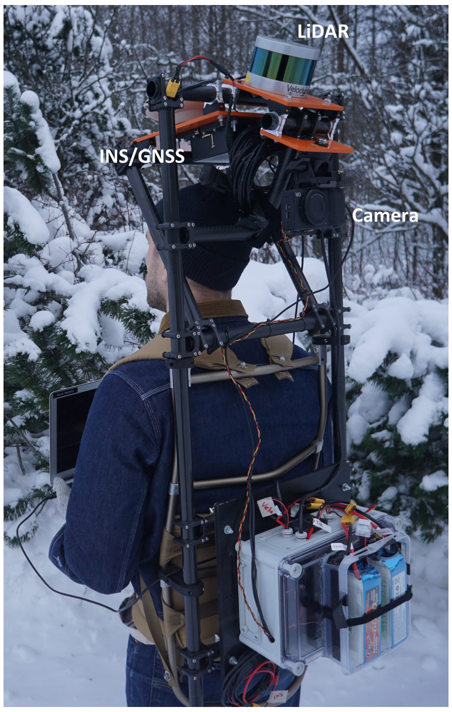


Figure 3.3 – Picture of the MLS with mounted camera.

3.2 Time synchronization of camera and INS

To know the position of the camera when it is taking a picture, the time for when the photo is taken need to be registered. This was done with the help of an Arduino Mega board, which is a type of micro controller, and the hot shoe of the camera. When the camera takes a photo the hot shoe will send a "signal" to the Arduino, which will then tell the computer that it registered that the camera was triggered. The hot shoe does not send this signal by itself, but a voltage has to be applied to the hot shoe since it then works as switch, closing the circuit when the flash should fire.

In this case the circuit in Figure 3.4 is used for the connection to the hot shoe. As can be seen 5 V is applied to the hot shoe from the Arduino, from pin 51 in the figure. Since the circuit of the hot shoe is not closed the micro controller will be able to register the 5 V in the analog pin (A0 in the figure). When the hot shoe closes it circuit we will see a sudden decrease in the voltage read by the analog pin, since the resistance will be lower in the path through the hot shoe in the circuit. This can then be used to see when the camera took the photo.

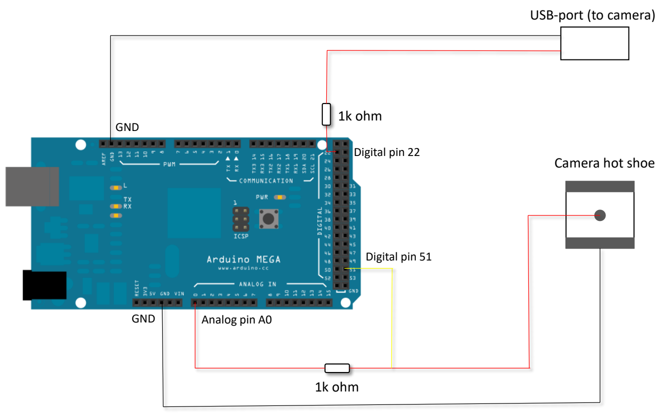


Figure 3.4 – Figure showing how the Arduino board is connected to the hot shoe and the camera USB cable.

When the python-script on the computer registers that the camera has been triggered, it saves the current GPS-time in a text file. This time will however not be precisely exact, since there will always be a small delay when sending and reading information. To see how long this delay was, a test was performed. The test was basically to print the current time on the screen of a computer about every microsecond and take pictures of it. The last time visible in the picture was then compared to what the logged time was when the computer thought the camera had taken a picture.

3.2.1 Camera triggering

To trigger the camera to take a photo the same Arduino board as for the time synchronization was used. The micro controller is connected to the camera via an USB cable, and through this cable "pulses" are sent to the camera. These pulses simulates what happens when the shoot button on the camera is pressed. First the Arduino sends a signal that the button is half pressed (i.e. the camera focuses), then that it is fully pressed and finally that the button is released [7]. Due to the length of these pulses, and that there has to be some pause between them, the shortest time between photos is limited to about 1 second.

Note that the code on the Arduino is specifically for the Ricoh camera. If a different camera is used, then the complete code for triggering the camera probably has to be changed.

3.3 Extrinsic calibration

In this section we present a simplified approach for solving the extrinsic calibration between a camera and LiDAR. Assuming that the backpack-based system is completely rigid; the task boils down to determine the rigid transformation from the LiDAR frame to the camera frame. Similar previous work has been carried out by, for instance [2], which provide accurate results. Such a method is more involved than what we present. It is however highly recommended for future reference since naturally we wish to minimize the error when mapping points between sensor frames. The approach considered deals with finding a common set of points corresponding to a calibration object in both frames. A schematic sketch of the experimental setup is presented in Figure 3.5. In particular, the common set of points will correspond to the four corners of the calibration object. For the point set in one frame, we use the single value decomposition to find the optimal rotation and translation that maps these points to the other frame. The procedure is more thoroughly described later in this section.

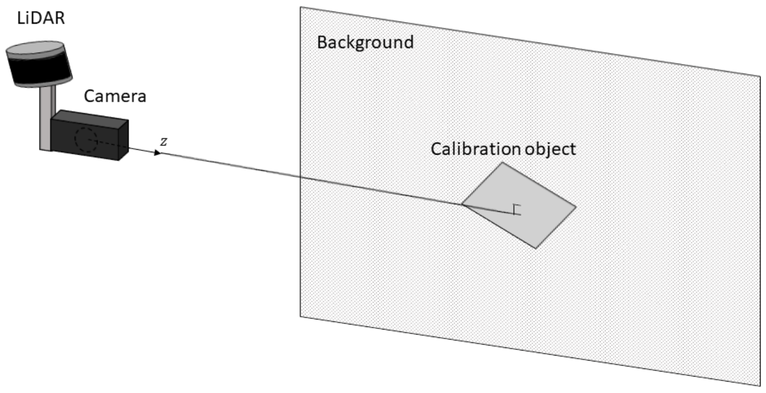


Figure 3.5 – Schematic sketch of the experimental setup regarding the extrinsic calibration between camera and LiDAR frame. Note that the object is mounted such that its surface is perpendicular to the z-axis in the camera frame. Also note that the LiDAR and camera are rigidly mounted together.

3.3.1 Object 3D-points in camera frame

This section concerns finding the set of 3D-points corresponding to the corners of the calibration object in Figure 3.5. We want to express the corners in the camera frame using real world distances. The somewhat tricky part arises since we which to map 3D-points acquired from the LiDAR into an image plane. For this task we use a somewhat sloppy, but direct method to obtain the real world distance in $\mathbb{R}^3$ from the camera frame to the corners. Before we begin, a brief description regarding image projection follows.

A camera projects 3D-points into a 2D-image plane according to Figure 3.6. More specifically the projection of 3D-points, $\mathrm{P}^{(w)}$, into 2D image coordinates, $\mathrm{P}^{(i)}$, may be expressed as

$$\mathrm{P}^{(i)} \sim \mathrm{P}^{(w)}\mathcal{P},$$

up to a scale, where $\mathcal{P} \in \mathbb{R}^{4\times 3}$ is the camera matrix. It is a the matrix product of the camera extrinsic and intrinsic parameters according to

$$\mathcal{P} = \begin{bmatrix} \mathrm{R} \\ \mathrm{t} \end{bmatrix} \mathrm{K},$$

The extrinsic parameters $[\mathrm{R},\ \mathrm{t}]^T$ describe the transformation from the world frame to the camera frame, where R and t constitute the rotation and translation. The intrinsic parameters, K, map the camera coordinates into the image plane. The

3×3 intrinsic matrix takes the form

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where f_x, f_y is the focal length, c_x, c_y the principal point and s is the skew [3]. Note that this matrix does not take into account lens distortion.

The procedure from world coordinates to image coordinates is summarized in Figure 3.7.

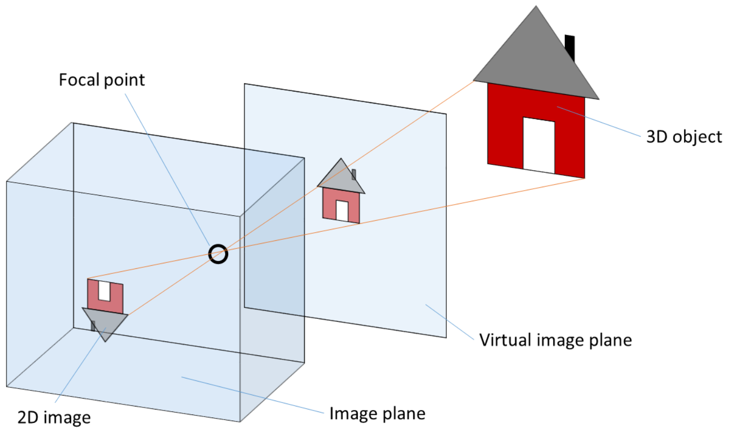


Figure 3.6 – Illustration of how a camera projects 3D-points to a 2D-image plane. Firstly, A point in 3D passes through the focal point of the camera and is rigidly transformed to a virtual image plane. From the virtual image the 3D-point is projected into a 2D-image plane.

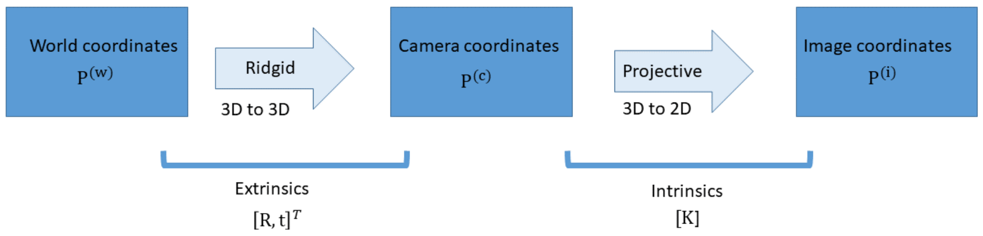


Figure 3.7 – Summary of how the extrinsic and intrinsic parameters relate to the projection of world coordinates to camera coordinates and finally to image coordinates.

The first step is to determine the intrinsic matrix K. For this task we make use of the MATLAB application `CameraCalibrator`. The procedure is simple and consists of gathering a number of images from a checkerboard pattern. For further details see [4].

We make use of a rectangular calibration target as presented in Figure 3.5. The object is assumed to be planar, where the real world dimensions are know. Given an image of the object, say that we manually select corners of the target in the image. A corner pixel point is expressed in homogeneous coordinates, i.e., $(x, y, 1)$. In order to go from pixel coordinates to the camera frame a point is multiplied with the inverse of K and scaled by a constant λ. This corresponds to moving one step from right to left in Figure 3.7. If λ is unknown we are stuck in the virtual image plane as illustrated in Figure 3.6. We can estimate λ by assuming that the z-axis in the camera frame is perpendicular to the calibration object. This translates to the z-component of each point in the virtual image being the same. Since the real world distances between all corners are know, we find the λ which produces these measures. Using this constant, each of the four corners of the calibration target may be expressed in the camera frame.

3.3.2 Object 3D-points in the LiDAR frame

Finding the corners of the calibration target in an image can be done manually with decent precision. Unfortunately, this is not the case for the LiDAR points. Since the LiDAR points are not evenly distributed, but concentrated along its 16 layers, the obtained points do not directly constitute the calibration target. Additionally points merely provide information regarding distance and intensity and not a color scale as in an image.

The experimental setup has been constructed such that points of interest are selected through intensity. The material of the calibration object is chosen such that points reflected from its surface differ in intensity compared to the background. Specifically we use a white dull paint as background and cover the calibration target with duct tape. Due to the uncertainty in the measurements points do not lie on a plane. As visible in Figure 3.8 we apply a plane fitting algorithm to the point set and project each point to that plane.

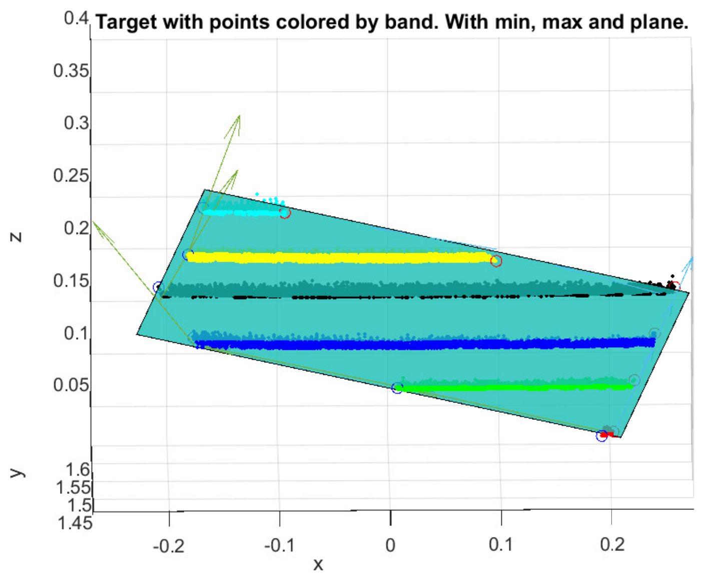


Figure 3.8 – Shows the 6 layers of laser data points which struck the calibration target and were collected through intensity. The blue surface corresponds to the fitted plane with respect to those points. Note that we have yet to project the data points to the plane. In this figure the shaded points in each layer lie behind the plane. The circular markers correspond to the maximum and minimum horizontal (x-axis) position for each layer. The vectors point in the direction of adjacent minimum/ maximum markers.

In the LiDAR frame; for each layer the two points corresponding to maximum and minimum horizontal (x-axis) position are selected. Using these points we construct vectors between adjacent layers, along all edges of the object. An example of this is presented in Figure 3.8. Given this set of vectors and the normal to the plane a basis for the point set is constructed. For a coordinate system using this basis, the left bottom corner of the calibration object is located and set as origin. Taking advantage of the known real world dimensions we estimate the position of the remaining three corners. Transforming these points back to the original LiDAR frame concludes this step.

3.3.3 Rigid body transformation

At this point we have managed to estimate the $\mathbb{R}^3$ coordinates to the four corners of the calibration target in both sensor frames. What remains is to determine the optimal rotation and translation which allows mapping of points from one sensor frame to the other. Note that we can choose which ever world coordinate system we please and we have not yet moved from the center to the left box in Figure 3.7. In particular, we choose the world coordinate system to be the LiDAR frame. Thus, solving for the extrinsics we have the necessary tools to map points between the sensor frames.

We denote the ordered set of corner points in the LiDAR frame, ${}^{\ell}\mathrm{P} = \{x_i\}_{i=1}^{4}$, and in the camera frame, ${}^{c}\mathrm{P} = \{y_i\}_{i=1}^{4}$. We seek the extrinsics, i.e., the translation vector ${}^{c}\mathrm{t}_{\ell} \in \mathbb{R}^3$ and rotation matrix ${}^{c}\mathrm{R}_{\ell} \in \mathbb{R}^{3\times 3}$ which minimizes the objective function

$$f({}^{c}\mathrm{R}_{\ell}, {}^{c}\mathrm{t}_{\ell}) = \sum_{i=1}^{4} ||({}^{c}\mathrm{R}_{\ell}\mathrm{x}_i + {}^{c}\mathrm{t}_{\ell}) - \mathrm{y}_i||_2^2. \tag{3.2}$$

The problem can be solved by applying the method presented in [8]. In short, this concerns first finding the optimal translation vector ${}^{c}\mathrm{t}_{\ell}$. Then finding the optimal orthogonal rotation matrix ${}^{c}\mathrm{R}_{\ell}$ and finally, if ${}^{c}\mathrm{R}_{\ell}$ is not a pure rotation, adjust it to be a pure rotation.

After completing the above steps a point in the LiDAR frame ${}^{\ell}\mathrm{P}_i = [{}^{\ell}\mathrm{P}_x,\ {}^{\ell}\mathrm{P}_y,\ {}^{\ell}\mathrm{P}_x]^T$ may be expressed in the camera frame according to

$${}^{c}\mathrm{P}_i = {}^{c}\mathrm{R}_{\ell}{}^{\ell}\mathrm{P}_i + {}^{c}\mathrm{t}_{\ell}. \tag{3.3}$$

Similarly a point in the camera frame is defined in the LiDAR frame according to

$${}^{\ell}\mathrm{P}_i = {}^{c}\mathrm{R}_{\ell}^{-1}({}^{c}\mathrm{P}_i - {}^{c}\mathrm{t}_{\ell}), \tag{3.4}$$

where we simply have reshuffled Eq. (3.3). This completes the extrinsic calibration. We now have the necessary tools to map points from the LiDAR frame to the camera frame and vise versa.

3.4 Tree classification using convolutional neural networks

For the sake of tree species classification we make use of a convolutional neural network (CNN). In machine learning, CNN is a class of deep neural networks which have proved to be very successful in the context of image analysis, among others. In particular, compared to the regular Neural Networks, the CNN takes advantage of the fact that the inputs are images. Along with this property the CNN makes for a suitable choice when it comes to tree species identification through images. A comprehensive review and introduction to CNN for image analysis is given by [15].

3.4.1 Transfer learning

One drawback when using a CNN is that is requires a sufficiently large dataset for training. In practice one typically takes a pretrained networks and use as a starting point or fixed feature extractor to learn a new task. This approach is commonly referred to as transfer learning. There are plenty of pretrained networks available. We have chosen to consider Alexnet, GoogLeNet, VGG-16 and VGG-19, which all have been trained on subsets of the ImageNet database [5]. The work carried out concerning these networks revolves around determining which is best suited for our particular application.

The three species of trees we wish to classify are Norway spruce (Picea abies), Scots Pine (Pinus sylvestris) and the more general classification of birch (Betula). This mean we require three categories. In particular, the tree species classification is to be carried out for image segments containing bark. Since no pretrained network specializes in classification regarding these three categories we must in some sense modify a prexisting one. Although ImageNet consists of $\sim$ 15 million images we note that it contains many categories and subcategories. We assume that our dataset will contain approximately the same number of images as a typical subcategory from ImageNet. In that sense our dataset can be considered relatively large. Another aspect to take into account is the similarity between the images in the pretrained network and the ones we wish to classify. A large difference is for instance comparing an object such as a pencil with a microscopic image. We further assume that there is only a slight similarly between the pretrained networks and the images we wish to classify. In summary we have a relatively large dataset with large difference in data compared to the original. It is then recommended to fine tune the network by retraining it from scratch, where we use the weights from a pretrained model as initializer. Since our dataset is considered large the chances of overfitting the data are slim [6].

3.4.2 Data acquisition and two classification problems

Since the tree species classification is to be carried out for image segments containing bark we require modifications of two separate pretrained network. These two networks are then to solve different classification problems. In order to proceed we must gather a sufficiently large relevant dataset. This translates to collecting a set of tree images, preferably similar to images obtained from the backpack-based system in a real experiment. In practice, these images were gathered by taking $\sim$ 400 photos in forest environments near Umeå University. In order to acquire as much data as possible each image was mirrored resulting in an original dataset of

$\sim$ 800 relevant images.

The pretrained networks have different demands regarding the input images sizes. Given a full-size image of a forest environment we construct a set of subimages with specified size through image segmentation. This was carried out by slicing an image according to a grid structure. Since each image was sliced into approximately 10 subimages our dataset now consists of approximately 8000 subimages. We wish to classify each of the subimages as one of three categories: Tree, partial tree or no tree. This requires the retraining of a pretrained network. For this task subimages were manually sorted in three folders corresponding to its category. After modifying the output layer the sorted dataset was used to retrain the network.

Assuming that we now have a network which, to sufficient accuracy, can classify a general tree bark image, we wish do determine the species. This requires another network. Thus, the set of images containing tree bark are further manually sorted into folders corresponding to the three species of interest. Using the sorted dataset allows for another training of a CNN such that the new network specializes in tree species classification from bark images.

With the two specialized networks and an efficient method for image segmentation we have all the necessary tools. Given an image from a forest environment we generate size specific image segments, collect the subset classified as tree bark images, and finally classify the species of those images. Note that, since we have no trash category, every tree bark image will be classified regarless of species. For instance, an image of a maple tree will be classified as one of three chosen tree species.

3.5 Post processing

After collecting data, there are a rather large amount of post processing to do. Both with the data from the INS and the laser scanner. The INS data is post processed using Inertial Explorer and the steps involved in doing this can be seen in the user guide in Appendix B. After the INS data has been processed the laser scanner data and the INS data is fused together using four MATLAB tools developed by Michael Tulldahl at the Swedish Defence Research Agency (FOI). This step is often referred to as the dynamic calibration.

3.5.1 Dynamic calibration tools

The dynamic calibration basically consists of four steps, but mainly the three first has been used in this project due to some unstable results from the fourth step.

The first step consists of unpacking the raw LiDAR data into different chunks. The second step combines the processed INS data with the unpacked LiDAR data. The third step is the dynamic calibration processing step. It finds trees in the laser data and calibrates the walked path a bit so that the trees will appear in the same place. If the same tree is walked by a second time, it will however be considered as a second tree and due to drifting in the INS measurements this tree will often not appear at exactly the same place as the other.

The fourth step consists of fusing together multiple trees that should be the same and correcting the height of the ground so that there is not multiple grounds.

For more information about the dynamic calibration see Appendix B.

3.5.2 Mapping laser data to pictures

When the entire point cloud is generated, and expressed in a global coordinate system, it is possible to start mapping laser data into the pictures. The location and pose of the camera at the time of each picture is known from the time log and the INS data which is gathered during the survey. This means that, for each picture, it is possible to transform the point cloud to the camera coordinates. With the point cloud expressed in the camera coordinate system, it is possible to filter out points which are not in the field of view of the camera. After this, the remaining points can be mapped to the image by first transforming to homogeneous coordinates, that is, normalize by the z-coordinate, and then multiplication from the left with the K-matrix, the intrinsic matrix of the camera.

4 Results

Results from the time synchronization and extrinsic calibration of the extended system are presented in section 4.1 and 4.2, respectively. In section 4.3 results for 3D point mapping of trees to camera images are presented.

Classification results for our two CNN are also presented below, in section 4.4. But since the code for connecting post processed survey data with the CNN networks is not ready for use, we can not present any classification results from an actual survey using the extended system. Instead, the classification results are for manually taken bark images.

4.1 Results from time synchronization

The time synchronization was tested by printing the current time on the screen of a computer about every microsecond and taking pictures of it. The idea was to compare the last time visible in the picture to what the logged time was when the computer thought the camera had taken a picture. However, a camera does not take a picture in one instant, since the shutter of the camera has to be open for some time. This, and the fact that the camera had a rolling shutter (i.e. it does not take the complete picture in the same time but starts with the top pixels and moves down or the opposite). These two factors made it a bit difficult to decide which time actually was the last one in the picture. Different times got printed over each other, and the ordering of them was sometimes not as it should (with later times being before earlier times).

By ignoring some of the difficult pictures, the time lag between the registration of the picture and when it seemed to be taken was determined to be about 0.056 seconds, as can be seen in Table 1. There it is also possible to see that the maximum uncertainty of that number is at about 16 milliseconds. When walking in a straight line this should not be any problem. It could possibly be a problem if there are sudden movements when the picture is taken.

Table 1 – Summation of the results from the time synchronization test

Total average time difference	0.05642 s
Maximum time difference	0.07213 s
Minimum time difference	0.04113 s
Standard deviation in time diff.	0.00657 s
Number of samples	134

4.2 Results from extrinsic calibration

The parameters corresponding to the intrinsics of the Ricoh RG camera are presented in Appendix A. Using these intrinsics together with the translation vector and rotation matrix as presented in Appendix A we obtain the relation between the LiDAR and camera frame. For the rotation matrix we have used Euler angled X-Y-Z. The obtained matrix and vector appear reasonable from what can be estimated from visually examining the geometrical relation between the sensors.

In Figure 4.1 the calibration object together with the LiDAR points mapped to it can be seen. This was for a stationary case, i.e. the INS/GNSS was not active during the gathering of the data. As can be seen the fit is not completely perfect, but sufficient for the application. The difficulty in determining the calibration target edges from the LiDAR data is reflected in the figure. However, the final predictions of the object corners shows good precision. These are also included in Figure 4.1 and are represented by lonely blue markers close to the corners.

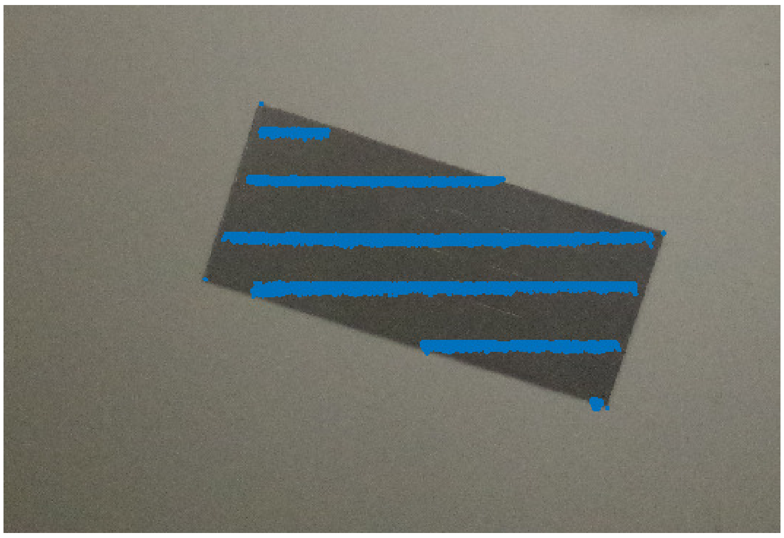


Figure 4.1 – Picture of calibration object, with the corresponding points from the laser scanner points mapped to it from the extrinsic calibration.

In Figure 4.2 the mapping of the LiDAR data to an image can be seen for a test case when the LiDAR was stationary. As in Figure 4.1 the fit is not completely perfect. The deviations appear to increase when moving from left to right in the figure. In particular, the deviations are especially noticeable at the edge of the rightmost wall.

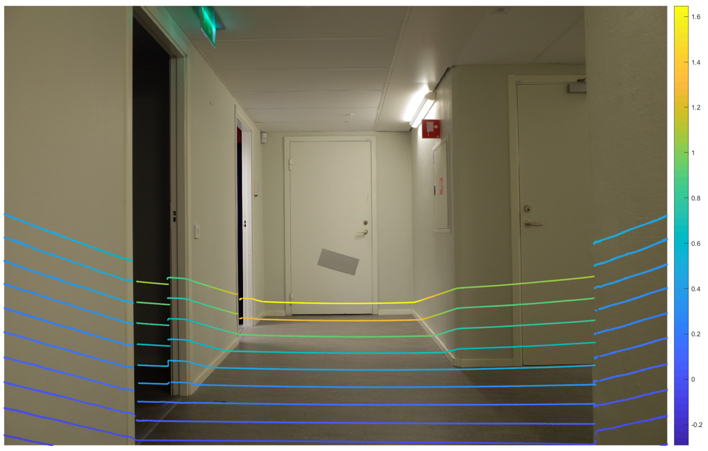


Figure 4.2 – Picture of a corridor together with the point cloud mapped to it.

4.3 Survey results

The complete MLS system was tested by walking in a small forest area between SLU and Umeå University. The camera was set to take a picture every 5th second. In Figure 4.3 the complete walked path can be seen together with the laser data for the tree stems. In that figure one of the pictures shows the mapped stem laser points mapped and it seems like a rather good match. That is, however, only one of the pictures and for a few of the matches it was not at all as good. This can be seen in the two examples in Figure 4.4. In one of them the the MLS was not moving, but was tilted a bit to test what effect it would have on the result. The other one is taken while moving away from a tree.

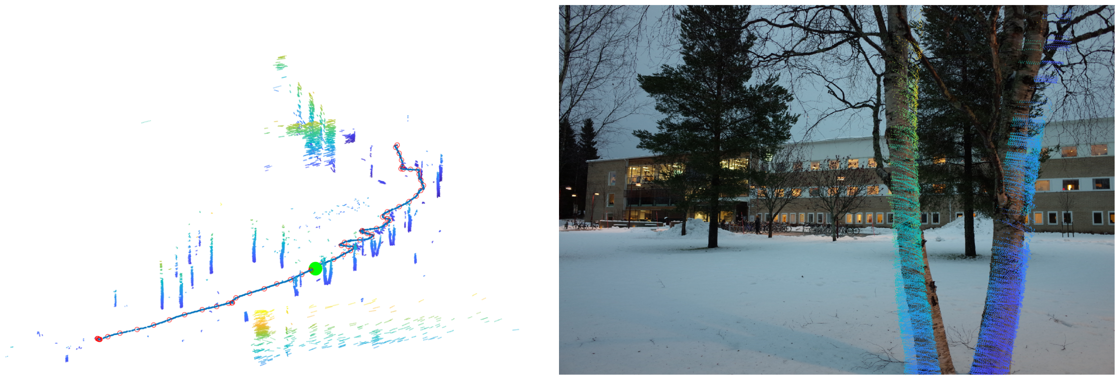


Figure 4.3 – In the left figure the walked path marked with red circles where a picture was taken can be seen. There the laser data for the nearby trees are visible. The green dot marks where the figure to the right was taken. The right figure shows a tree from the laser data mapped to a picture of the same tree.

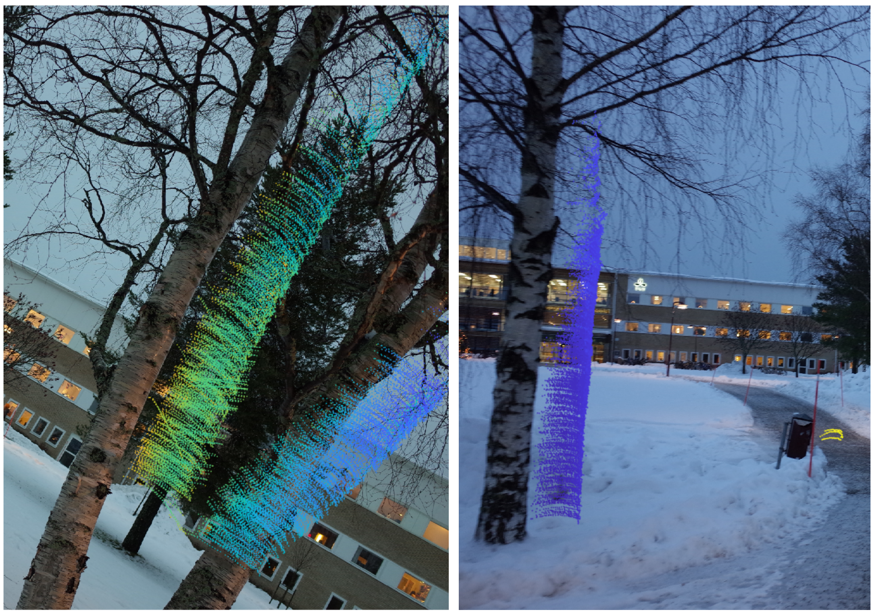


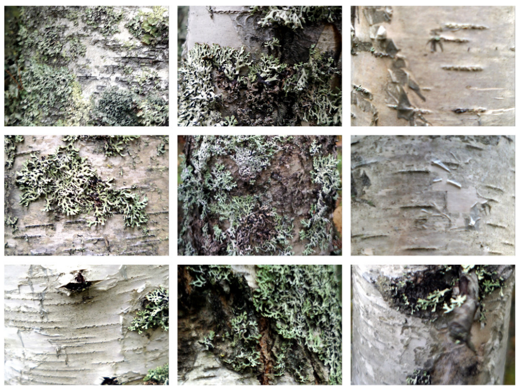
Figure 4.4 – Pictures of trees from the laser data mapped to the camera pictures for the corresponding trees.

The laser data in the pictures has not been through the fourth step in the dynamic calibration where multiple trees are deleted and a vertical correction is performed. This is due to very strange results from that step, with a ground level shifted to be 60 m below another ground level. This might be explained by the fact that there where not enough trees in the area where the test was performed, since this step has shown unstable behavior in areas with few tress, according to [1].

4.4 Results from CNN tree classification

The CNN was trained to classify images into three classes, birch, spruce and pine. During training, 20% of the data set was used as a validation set, and the rest of the images were used for training. The network achieved a higher classification accuracy than 95%. In the image below, Figure 4.5, we see a small subset of the validation set consisting of nine images of each class. Out of these images, only one of the images was incorrectly classified. This was the top right image of the spruce data set, which was incorrectly classified as an pine image.

(a) Stem sections pine.


(b) Stem sections birch.

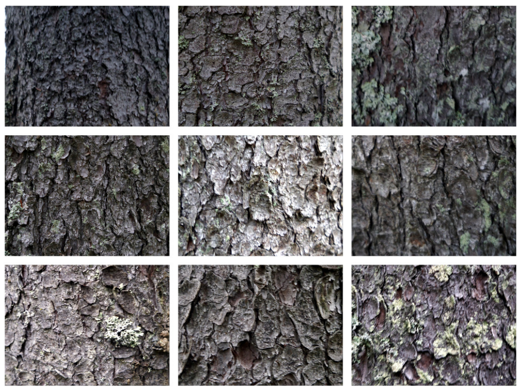

(c) Stem sections spruce.

Figure 4.5 – Images of stem sections which constitute a small subset of the validation set. All images was correctly classified by the CNN, except the top right spruce image which classified as pine.

As we see, we have a very high classification accuracy. However, the images in the training and validation sets have not been taken using the MLS.

5 Discussion

5.1 Ricoh GR camera

There are some parameters regarding the camera that has not been investigated. Such as, what should the optimal time between triggering be, what shutter speed and ISO should be used by the camera, should auto focus or manual focus be used and if the Ricoh GR camera is optimal for the purpose of the project.

The basic reason why the Ricoh camera was chosen was because it was already available, it was small and it had a hot shoe (which is very important). Some reasons to change camera might be to have one with a higher resolution or to get one with a global shutter. The problems that can occur with a rolling shutter, such as things bending when they are moving fast, does not seem affect the results in the tests done in this project but it does not mean that they will not appear at all. As said before, this is not something that has been investigated.

Another part of the camera that has not been investigated is the battery life time. Only shorter tests has been done with the complete MLS with the camera and thus it is not known how long the camera will operate before it powers down. It might be worth investigating since if the battery time is short another way of supplying the camera with power might be needed.

5.2 Extrinsic calibration

One thing that might result in an error is that the camera is assumed to be parallel to the calibration object. This might well not be the case since we merely used our eyes and a ruler when aligning the camera to the wall on which the calibration object was hung. To minimize such an error, we suggest using a more rigorous approach such as stereo-photogrammetry of the extrinsic calibration setup.

Another source of the error might be the fact that in the method used for the extrinsic calibration, the errors are assumed to be isotropic. This could be a false assumption since the depth/z-direction (distance from camera) should be much more difficult to decide compared to the other directions, and thus the extrinsic calibration might not be the most optimal one.

Despite the above we do not believe that the extrinsic calibration is the major source of error. However, for future reference a more accurate method is highly recommended. Consider for instance [2].

5.3 CNN tree classification

It is of great interest to evaluate the performance of both CNNs using image segments obtained from a survey. Although the classification results appear promising, the task of true testing the CNNs remain. Depending on the outcome from such testing, it might be necessary to re-train one or both networks using a larger data set. In particular, it would be optimal to re-train both networks using a sufficiently large data set consisting of image segments obtained from several different experiments. This will fine tune the network with images which are taken, and correctly labeled, during the same conditions as the actual application will present. For instance, since images are to be taken while moving, there is a possibility that movement blur in the images might cause a problem for the classifier. However, if an accuracy of about 80% can be achieved during actual application this would be a sufficient accuracy as a proof of concept.

5.4 Post processing

From what we have observed, it has been concluded that the post processing seems to be the major source of errors particularly evident in Figure 4.4.

Consider the right subfigure in Figure 4.3 and the left subfigure in Figure 4.4. Both these are results obtained when the MLS was temporarily stationary. The only difference is that the backpack was tilted in the latter, yet we see a large difference in accuracy of the projected points. We believe that this kind of error resides from the third post processing step as explained in section 3.5. Since the algorithm which extracts stem sections has not been constructed by us, the process is somewhat vague. One possibility is that, as this step extracts trees, a tree receives a fix position according to the first identified stem section. As further stem section are identified, these are fitted to match the first. The tree in the right subfigure in Figure 4.4 is first detected when the backpack carrier faces the tree. At this point the laser has its greatest uncertainty in depth. If we were to position ourselves at the trash-can in Figure 4.4 and glance in the direction of the tree, the projected laser data would have a nice overlap. However, from the position the image was taken, the error is large. This is due to the fix position the tree obtained at identification of the first stem section.

It is possible that the fourth processing step, described in section 3.5, could improve the results. This step only makes sense if, for instance, the start and end positions of the trajectory are near. This is such that the backpack carrier passes each tree twice or more. At least once on the way out and once on the way back. Assuming a tree obtains a fixed position on the way out with some error. The same tree would then be detected on the way back with opposite error. The fourth steps takes this into account and estimates a suitable final tree position. Unfortunately, we were not able to apply this step on the survey since the density of tree was far to low [1].

Future work should definitely concern a survey with sufficient tree density and appropriate trajectory as to apply the fourth post processing step. It would be of great interest to observe the accuracy of laser data projected to images captured during such a survey.

6 Conclusions and outlook

The extended backpack-based MLS system covered in this paper shows promising features and results. The concept of mapping trees from a 3D-point cloud onto camera images depicting the corresponding trees has been successfully implemented. Although, currently the mapping quality is of inconsistent quality when the entire system is used in a real-life situation. This can be due to a number of reasons.

Any errors associated with the technical methods and solutions that were provided by us in connection with integrating the camera, lies in the time synchronization and extrinsic calibration. The impact of these errors are, however, difficult to consider before the pre-existing code for post processing a 3D point-cloud has been investigated further. This code was not developed with any camera mapping in mind, and seems suboptimal in that regard. We strongly believe that this should be a first step to develop this MLS system further.

With a more accurate mapping, the task of segmenting images into stem sections still remains. Our idea was to use a bounding box to enclose the stem section of each tree. Each such bounding box would then be partitioned into images adequately sized to be handled by our CNN networks. We have developed code modules aiming for that, but these need to be tested and, most likely, further developed.

7 References

[1] Tjernqvist M. (2017) *Backpack-based inertial navigation and LiDAR mapping in forest environments* (Master's thesis). Retrieved from http://umu.diva-portal.org.

[2] Rodriguez S, Fremont F, Bonnifait P. *Extrinsic calibration between a multi-layer lidar and a camera. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, MFI 2008, Aug 2008, South Korea. pp.214-219, 2008, .

[3] Calibrator, C. and App, S. (2017). What Is Camera Calibration? - MATLAB & Simulink - MathWorks United Kingdom. [online] Se.mathworks.com. Available at: https://se.mathworks.com/help/vision/ug/camera-calibration.html (Accessed 2017-12-11)

[4] Calibrator, C. and Calibrator, S. (2017). Single Camera Calibration App - MATLAB & Simulink - MathWorks United Kingdom. [online] Se.mathworks.com. Available at: https://se.mathworks.com/help/vision/ug/single-camera-calibrator-app.html (Accessed 2017-12-11)

[5] Image-net.org. (2017). *ImageNet.* http://www.image-net.org/

[6] Cs231n.github.io. (2017). *CS231n Convolutional Neural Networks for Visual Recognition.* [online] Available at: http://cs231n.github.io/transfer-learning/ (Accessed 2017-12-20)

[7] Muttyan's home page. *SDM with RICOH CA1 Remote Release Cable* http://stereo.jpn.org/eng/sdm/ca1.htm (Accessed 2017-10-26)

[8] Söderkvist, I. and Wedin, P. (1993). *Determining the movements of the skeleton using well-configured markers.* Journal of Biomechanics, 26(12), pp.1473-1477.

[9] NovAtel. *GPS-702-GG*
https://www.novatel.com/products/gnss-antennas/high-performance-gnss-antennas/gps-702-gg/ (Accessed 2017-12-18)

[10] NovAtel. *SPAN-IGM-A1*
https://www.novatel.com/products/span-gnss-inertial-systems/span-combined-systems/span-igm-a1/ (Accessed 2017-12-18)

[11] Velodyne. *PUCK - VLP-16* http://velodynelidar.com/vlp-16.html (Accessed 2017-12-18)

[12] Ricoh Imaging. *GR - Överblick* `http://www.ricoh-imaging.se/se/kompaktkameror/group/5/body/overview/ricoh-gr.html` (Accessed 2017-12-18)

[13] QImaging (2014). *Rolling vs. Global shutter.* `https://www.qimaging.com/ccdorscmos/pdfs/RollingvsGlobalShutter.pdf` (Accessed 2018-01-10)

[14] Velodyne LiDAR (2015). *VLP-16 USER'S MANUAL.* `http://velodynelidar.com/docs/manuals/VLP-16%20User%20Manual%20and%20Programming%20Guide%2063-9243%20Rev%20A.pdf` (Accessed 2018-01-11)

[15] Rawat W, Wang Z. (2017) *Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review.* 2017 Massachusetts Institute of Technology. Neural Computation 29, pp.2352–2449.

A Extrinsic calibration parameters

The intrinsic parameters of the Ricoh GR was found as

$$\mathrm{K} = 10^3 \begin{bmatrix} 3.925 & 0 & 2.4216 \\ 0 & 3.9256 & 1.6473 \\ 0 & 0 & 0.0010 \end{bmatrix}.$$

The rotation matrix and translation vector for expressing a point in the LiDAR frame in the camera frame is given by

$${}^{c}\mathrm{R}_{\ell} = \begin{bmatrix} 0.9999 & 0.0122 & 0.0021 \\ -0.0071 & 0.4241 & 0.9056 \\ 0.0101 & -0.9055 & 0.4242 \end{bmatrix}, \quad {}^{c}\mathrm{t}_{\ell} = [-0.0300, -0.2200, -0.0400]^T$$

where ${}^{c}\mathrm{R}_{\ell}$ is described by Euler angles by the Z-Y-X convention.

B User guide for the MLS system

This user guide provides a thorough description of the actions required to take in order to set up, Section B1.1, and carry out a survey, Section Bhttps://preview.overleaf.com/public/s using the extended mobile laser scanning (MLS) system with an integrated camera. In addition, a description of important steps in order to successfully conduct the necessary post-processing of the raw data acquired are provided in Section B2. These instructions together with user guidelines presented in Appendix A and B of Mattias Tjernqvist master's thesis work *Backpack-based inertial navigation and LiDAR mapping in forest environments* serves as a complete description of the data post-processing. The purpose being to acquire essential data for further performing point cloud and image based tree detection and tree species identification. Finally, as a an unexperienced user of this system, *navigate the user guide Section-wise from top to bottom* for a successful outcome.

B1 Gathering data

The software dealt with for collecting the required data consists of:

- NovAtel Connect for initializing and aligning the INS.
- WireShark to log laser scanner data.
- A Python script to request and log INS data, enable camera triggering at predefined time intervals and to log timestamps of captured camera images.

Following the steps in the current section you will have acquired camera images from later which tree species may be identified and the data required for carrying out the post-processing as described in Section B2. The data includes:

- INS data contained in a .gps-file:

  `INSlogYYYY-MM-DD_hh-mm-ss.gps`
- Two laser scanner data files, one WireShark capture file (.pcapng) and one K12-file.
- Timestamps of captured camera images contained in a .txt-file:

  `cameraTimeLogYYYY-MM-DD_hh-mm-ss.txt`
- Camera images saved on the camera memory card.

The contents of the log file of camera image timestamps are simple. The file contains a single column of successive timestamps of the images captured. The timestamps are expressed in seconds since start of week. This is the time format used by the INS for which positions and orientations corresponds to in the .txt-file obtained performing the post-processing of the INS-data. The raw INS data contained in the .gps-file is used in the INS data post-processing, Section B2.1. The two laser scanner data files are required by the dynamic calibration algorithm and used for post-processing the laser scanner data, addressed in Section B2.2.

B1.1 Setting up

B1.1.1 Hardware

The user of this system is required to be aware of essentially three hardware components: the computer, the camera and the MLS. Firstly, configure the camera such that *manual camera flash is enabled*, this is to be able to log timestamps of the captured images. Other camera and flash settings can be used, but since they have not been tested thoroughly it can not be guaranteed that it will work for all cases.

Step 1. Power up the camera and set the settings wheel, see Figure B1.1, to not be set at auto. It should preferably be set to "T" or "TA", where the shutter speed can be determined. A shorted shutter speed is recommended.

Step 2. The internal camera flash need to be activated to be able to change the camera settings, so activate the flash. This is done by opening the flash by pulling the switch marked in Figure B1.2.

Step 3. Change the flash settings to be manual, by pressing the button marked in Figure B1.3 until "Manuell blixt" is marked.

Step 4. Push down the internal flash to deactivate it.

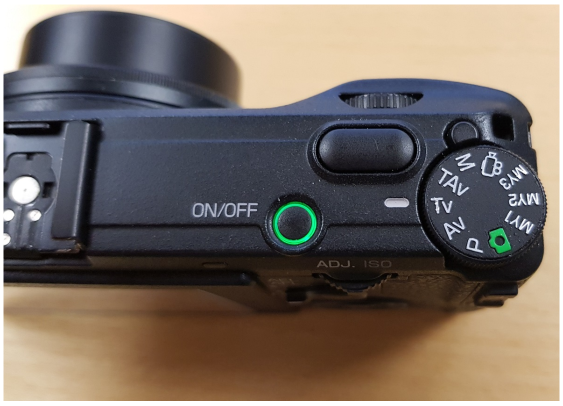


Figure B1.1 – Camera top.

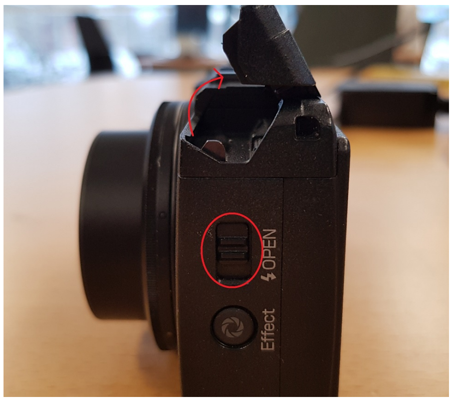


Figure B1.2 – Camera side.

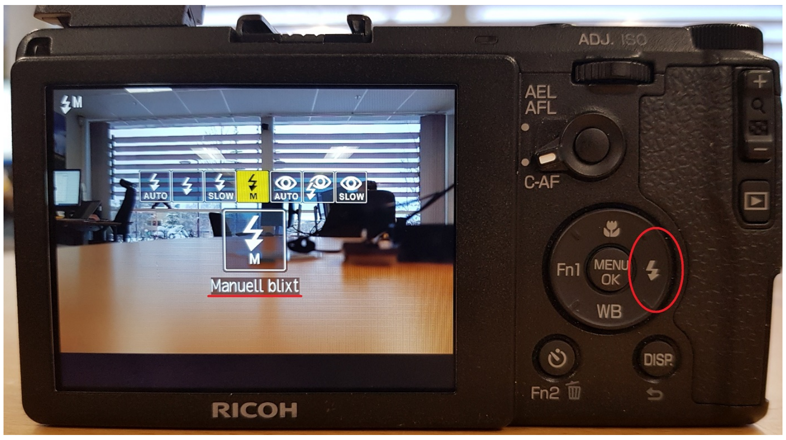


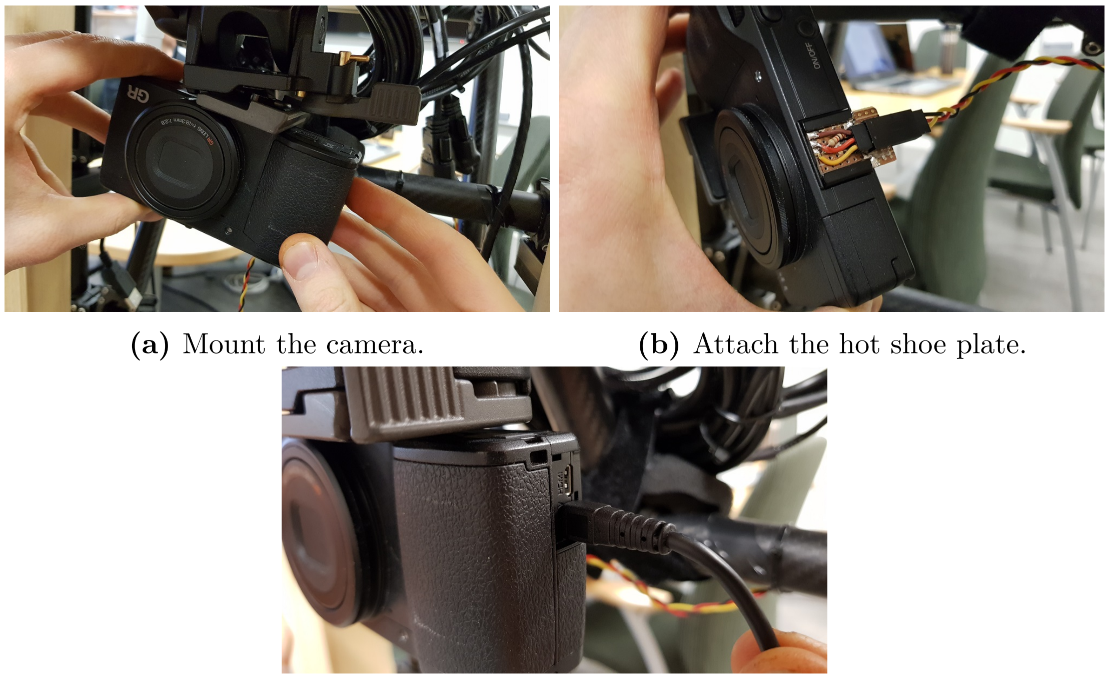
Figure B1.3 – Camera flash settings.

Secondly, having configured the camera the following **Step 5-8** provides information regarding connecting the remaining parts included in the MLS system.

Step 5. Mount the camera on the MLS system using the attachment beneath the laser scanner as shown in Figure B1.4a.

Step 6. Remove the camera hot shoe cover such that the hot shoe is visible as in the left periphery of Figure B1.1. Then, attach the hot shoe plate as displayed in Figure B1.4b.

Step 7. Connect the triggering cable to the camera USB/AV socket as illustrated in Figure B1.4c.



(a) Mount the camera. **(b)** Attach the hot shoe plate.

(c) Connect the trigger cable.

Figure B1.4 – Illustrations of how to mount and connect the camera to the MLS System.

Further, connect the cables from the INS, laser scanner and Arduino to the computer. At this stage, it is important that the included parts are connected to the computer using the correct sockets. For this reason, Figure B1.5 provides an illustration of the specific sockets to use and these are described in the next step.

Step 8. Consider Figure B1.5 and connect the black USB cable from the Arduino to left USB socket of the computer, the INS USB cable to the right USB socket and the laser scanner Ethernet cable with an attached Thunderbolt to Gigabit Ethernet adapter to the Thunderbolt socket on the computer.

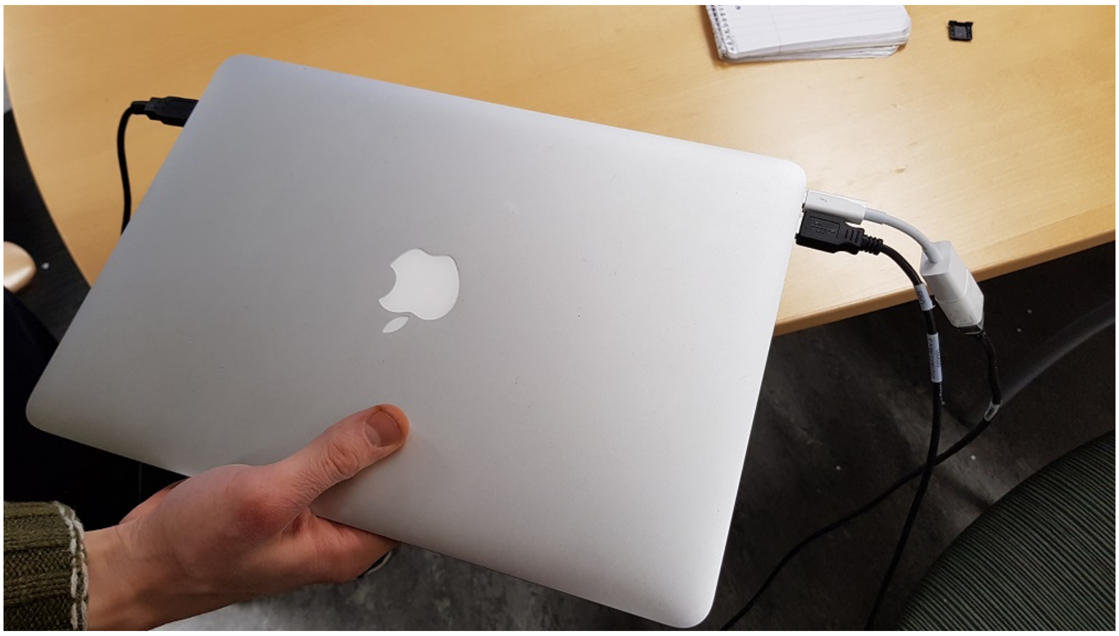


Figure B1.5 – Displays the proper sockets to use for connecting the Arduino (black USB, left in figure), INS (black USB, right in figure) and laser scanner (white Thunderbolt to Ethernet adapter, right in figure) to the computer.

Finally, in Section B1.1.2 and in **Step 1** in Section B1.2.2 it is addressed when appropriate to supply the INS and laser scanner with power. Currently, prior to supplying the two sensors with power, it is recommended to attach the two devices, one of them is shown in Figure B1.6, to the batteries. From these it is possible to read off the voltage of each cell of the two batteries which provides an indication when it is suitable to charge the batteries. That is, when the voltage read off is beneath 22 V.

Step 9. Attach the measurement devices as illustrated in Figure B1.6 to read off the voltage of each cell of the battery. Having successfully attached the devices, this is clearly indicated by a sequence of loud beeping noise for a short duration of time.

To conclude, Figure B1.7 provides indicates of how to supply the INS and laser scanner with power and are referenced later when this is appropriate.

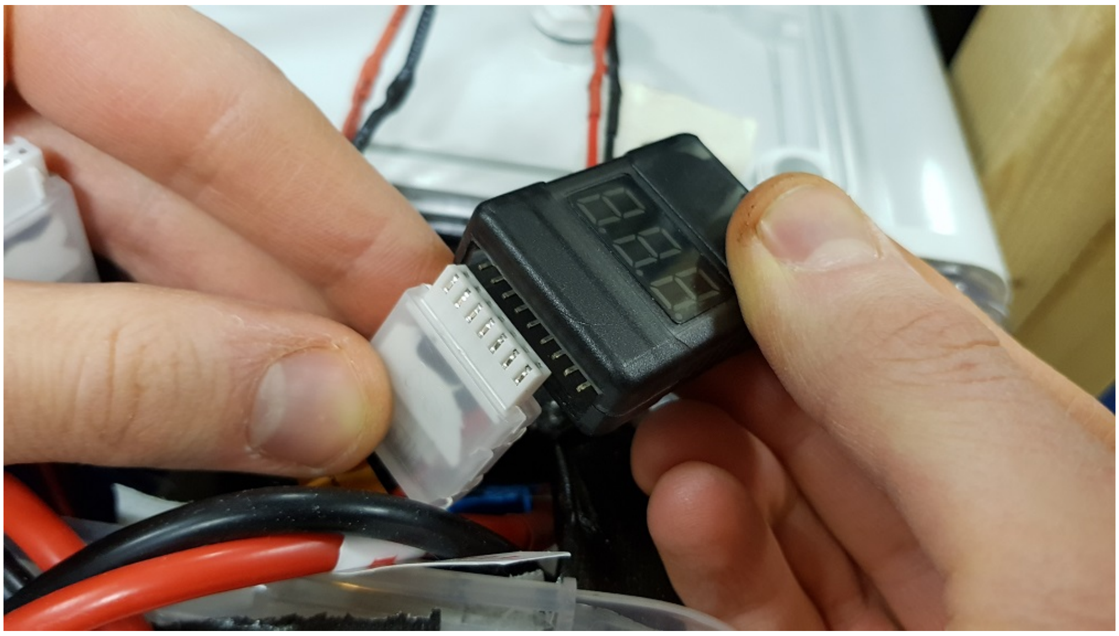
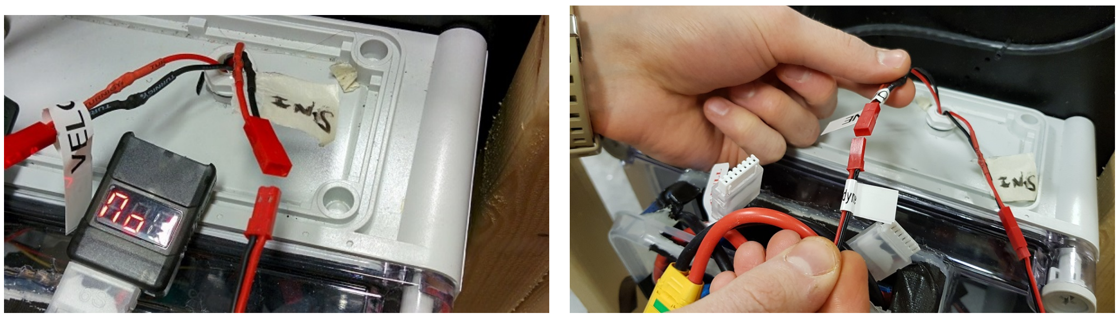


Figure B1.6 – Displays how to attach the measurement devices for reading off voltage and current of each cell of the battery it is connected to.



(a) INS power supply. **(b)** Laser scanner power supply.

Figure B1.7 – Illustrations of how to supply the INS (left) and the laser scanner (right) with power.

B1.1.2 Software

The setup regarding the software is brief and how to utilize the software is covered by the instructions given in the appropriate sections. It should be mentioned that the INS data file and the log file of captured camera image timestamps:

- `INSlogYYYY-MM-DD_hh-mm-ss.gps`

- `cameraTimeLogYYYY-MM-DD_hh-mm-ss.txt`

will be located in the same directory as the Python script is located in when running the script. Before carrying out the instructions in section B1.2 it is recommended to open up a command prompt which can be done, for example, as follows. In Windows, start the application "Run", in the field indicated by "Open:", write "cmd" and press the OK button which opens up a command prompt. Navigate to the directory where the Python script is located. This is performed by issuing the command:

```
cd <Path of the Python script>
```

and this is sufficient if the Python script is located in a directory on the C drive. Otherwise, if the Python script is located in a directory on the D drive. Also issue:

```
D:
```

and your current directory is now `<Path of the Python script>`. Keep the command prompt open as you will later return to it for **Step 2** in Section B1.2.2.

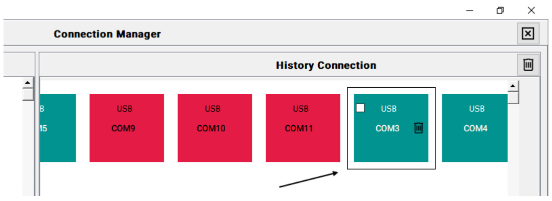
B1.2 Running

Prior to gathering data, it is required to initialize the INS and to perform a kinematic alignment. Supply the INS with power as shown in Figure B1.7a and follow the steps in Section B1.2.1.[1] Having performed the kinematic alignment following the steps in Section B1.2.1, the procedure for performing the actual data acquisition is described in Section B1.2.2.

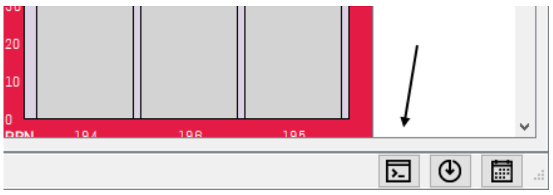
B1.2.1 INS initialization and kinematic alignment

The initialization and kinematic alignment of the INS begins with running the software NovAtel Connect and is monitored and controlled through the NovAtel Connect GUI which is shown in Figure B1.8. The procedure consists of four main steps which are described below Figure B1.8.

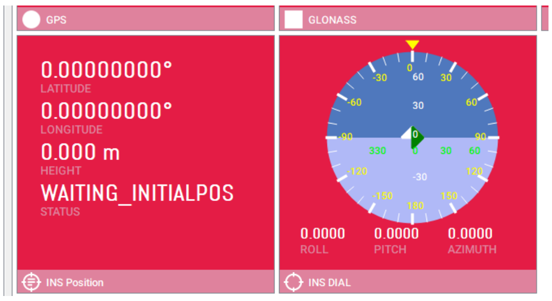
[1]At this point, it is perfectly fine to also supply the laser scanner with power. However, this is not required until later when the collection of laser scanner data is issued through the use of the WireShark software.



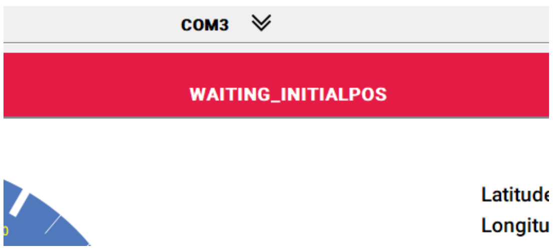
(a) COM-ports.



(b) Console window button.



(c) INS Position, the left window.



(d) INS status.

Figure B1.8 – Extracts of the NovAtel Connect GUI providing indicates of the steps required for performing the initialization and kinematic alignment of the INS.

Step 1. Having started the NovAtel Connect software, connect to an available COM-port, e.g. COM3[2], by double-clicking the box indicated in Figure B1.8a.

Step 2. Navigate to the console window button found in the bottom right corner of the NovAtel Connect GUI, indicated in Figure B1.8b. Click the button, and in order to properly initialize the INS enter the following six commands in the console window (one by one followed by pressing the enter key):

```
SETIMUORIENTATION 6
VEHICLEBODYROTATION 0 0 0
APPLYVEHICLEBODYROTATION enable
SETIMUTOANTOFFSET 0.1 0 -0.1
ALIGNMENTMODE AUTOMATIC
SETALIGNMENTVEL 1.15
```

Now, the INS is initialized and the final two steps concerns performing the kinematic alignment of the INS. This procedure is surveyed through the GUI of No-

[2]Since the serial communication between the INS and the computer as well as between the Arduino and the computer is currently utilized through COM-ports COM5 and COM6 respectively, these are unavailable. Therefore, a proper choice would be for example COM3.

vAtel Connect and are to be performed in a clear sky environment.[3] Navigate to and double-click the window in the NovAtel Connect GUI labelled "INS Position", that is, the left window shown in Figure B1.8c. The header displayed in Figure B1.8d which reads "WAITING_INITIALPOS" indicates the INS status. Proceed with the MLS system to a clear sky environment, the INS status should then change from "WAITING_INITIALPOS" to "ALIGNING". Finally, proceed with performing the following two steps in order to complete the kinematic alignment.[4]

Step 3. Walk in a straight line, that is, with constant heading at a speed of minimally 1.15 m/s until the INS status, Figure B1.8d, changes from "ALIGNING" to "ALIGNMENT_COMPLETE".

Step 4. Walk in a figure-eight motion as well as perform occasional random stops of a couple of seconds each until the INS status changes from "ALIGNMENT_COMPLETE" to "GOOD".[5]

This completes the initialization and kinematic alignment of the INS and thus, the steps in this current section.

B1.2.2 Performing the data gathering

The steps required to be carried out for successfully performing the data acquisition are given by **Step 1-9** below, but firstly, a word of advice for the data gathering of INS, laser scanner, camera image and camera image timestamp data. *It is crucial* to only log laser scanner data while the addtional data logging is conducted. In this context, this amounts to only run the WireShark data logging while the Python script is running.

Moving on, it is now appropriate to proceed to the area where the data gathering is to take place. Assuming that this is an area such as a canopy covered forest environment of difficult GNSS availability. It is then recommended to start the INS and camera data logging in a clear sky environment with sufficient satellite availability prior to entering the forest environment. In short, approach the forest environment and prior to entering, log INS and camera data, when inside the forest

[3]That is, an outside environment without any excessive canopy making satellite availability difficult and hence limits the positional accuracy of the GNSS.

[4]It is recommended trying to hold the roll and pitch of the MLS system, thus also the INS, as close to zero as possible while performing the kinematic alignment of the INS.

[5]Perform this step until the status locks at "GOOD", even when stopping. It is recommended to have patience with this step, since it can take a couple of minutes opposed to Step 1, which is generally achieved in quite less time.

environment start also the laser scanner data acquisition.[6] The full procedure is covered in detail by **Step 1-9** and it is sufficient to follow these in order to successfully gather the data required.

Step 1. Proceed to the area where the gathering of data is to take place. Then, supply the laser scanner with power, Figure B1.7b, and turn on the camera by pressing the “ON/OFF” button depicted in Figure B1.1.

Step 2. Having conducted the brief preparation described in Section B1.1.2, gather INS and camera data using the Python script by means of opening up the prepared command prompt and issuing the command:

```
python <Name of the Python script>.py
```

This will result in the following output:

```
Include camera triggering and logging (Y/N):
```

Enter `Y` and press the enter key to start the capturing of camera images, the logging of INS data and the logging of camera image timestamps. Note that the camera images are captured at a predetermined time interval, which currently is programmed in the Python script.[7]

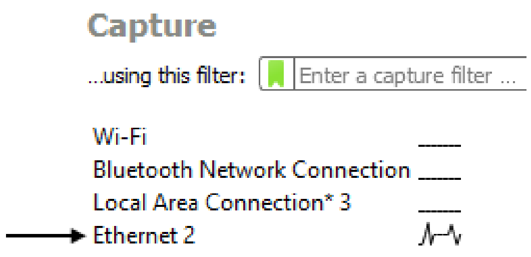
Step 3. While in the clear sky environment, before the laser scanner data acquisition. Approach the forest environment walking in, suggestively, a serpentine pattern for about 30 seconds.[8]

Step 4. When entered the forest environment, what remains now is to log laser scanner data. For this purpose, start by running the WireShark software. In order to log laser scanner data, double-click the field labeled “Ethernet 2” indicated in Figure B1.9a. The laser scanner data acquisition is now in progress, the WireShark GUI will show up similar to what is displayed in Figure B1.9b and the data gathering inside the forest environment may now be proceeded.

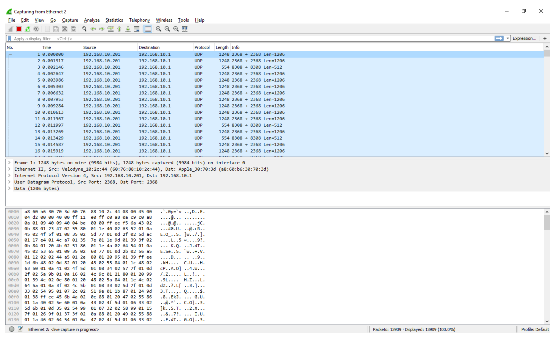
[6]It should be pointed out that for the current state of the post-processing algorithm, that is, the dynamic calibration algorithm. The quality of the results from the post-processed laser scanner data seems to benefit from being carried out on data gathered in an environment of relatively high tree density.

[7]Inside the function `initCamTrig`, the variable `startChar` controls the time interval for camera triggering as: `startChar = 'S'`: 5 seconds, `startChar = 'D'`: 4 seconds, `startChar = 'F'`: 3 seconds and `startChar = 'G'`: 2 seconds.

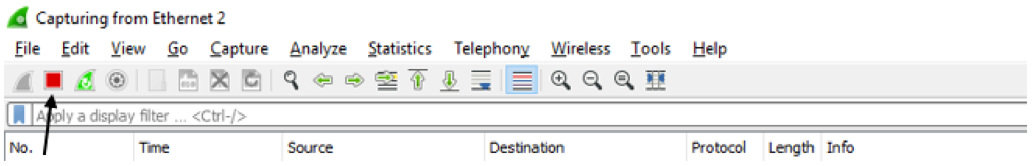
[8]This is carried out prior to entering the forest environment where satellite availability is difficult in order to obtain a proper trajectory.



(a) Start capturing.



(b) WireShark capturing window.


Capturing from Ethernet 2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No.
Time
Source
Destination
Protocol
Length
Info

(c) Stop capturing.

Figure B1.9 – Extracts of the WireShark GUI providing indicates of how to start logging laser scanner data, a display of the WireShark GUI when capturing laser scanner data and how to stop the data acquisition.

Step 5. While gathering data inside the forest environment, perform stops every minute of about ten seconds each trying to hold the velocity as well as the roll, pitch and yaw of the MLS system as close to zero as possible. That is, stop and don't move.[9]

Step 6. When having gathered the desired data approaching the end of the route inside the forest environment, stop the logging of laser scanner data pressing the red "Stop capturing packets"-button in the WireShark GUI as shown in Figure B1.9c. Note! Press the button, but do not quit the WireShark software until having saved the laser scanner data files, which is addressed in **Step 9**.

Step 7. End the route similarly as described in **Step 3** by leaving the forest environment and walk in a serpentine pattern for, as an example, 30 seconds in a clear sky environment to end the data gathering.

Step 8. Stop the logging of INS and camera data by proceeding to the command

[9]The purpose of these stops are to obtain a more accurate trajectory by later using the zero velocity update (ZUPT) feature when post-processing the INS data in Inertial Explorer. For more details consider the User Guidelines for Processing in Inertial Explorer referenced to as Appendix A of Mattias Tjernqvist master's thesis work at the end of Section B2.1.

prompt which were used to run the Python script. Mark the command prompt by a single left-click and issue the key combination: Ctrl+C. Resulting in the output:

```
Logging terminated by issuing the key combination:  ctrl + c
Logging stopped at PC clock:  YYYY-MM-DD hh:mm:ss
```

which indicates that camera triggering along with the logging of camera image timestamps and INS data are now disabled. The corresponding two log files mentioned in Section B1.1.2 can now be found in your current directory.

Step 9. The final step consists of saving the laser scanner data acquired using the WireShark software. In the WireShark GUI depicted in Figure B1.9c, choose File > Save As..., navigate to a proper directory and save the data as a WireShark capture file (.pcapng-file). Also, save the data as a K12-file by again choosing File > Save As..., locate the K12 text file format in the "Save as type" drop-down list and save the data.[10] Note! Before closing the WireShark software make sure that the data is properly saved. That is, by observing the green save-bar that appears in the bottom left corner of the WireShark capturing window, Figure B1.9b, when pressing the "Save" button.

This concludes the data gathering such that the camera may now be turned off and the power supply to the INS and laser scanner may be plugged out.

B2 Post-processing data

B2.1 Inertial Explorer

Inertial Explorer is used for processing of the INS data. To do this, data from close by base stations is needed first. This data can be downloaded from the SWEPOS homepage in the form of RINEX files.

RINEX (Receiver Independent Exchange format), is a standard format which is often used for different types of handling of raw GNSS data. For this case the .d, .n and .g files are needed (.s file is not necessary). Note that there is both daily and hourly SWEPOS data, and that for this case it is recommended to use the hourly. This is because Inertial Explorer probably has to resample the data to, for example, get it in the same sampling frequency as the INS-data. This will take

[10]The WireShark capture file (.pcapng) and the K12 file (.txt) are later utilized by the dynamic calibration algorithm in the post-processing of the laser scanner data.

a much longer time if the daily data is chosen. Also note that the current day is found as the number of day of the year.

The files names have the format: [ssss][ddd][f].[yy][t].zip, where ssss is the station id, ddd the day of the year, f the code for what time period the data file is from (0 means complete day, a-x for hours as shown in Table B2.1), yy is the year and t the file type corresponding to either d, n, g, or s.

Table B2.1 – Conversion table: [f] to hour.

[f]:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
Hour:	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23

When the base station files are downloaded, put them in the same folder as the Inertial Explorer project is going to be in.

Next, create a new project, using the “Project Wizard”:

Step 1. Name the project and save it in the wanted location. Press “Next”.

Step 2. For the GNSS data file browse the .gps-file from the INS. Press “Next”.

Step 3. For the antenna profile, chose **NOV702GG_1.03**. Press “Next”.

Step 4. Let “I would like to add base station data” be marked, to add the SWEPOS data. Press “Next”.

Step 5. Mark “Add station from file” and press “Next”. Now browse the .d file from the RINEX files, for one of the base stations downloaded. Press “Next”.

Step 6. Choose date to be **ETRS89**. Press “Next”.

Step 7. Repeat **Step 5-6** for all base stations. Then let “Finish” be marked and press “Next”. Then press “Finish”.

After these steps the actual post-processing can start. Inside Inertial Explorer, press the icon labeled “Process TC” and follow the instructions in Appendix A of Mattias Tjernqvist master’s thesis work *Backpack-based inertial navigation and LiDAR mapping in forest environments.*

Step 8. After conducted the steps described in Appendix A of *Backpack-based inertial navigation and LiDAR mapping in forest environments.* Make sure to export the project using the “SLU MLS” format.[11] This is performed by pressing the icon labeled “Export Wizard” (blue floppy disc with a yellow star) in Inertial

[11]This will produce the .txt-file of post-processed INS data later utilized by the dynamic calibration tools.

Explorer. Then, browse for a proper location to export the project, choose "SLU MLS" as Profile and simply press "Next" until the option "Finish" appears.

B2.2 Dynamic calibration

The dynamic calibration consists of a set of tools written in MATLAB that uses the laser scanner data obtained from the WireShark software as well as the post-processed INS data. That is, the data obtained in **Step 9** in Section B1.2.2 and in **Step 8** in Section B2.1 respectively. The dynamic calibration tools are used to process INS and laser scanner data in order find tree stem sections in the laser data point cloud and place them in a global coordinate system along with the trajectory obtained by measurements from the INS. Appendix B of Mattias Tjernqvist master's thesis work *Backpack-based inertial navigation and LiDAR mapping in forest environments* provides a complete description of how to carry out the dynamic calibration.

However, a few things to note is that the dynamic calibration tools has some problems which the user of this system should be aware of. For example, the tools cannot handle measurements starting in one hour and ending in another. That is, if one starts gathering data at 13:47 and stops at 14:12. For such a case the 3DUAV Tool will produce an error. Another issue that might occur using the dynamic calibration tools arises due to the fact that the last steps of the dynamic calibration uses laser data corresponding to stem sections of trees to calibrate position. Dynamic calibration of data gathered in areas with few trees can therefore yield some problems and cause the dynamic calibration to fail or produce unsatisfactorily results. Also, note that some of the steps in the dynamic calibration procedure can take some hours for large data sets and is quite computationally heavy.