SvalPoint

A Multi-track Optical Pointing System

Kinga Albert

Master's Thesis

Luleå University of Technology
20. August, 2014

Department of Computer Science, Electrical and Space Engineering
Division of Space Technology, Kiruna

Kinga Albert

SvalPoint: A Multi-track Optical Pointing System

Completed at The University Centre in Svalbard,
The Kjell Henriksen Observatory

Supervisor: Prof. Dr. Fred Sigernes
The University Centre in Svalbard
Longyearbyen, Norway

Examiner: Dr. Jana Mendrok
Luleå University of Technology
Kiruna, Sweden

Master's Thesis

Luleå University of Technology
20. August, 2014

Department of Computer Science, Electrical and Space Engineering
Division of Space Technology, Kiruna

Abstract

The Kjell Henriksen Observatory (KHO) is researching the middle- and upper atmosphere with optical instruments. It is located in a remote region, 15 km away from Longyearbyen in Svalbard, Norway. During the auroral season it is accessible only by special transportation, snowmobile or band wagon and during its approach protection against polar bears is also necessary. Due to these inconveniences a pointing system for the remote control of the instruments was desired.
The purpose of this thesis has been to develop a system optimising operations at KHO, with room for further extensions. SvalPoint offers a solution for multiple instrument pointing through the internet. The system has been defined during the time of the thesis work, incorporating new and previously developed applications into a software package. The different programs interact to define a target and point a number of instruments at it.
In the presentation of SvalPoint the key elements are the design of the software system, the algorithms used for the control and in order to ensure the correct operations the hardware calibration. To create a complete image of the system, in addition both the hardware and the projects incorporated are presented. As to finalise the software development its testing is described along with the assessment of the system's accuracy. Aspects regarding the work process are also presented: definition of goals, task analysis, conclusions and suggestions for further work.

Acknowledgement

Foremost I would like to thank Prof. Dr. Fred Sigernes, for providing me the opportunity to work on this exciting project at The Kjell Henriksen Observatory under his supervision. I would also like to thank him for all the times he has driven me up to the observatory in the belt wagon and for all the support he has provided during my work. I could not have envisioned a better supervisor.

I also appreciate the help of my teachers from Kiruna. I wish to express my deepest gratitudes to Dr. Anita Enmark for being the role model always believing in me, helping me to find my interests and offering advice and guidance on career choices. I am also thankful to Dr. Johnny Ejemalm for offering excellent advice in the choices regarding my thesis project.

I cannot possibly express my gratitude to all the extraordinary people I have met during my studies: they had much influence on me and my view of life. I would like however to thank the most my dear friend Patrik Kärräng for his company on Svalbard, for sharing the excitement in the discovery of the far Arctic, for the fun times, his moral support, and last but not least for being there whenever I needed a second opinion, another point of view or a second pair of hands during my work.

Lastly I would also like to thank my parents for their love and support during my years of education. I am the most grateful to them.

Contents

List of Figures

List of Tables

List of abbreviations

API	Application Programming Interface
DDE	Dynamic Data Exchange
DGPS	Differential GPS
EISCAT	European Incoherent Scatter Scientific Association
ETRS89	European Terrestrial Reference System 1989
FOV	Field of View
GPS	Global Positioning System
IDE	Integrated Development Environment
IP	Internet Protocol
IWRF	Instrument World Reference Frame
KHO	The Kjell Henriksen Observatory
LTU	Luleå University of Technology
PSRF	Pointing System Reference Frame
RAD	Rapid Application Development
SWRF	Sensor World Reference Frame
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNIS	The University Centre in Svalbard
UTM	Universal Transverse Mercator
VCL	Visual Component Library
WGS	World Geodetic System

Chapter 1

Introduction

The Kjell Henriksen Observatory (KHO) is an optical research station focusing on the middle- and upper atmosphere. It is located in Breinosa, at 15 km distance from the centre of Longyearbyen, on Svalbard, Norway. During the past two years there has been an acquisition of optical instruments that are capable of all-sky tracking by the use of 2-axis motorized flat surface mirrors. The observatory has however been yet lacking an efficient system for the control of these instruments. The goal of this thesis is to define and develop a real time control system that is convenient to use for observations, ultimately named SvalPoint.

1.1 Project aims

The optical instruments are placed in the instrumental modules of the observatory building. Each module consists of a dome room, housing the instrument and ensuring a 180° visibility to the sky through the roof of the observatory, and a 1.25 m wide control room, housing a computer dedicated to the instrument. For the conduction of the observations there is a separate operational room of 30 m^2, large enough to house a number of work stations, therefore suitable for a group of people to work comfortably at the same time. (See more information on the KHO building in *Appendix A* and at KHO (n.d.*c*).) One of the problems desired to be solved is the need of personnel in the control rooms. The new system shall make it satisfactory to sit only in the operational room, or alternatively to work from Longyearbyen as the travel to KHO, due to its location, is not possible by car but only by belt wagon or snowmobile during the observational season.

The second problem seeking solution is the acquisition and position determination of targets. An intuitive, easy solution is desired, with multiple options, easily adapted to future needs. There are a number of software applications already developed at The University Centre in Svalbard (UNIS) that are suitable for target location, such as the Sval-X Camera and SvalTrack II (see *Sections 2.2* and *2.3* for more information on the programs). These applications shall be integrated in SvalPoint as user interfaces for finding and locating the target.

An additional goal for the system is to enable simultaneous control of multiple instruments, possibly installed at different locations, pointing them at the same target; hence acquiring the name multi-track optical pointing system. This will enable a large range of observations not possible before at the observatory.

1.2 Task analysis

To meet the goals defined in *Section 1.1* the main problems to be solved in the projects are:

- design of the system as a software package: what are the functionalities of different applications working together, description of the new programs that need to be developed,
- definition of control methods for the instruments: how is the target defined, what is the information needed about it,
- design of the connections between the programs: used communication protocols and the definition of high-layer protocols,
- development of algorithms controlling the instruments for each of the defined pointing methods,
- definition of calibration parameters and methods.

Regarding the instrument control problem, the pointing is done by the operation of two motors. The control of their motion is done by azimuth and elevation angles from the spherical coordinate system. Motor controller units are connected to each pair of motors, therefore their control is possible by command words sent on serial ports, making the extent and of this work to be that of high level, logical control.

The tasks at hand are:

- finding optimal solutions to the problems listed above,
- the implementation of the new programs,
- calibration of the system,
- performance evaluation,
- validation of SvalPoint.

During the practical work related to this thesis the software development and the solving of the problems are done in parallel by exploring possibilities, trying out methods, identifying necessities.

An additional problem to be solved is that of work arrangements due to the remote location of the observatory. A test system using a pan-tilt unit mounted on a tripod, pointing a web-camera has been installed at UNIS, therefore the need for visiting KHO during the implementation and testing of the algorithms is eliminated. See the sketch of the system in Figure 1.1.

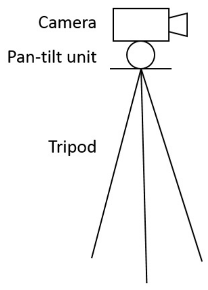


Figure 1.1: Sketch of the test system used for the implementation of the pointing algorithms.

1.3 Outline of Thesis

This thesis presents the software applications that form the SvalPoint system and their interaction. The protocols and algorithms used for the pointing control of the instruments is presented in detail, being the major contribution of this project apart from the definition and creation of the system. Furthermore the calibration processes related to the system are also described in addition to the validation and conclusions about the system.

Chapter 1 serves as an introduction to the SvalPoint system, presenting the problems that shall be solved by it followed by the description of the tasks that need to be carried out during the thesis work.

Chapter 2 presents projects that precede SvalPoint, conducted at UNIS. The first section describes a software solving similar coordinate-system change problems as the ones met in this project. Lastly, the third and fourth sections describe the programs used in the SvalPoint as user interfaces for target acquisition.

In *Chapter 3* the reader will find a description of the hardware used in SvalPoint. Some terminology used further on is defined and the different instruments operated by SvalPoint are shortly summarized.

In the first part of *Chapter 4* the components of the system are described, along with their interaction, communication protocols, including high layer protocols defined during the project. The last section presents what errors are detected in SvalPoint and the actions taken at their appearance.

Chapter 5 starts with the definition of naming conventions used for the description of the algorithms. In the followings the algorithms on the server side are presented, used for the pointing of the instruments.

Chapter 6 describes the necessary calibration processes for the system along with details regarding the calibration done during the practical work.

Chapter 7 is dedicated to the discussion of accuracy. All factors contributing to it are described with numerical calculations where possible.

In *Chapter 8* the tests necessary for the validation of the system are defined. Further on the partial validation of the system is presented together with expected results for the final trial (as in contrast to the present results).

Finally *Chapter 9* summarizes the conclusions of the thesis and presents some suggestions for future work to enhance the existing system.

Chapter 2

Previous projects at UNIS

This chapter presents projects related to the SvalPoint system that has been studied during the work process or used directly in the system. The SvalPoint system is preceded by two standalone software applications, the Sval-X Camera and SvalTrack II that are used as parts of SvalPoint with minimal modifications.
The airborne image mapper software on the other hand is presented and studied as a similar application in some aspects and solutions in it are adapted in the new programs of the SvalPoint.

2.1 Airborne image mapper

The airborne image mapper is a software that projects photos taken by a camera mounted on a helicopter to the Google Earth map as part of a large project, presented in Sigernes et al. (2000). By knowing the location of the helicopter in geodetic coordinates, its attitude and the initial orientation of the camera in reference to the vehicle carrying it, the location of the picture is identified. In other words this program solves the transformation of points defined in a local reference frame, in the image, into points in world reference frame, identifying their place on the map.
The same transformation is a key element of any pointing system for multiple instruments based on target identification by a sensor. The target at first is found and defined in comparison to the sensor, in its local reference frame. To find the independent position, a similar algorithm shall be used as in the airborne image mapper that transforms the location into world reference frame. Later, when the scientific instruments shall be pointed at the target, the location is transformed from the world reference frame into the local reference frame of each instrument apart, doing the same transformations in the opposite direction.

The algorithm used in the Airborne image mapper software is called Direct Georeferencing, the problem being illustrated in Figure 2.1. Point P' is the point identified in the image taken by the camera, this being the four corners of the picture in the program. P is the point sought, that in reality, of which the image is taken through the focal point.

The local level coordinate system, placed in the focal point of the camera is noted by $X_1Y_1Z_1$, while the world reference frame is $X_2Y_2Z_2$. (See Figure 2.1) The position of the point in reality, $\underline{v}_p$, represented in world reference frame ($X_2Y_2Z_2$) is:

$$\underline{v}_p = \underline{v}_o + \lambda \cdot DCM \cdot \underline{v}_i, \tag{2.1}$$

where $\underline{v}_o$ is the position of the local coordinate system's origin in the world coordinate system, λ is a scale factor specific for the lens used relating P to P', DCM is the direct cosine matrix for the rotations that transform a vector in the local reference frame into the world reference frame and $\underline{v}_i$ is the position of point P' in the $X_1Y_1Z_1$ coordinate system.

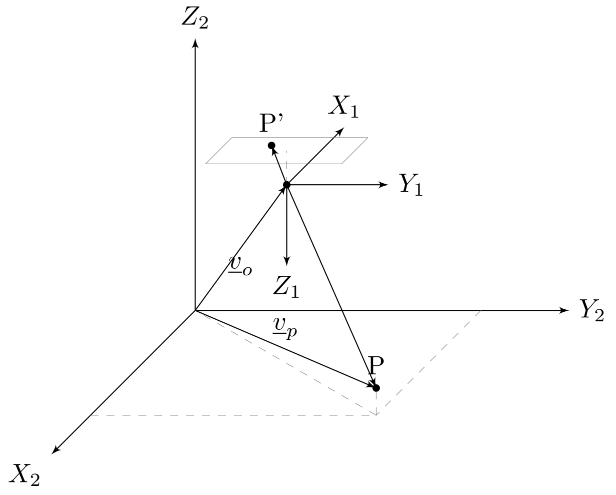


Figure 2.1: Illustration of the problem solved in the airborne image mapper software. Image adapted from Sigernes (2000).

Furthermore expressing the terms of Equation (2.1):

$$\underline{v}_i = \begin{bmatrix} x_i \\ y_i \\ f \end{bmatrix}, \tag{2.2}$$

where x_i and y_i are the two Cartesian coordinates of point P' in the image, f is the focal length of the lens; and

$$DCM = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\sigma) & sin(\sigma) \\ 0 & -sin(\sigma) & cos(\sigma) \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

where σ, θ and ψ are the roll, pitch respectively the yaw angles for the position of the local coordinate system in relation to the world reference frame. See also Muller et al. (2002).
The equations presented are adapted to the SvalPoint system, see *Chapter 5*.

2.2 Sval-X Camera

The Sval-X Camera is a software developed at UNIS, that collects image feed from any camera connected to the computer, given that its driver is installed and it has DirectShow application programming interface (API). The software contains numerous functions for video and image processing, such as: tracking of an object in the video or summing up a user-defined number of frames to get a sharper and more clear image of a static object. The program is used as a possible user interface for control in the SvalPoint system with minimum modification.

One of the functions in the control interface is the video overlay for all-sky cameras. This overlay indicates the cardinal directions in the image, in addition to showing the line of horizon. The necessary attitude and position information for this overlay may come from both a gyroscope - GPS pair, as dynamic values, or can be static calibration inputs. It also calculates the direction to a target, identified by a click on the image, defining the azimuth and elevation angles for that point from the origin of the 'lens coordinate system' (placed in the optical centre of the lens, aligned with the direction of the axis of focus). The values are later transformed into a local level absolute reference frame, placed in the centre of the lens, aligned with north-east and up directions, making the direction of target independent of the orientation of the camera, the only dependency remaining: its geodetic location. These calculations give information about the location of target used for the control of instruments. In the followings the method for calculating the angles in the lens coordinate system is presented.

Considering a fish-eye lens, that all-sky cameras use, Figure 2.2 can be drawn with two coordinate systems associated with the camera: the coordinate system of the lens, noted by $X_1Y_1Z_1$ and the coordinate system of the image plane: $X_2Y_2Z_2$. The focal length of the lens is f. The aim is to define the position of P (point in the real world) based on the position of P' (point in the obtained image), indicated in Figure 2.2. The position of P' can be determined in the image by the values r (the distance between P' and the centre of the image coordinate system) and ψ_2 (angle between axis Y_2 and r). The position of point P shall be defined by the direction of vector ρ, by calculating the azimuth and elevation angles for ρ.

According to Kannala & Brandt (2004) the fish-eye lenses are usually designed to obey one of the following relations for projections:

$$\begin{aligned} r = 2 \cdot f \cdot \tan(\theta/2) \text{ (stereographic projection)},\\ r = f \cdot \theta \text{ (equidistance projection)},\\ r = 2 \cdot f \cdot \sin(\theta/2) \text{ (equisolid angle projection)},\\ r = f \cdot \sin(\theta) \text{ (orthogonal projection)}. \end{aligned} \tag{2.4}$$

All four expressions are implemented in the Sval-X Camera application, subject to user settings. In the followings the relations are to be developed with the use of the most common type of fish-eye lens, with equidistant projection.
Based on Equation (2.4) and Figure 2.2:

$$\begin{aligned} r = r(\theta) = f \cdot \theta,\\ x_2 = r(\theta) \cdot \sin(\omega_2),\\ y_2 = r(\theta) \cdot \cos(\omega_2),\\ z_2 = f. \end{aligned} \tag{2.5}$$

It shall be noted that due to the fact that the lens is circular:

$$\omega_1 = \omega_2 \tag{2.6}$$

From equations 2.5 and 2.6 the value of θ and ω_1 can be calculated directly. These values represent the azimuth and elevation of the point in the real world, based on the image captured by the all-sky camera.

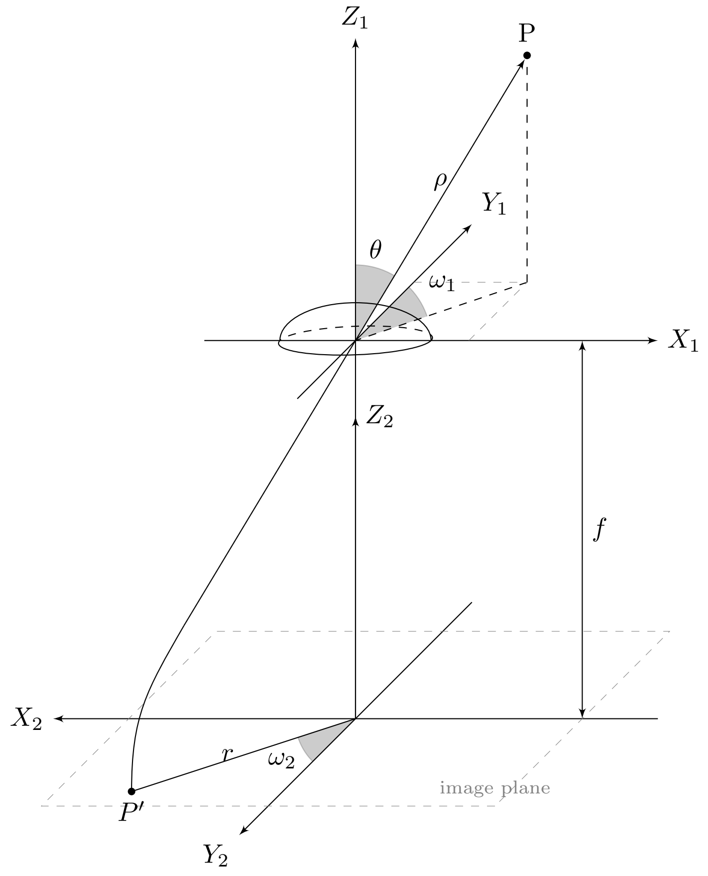
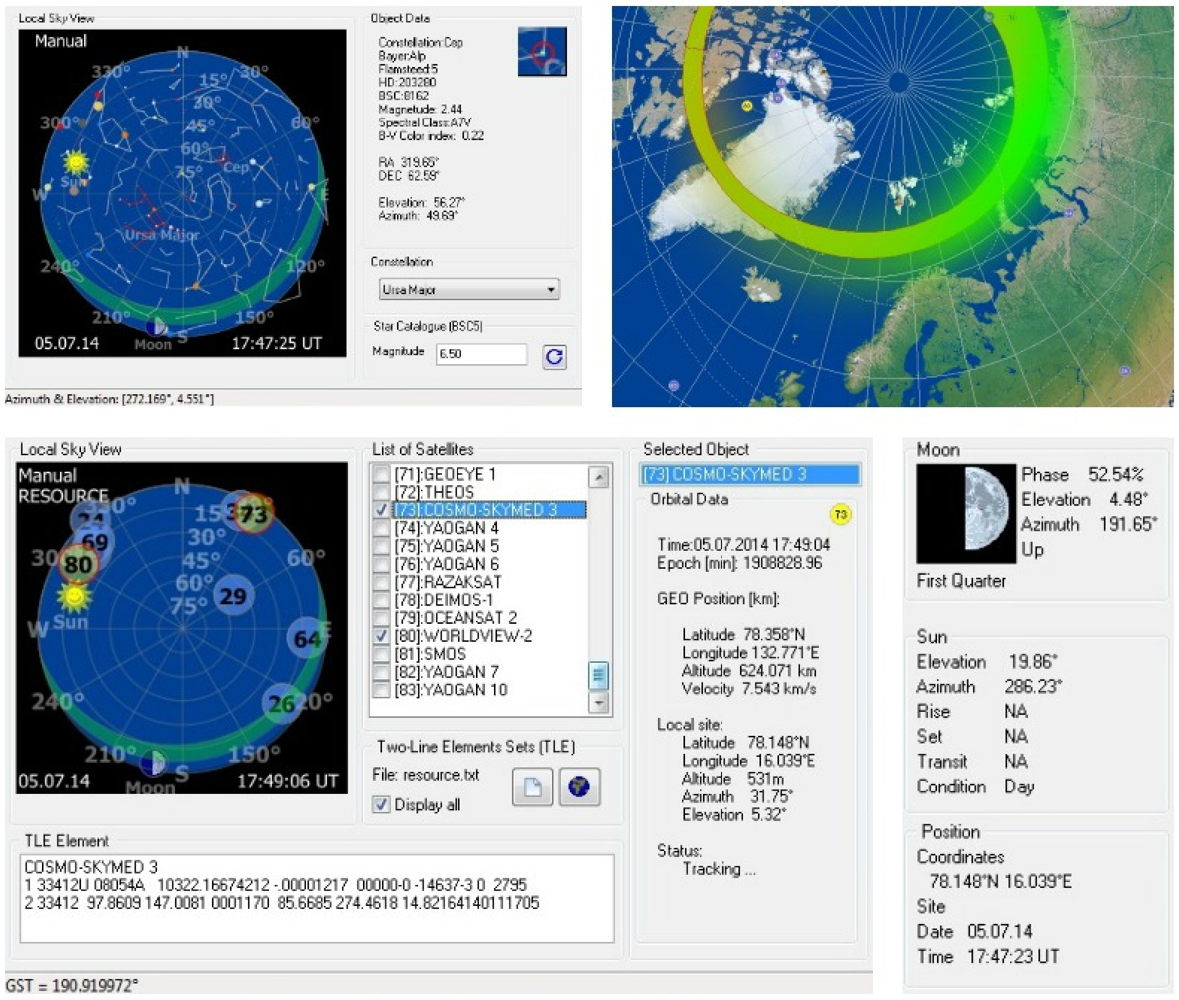


Figure 2.2: All-sky lens model. Figure adapted from Kannala & Brandt (2004).

The robustness of the software is guaranteed by input for the attitude of the instrument. This is used on one hand to be able to place the projection of the point in a local level absolute reference frame, not related to the orientation of the camera, and on the other hand to draw the line of the horizon on the picture captured by the instrument.
The application of the rotational matrix and hence the disconnection of the coordinates from the camera's orientation is done with the use of rotation matrices, see the previous section and related parts in *Chapter 5*. In order to be able to apply them, the vector shall be expressed with Cartesian components, the equations for the transformation are given in *Chapter 5*.

2.3 SvalTrack II

The SvalTrack II software (see publication about it: Sigernes et al. (2011)), developed at UNIS, is a sky observer program. It implements different models for the determination of the auroral oval, information about celestial bodies and satellites. It is a real time software, updating the information about all the previously mentioned in each second, providing azimuth and elevation information about them from the geographic position of the program (subject to user setting). Satellite information is extended with their altitude as well.
This program is used as another possible control interface of the SvalPoint. As it does not use any sensor for target acquisition, it is considered to be a *mathematical control interface* and henceforth it is referred to as such.
See Figure 2.3 for screen captures of the software's graphical user interface.



Object Data
Constellation:Cep
Bayer:Alp
Flamsteed:5
HD:203280
BSC:8162
Magnetude: 2.44
Spectral Class:A7V
B-V Color index: 0.22

RA 319.65°
DEC 62.59°

Elevation: 56.27°
Azimuth: 49.69°

Constellation: Ursa Major

Star Catalogue (BSC5)
Magnitude 6.50

Azimuth & Elevation: [272.169°, 4.551°]

List of Satellites
- [71]:GEOEYE 1
- [72]:THEOS
- ✓ [73]:COSMO-SKYMED 3
- [74]:YAOGAN 4
- [75]:YAOGAN 5
- [76]:YAOGAN 6
- [77]:RAZAKSAT
- [78]:DEIMOS-1
- [79]:OCEANSAT 2
- ✓ [80]:WORLDVIEW-2
- [81]:SMOS
- [82]:YAOGAN 7
- [83]:YAOGAN 10

Two-Line Elements Sets (TLE)
File: resource.txt
✓ Display all

TLE Element
```
COSMO-SKYMED 3
1 33412U 08054A   10322.16674212 -.00001217  00000-0 -14637-3 0  2795
2 33412  97.8609 147.0081 0001170  85.6685 274.4618 14.82164140111705
```

GST = 190.919972°

Selected Object
[73] COSMO-SKYMED 3
Orbital Data
Time:05.07.2014 17:49:04
Epoch [min]: 1908828.96

GEO Position [km]:
Latitude 78.358°N
Longitude 132.771°E
Altitude 624.071 km
Velocity 7.543 km/s

Local site:
Latitude 78.148°N
Longitude 16.039°E
Altitude 531m
Azimuth 31.75°
Elevation 5.32°

Status:
Tracking ...

Moon
Phase 52.54%
Elevation 4.48°
Azimuth 191.65°
Up
First Quarter

Sun
Elevation 19.86°
Azimuth 286.23°
Rise NA
Set NA
Transit NA
Condition Day

Position
Coordinates
78.148°N 16.039°E
Site
Date 05.07.14
Time 17:47:23 UT

Figure 2.3: The graphical user interface of the SvalTrack II software.

Chapter 3

SvalPoint Hardware

The hardware of the system contains two types of units: **computers** and different **instrumentation**. Each instrument has its dedicated computer for its control and the acquisition of data from it. One of the computers that acts as the controller of the SvalPoint system, that can be separate or one connected to an instrument.

The instrumentation falls in two categories from the SvalPoint's point of view: **Instruments** and **Sensors**. The *instruments* are the units controlled through pointing and they collect the scientific data. Though instruments is a generic term, in the case of SvalPoint it refers strictly to the instruments that are the subject to pointing. The data acquired by the instruments do not affect the system.
The *sensors* are instruments that collect information about the target, providing a mean to its identification. The sensors are active part of the SvalPoint system affecting the result of the pointing.
In the following the basic parameters of the instruments and sensors available at KHO at the moment and possible to be used with the current version of SvalPoint system are presented.

3.1 Instruments

3.1.1 Narrow Field of view sCMOS camera

The Narrow Field of view sCMOS camera is designed to study small scale auroral structures. The instrument is composed of two parts: an all-sky scanner and a camera. (See Figure 3.1)
The All-Sky scanner is a Keo SkyScan Mark II, a dual first-surface mirror assembly, with a 360° azimuth and a 90° elevation scanning, put into motion by servo motors. The accuracy of the pointing is ±0.05° with 9°/s azimuth and 27°/s elevation speed. The camera used is an Andor Neo sCMOS, on a -40° vacuum cooled platform, mounted with a Carl Zeiss Planar 85 mm ZF with a relative aperture of f/1.4. (See more at KHO (n.d.*b*).)

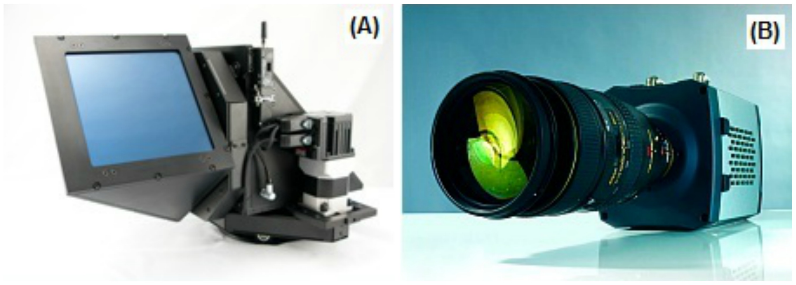


Figure 3.1: The Narrow Field of view sCMOS camera. Panel (A): Keo SkyScan Mark II. Panel (B): Andor Neo sCMOS camera. Image from KHO (n.d.*b*).

3.1.2 Hyperspectral tracker (Fs-Ikea)

The Hyperspectral tracker is a narrow field of view hyperspectral pushbroom imager. The instrument composes of two parts: an all-sky scanner and a spectrograph. (See Figure 3.2)

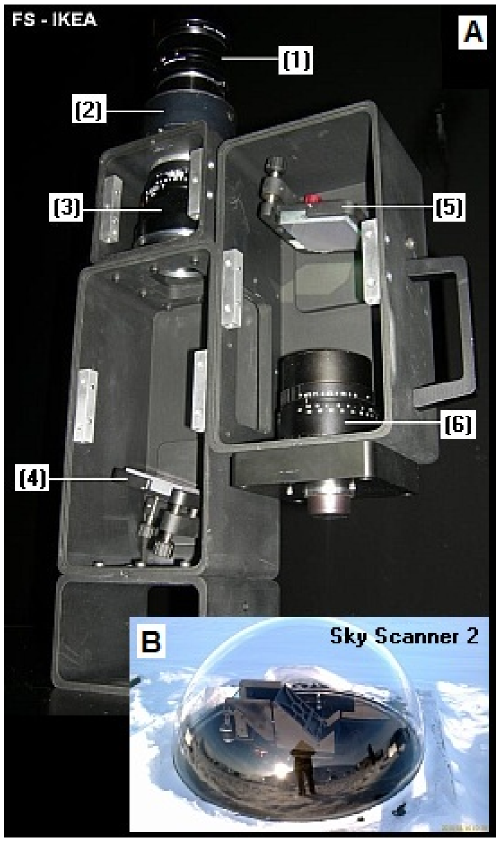


Figure 3.2: The Fs-Ikea spectrograph with its protective covers off. Panel (A): (1) Front lens, (2) Slit housing, (3) Collimator, (4) Flat surface mirror, (5) Reflective grating, and (6) Camera lens. Panel (B): All-Sky scanner. Image from KHO (n.d.*a*).

The spectral range of the instrument is between 420 and 700 nm, with a bandpass of approximately 1 nm. The exposure time for one spectrogram is approximately 1 s. One of the possible lenses to be used is a Carl Zeiss Planar 85mm ZF with the relative aperture f/1.4.
The all-sky scanner is composed of two first surface mirrors, mounted on two stepper motors. The azimuth range of the instrument is 360°, while the zenith angle range is ±90°. The resolution of the motion is 0.0003° with an accuracy of ±0.05°. (See more at KHO (n.d.*a*).)

3.1.3 Any instrument constructed with the PTU D46 platform

The PTU D46 is a pan-tilt unit suitable to point any instrument weighing up to 1.08 kg, in a range ±159° in azimuth and 31° down and 47° up in elevation with resolution of 0.0129°. (Directed Perception 2007)
This unit is used widely at KHO, mainly pointing cameras with different lenses. One example for its use is the **Tracker Camera**, a mount of the Watec LCL-217HS colour and the Watec 902B monochrome camera on a PTU D46 unit. The lenses used are 30 mm Computar TV Lens with a relative aperture f/1.3 (for the colour camera) and a 75 mm Computar lens with f/1.4 (for the monochrome camera). This instrument is used to provide a live feed on the internet at the web page KHO (n.d.*e*). See Figure 3.3 for the image of the Tracker Camera and the PTU D46 unit.

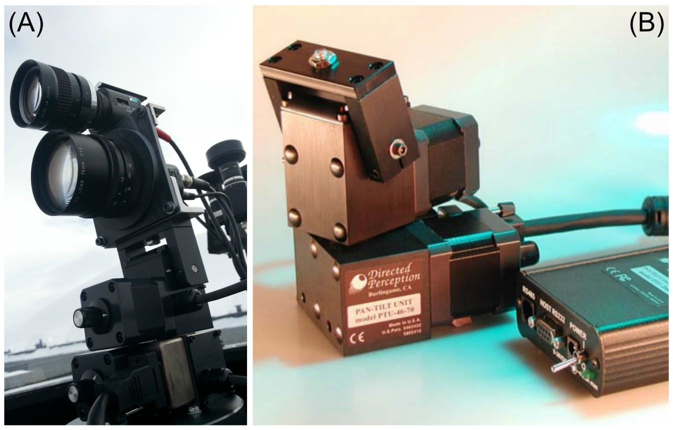


Figure 3.3: The PTU D46 platform. Panel (A): Example for the use of the PTU D46 unit at KHO: the instrument Tracker Camera. The Watec LCL-217HS is mounted on the top, while the Watec 902B is the bottom camera. Panel (B): The PTU D46 unit with its motor controller. Image from Directed Perception (2007).

3.2 Sensors

3.2.1 All-sky camera

The all-sky camera at KHO is constructed of a DCC1240C-HQ C-MOS camera from Thorlabs and a Fujinon FE185C046HA-1 fish-eye lens. (See Figure 3.4.) According to Fujinon Fujifilm (2009) the lens has equidistant projection ($f\theta$ system), a focal length of 1.4 mm and a field of view of 185°[1]. The camera has a resolution of 1280 x 1024 Pixels and a sensitive area of 6.78 mm x 5.43 mm, see Thorlabs (2011).

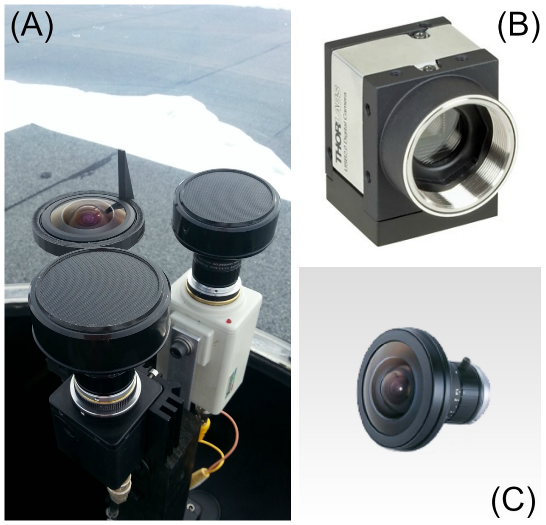


Figure 3.4: The all-sky camera. Panel (A): The all-sky camera at KHO with its lens cover off. The ring around the lens with the spike is a sun-blocker to avoid direct sunlight on the sensor. Panel (B): The DCC1240C-HQ C-MOS camera. Image from Thorlabs (2011). Panel (C): The Fujinon FE185C046HA-1 fish-eye lens. Image from Fujinon Fujifilm (2009).

[1]The field of view has been found to be 194° during calibration.

Chapter 4

Software design

The design of the system as a collection of software applications, the scope of each program and their interaction is one of the main contributions of this thesis. The final structure has been defined well in the development of the software by identifying needs, and trying out different alternatives. The end result of this process is presented in this chapter.

4.1 System definition

The aim of the SvalPoint is to fill in the existing gaps in the current process of instrument operation. The system shall provide easy options for target location and acquisition and link it to the pointing of the instrument, eliminating the necessity for presence in the control room (see *Section 1.1* for the complete description of project aims). The acquisition of data from the target instruments is already automated, solved for each instrument apart, independent of the SvalPoint system.

The system is composed of three parts:

- a **control interface**, which can be any of the two already existing applications developed at KHO: SvalTrack II or Sval-X Camera, with the responsibility of acquiring the target and determining its location;
- **server programs**, dedicated to each instrument and installed on the computers in the control room with the responsibility of pointing the instrument, developed in the duration of this thesis project;
- a **client program** that acts as a data transmitter between the control interface, and the server applications, its responsibility being to transfer the commands and all necessary information to the server and to display the messages sent as feedback from the server; this program is also developed during the work associated to this thesis.

At the definition of the system there has been two options to consider: either to integrate the client application into the control interface programs, or to develop a separate software for it. The decision fell on the latter one for the following reason: the user interface programs implement many functionalities already and there was no room planned for such an extension from the beginning of their development.
The control interface programs are preferred to provide a continuous data stream without waiting for response, sending positions at regular time intervals. The reason for this is that in case there is hold-up for feedback the control interfaces would not be able to update their real time features such as the basic video acquisition or the tracking of an object.
Displaying feedback from the instruments in the client is considered not a vital, however a highly desired feature. Since the locations of the control personnel and the instrument can be kilometres apart it is helpful to know whether the command has been executed or not. One might argue that it is visible from the images captured by the instruments. However there might be exceptions to this assumption, such as in case the images are not streamed over the internet, but saved on the local drive. Moreover the isolation

of problems in control and data acquisition would not be possible.
As what regards the continuous command stream, that is not desired on the server side since reading, interpreting, executing and confirming the execution of commands takes time, this being true even in case the command was the same as the one before. This lead to the very logical design decision: the same commands shall be filtered out from the stream, leaving only the ones that change the pointing of the instrument. As the servers are aimed to be kept general, simple and easy to use, it is the client program that is desired to act as the filter of the data stream.

The functionalities of the servers are therefore defined to include the reading of the commands, the interpretation and calculation of the direction of pointing, and the sending of feedback to the client upon execution. The client application, named SvalCast, is defined to be responsible for the reading of the data from the control interface programs, the filtering of the stream and the sending of the commands to the servers. The control interface from the system's point of view is responsible for providing a stream of pointing directions for the instruments. See Figure 4.1 for an overview of the data exchange between the elements of the system.

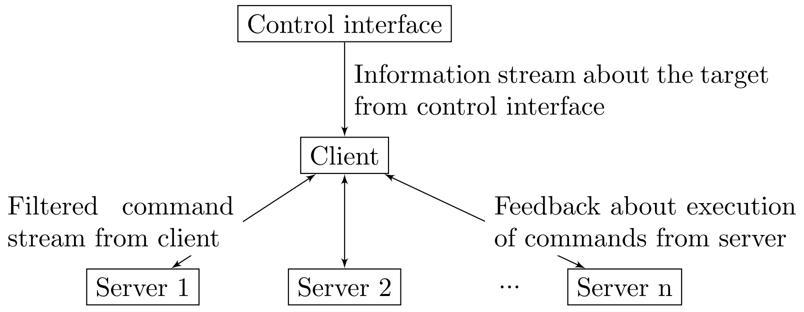

Control interface
Information stream about the target from control interface
Client
Filtered command stream from client
Feedback about execution of commands from server
Server 1
Server 2
···
Server n

Figure 4.1: Overview of the data exchange between the system parts.

The pointing methods for the instruments defined, that all servers must be able to execute, fall into two main categories. The first category is the target-independent commands, with only one command falling in it, the HOME command. This command sends the instrument to its home position defined by the control unit. The second category is pointing at target which can be defined either by geodetic location, the method being called *geo-pointing*, or by azimuth and elevation of target acquired from a sensor. The latter, referenced to as *direction based pointing* henceforth, may have two options: when the range of the target is unknown and the target is considered to be infinitely far away; and when the range can be estimated by estimating its height of target above the reference ellipsoid. The algorithms used for the different methods are described in the *Chapter 5*.
It shall be noted that none of the current control interfaces point the instruments by the use of geodetic coordinates. At the moment this feature is used only in calibration.

The chosen language for the implementation of the applications is Delphi for a number of reasons. Foremost, all other programs associated with the control of the instruments, developed at KHO are written in Delphi making it the first choice due to considerations of program extension, maintenance or future changes.
Another reason for choosing Delphi is that the applications require a graphical user interface that is tedious to develop in many languages, in contrast to Delphi. Delphi has been developed with the aim of providing a rapid application development (RAD) software, based on a set of intuitive and visual tools that help the programmer. The user interfaces are constructed visually, using a mouse rather than by coding, that greatly helps in visual interface development.
Moreover Delphi offers an object oriented programming, based on Pascal, with its own integrated development environment (IDE). The IDE also implements a debugger, offering options for setting breakpoints in the code, step by step running and the display of the values of variables, all helping the program development process greatly.

4.2 Communication between programs

The applications defined in *Section 4.1* shall be connected among each other to be able to transfer the required data. Two types of connections are needed: a connection between the server and client and one between client and control interfaces.

As discussed in *Chapter 1* as well, it is desired to have a remote control over the server programs from any location. This goal demands a connection with Internet Protocol (IP) between the client and the server, one that is suited to transfer short strings. The best way to establish a communication optimal for this task has been identified to be a network socket. Two transport protocols has been considered: the User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). The TCP has been chosen as it is well suited for applications that require high reliability, however transmission time is not critical. In TCP there is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent by the establishment of an enduring connection between computer and remote host. In contrast the UDP is faster but it does not guarantee that the data arrives at all, a major disadvantage over the TCP in this case.
The control interface and client run on the same computer, therefore any connection that ensures communication between two programs in Microsoft Windows is satisfactory as long as the communication is fast enough for handling the data stream. Two options are considered, one of them being the use of the clipboard, the other one the Dynamic Data Exchange. The clipboard is however already in use by the control interface applications for communication with different small programs, and as the protocol provides no mean of targeting the data written to it, this option is discarded, leaving the Dynamic Data Exchange as the mean of communication.

The client-server model followed is presented in Figure 4.2. There are several control interfaces that can command the client to which multiple servers are connected through the internet. It shall be noted, that despite the possibility to start multiple control interfaces at the same time, it is counter-advised. It is not restricted due to possible advantages of this feature, however unintentional use shall be avoided by always closing one control interface before opening another.

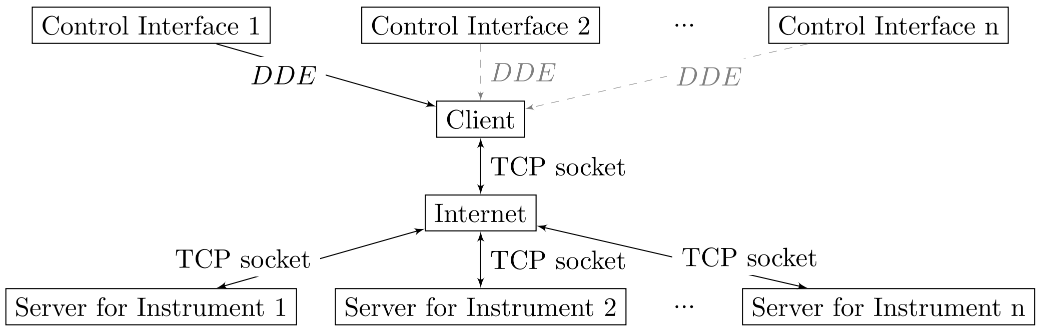


Figure 4.2: The general network model followed.

One of the challenges in the development of the system is to make the programs 'freeze'-proof. At the usage of any system one of the most irritating things is when one of the programs stops responding to commands and must be closed from the Command Manager of Windows. That happens each time an exception is thrown and it is not verified and treated in the code. The connections between programs throw a large number of exceptions (e.g. the client is not responding, the connection cannot be established) therefore a special emphasis is placed on their treatment during the development of the software applications.

The security of the system is mainly ensured by the firewall on the computers, blocking any attempts for connection coming from another computer than one registered in the KHO network. The security related considerations implemented in the servers are the verification of command formats and values.

4.2.1 TCP socket

The TCP socket is the endpoint of inter-process communication flow based on internet protocol between server (sender of data) and client (receiver of the information). The client application establishes point to point virtual connection, also known as TCP session, between two sockets defined by IP address (identifying the host interface) and port number (identifying the application and therefore the socket itself). The server applications create sockets that are 'listening' and responding to session requests arriving from clients.
An internet component suit in Delphi 5 is provided by Indy (Internet Direct) in the form of a Visual Component Library (VCL). The Indy.Sockets includes client and server TCP sockets and it has been used in the implementation of all servers and the client application.
Indy makes the program development very fast, however it does have one downside: it uses blocking socket calls. Blocking socket calls mean that when a reading or writing function is called it does not return until the operation is complete. Due to this it is very easy to program with these functions, however they block the thread of the application, causing the program to 'freeze': it does not respond to any command and it need to be closed from the Task Manager. This would be a major problem if a continuous data stream was sent over the TCP sockets. However, as explained earlier in this chapter this problem has been out-ruled by design. Another implication of the blocking sockets is that during implementation special attention shall be paid so that each message sent over the socket is read on the other side of the communication. (Hower, C. Z. & Indy Pit Crew 2006)

4.2.2 Dynamic Data Exchange

The Dynamic Data Exchange (DDE) is a protocol in Microsoft Windows for fast data transfer between applications that was introduced to exchange the clipboard operations in 1987. Each side of the communication, both the server and the client of the DDE data transfer may start a conversation by transferring data or requesting data from the other. The role of the server and client can be switched during one session and there may be multiple clients in the conversation. (Swan 1998)
In the case of the control interface and client (SvalCast) connection the communication is one-way. SvalCast acts as the client in the DDE, while the control interface is the server. Once the communication is established there is a continuous data stream between the two applications.

4.2.3 SvalPoint high-layer protocols

The SvalPoint system has two different high-layer communication protocols defined: information format for data communication between control interface and client, and command formats for server control.

Control interface - client protocol

The control interface and client protocol communication must contain the following information: the geodetic location of the observation (latitude and longitude in degrees, altitude in metres), the direction of target (azimuth and elevation in degrees) and the altitude of the target in kilometres. A target altitude equal to zero signifies the lack of information on the altitude, in which case the target is considered infinitely far away.
Note: As the geo-pointing is not possible at the moment through the control interfaces no protocol has been defined for it yet.

The control interface sends one string of characters to the client containing information in the following format:
'A '+[latitude]+' B '+[longitude]'+' C '+[altitude]+' D '+[azimuth]+' E '+[elevation]+' Z '+[altitude of target]+' S'.

Client - server protocol

The client-server communication is bidirectional. The client sends commands to the server to control the instruments, while the server sends a feedback to the client about the success of the execution of commands.

The protocol for messages sent by the client to the server is based on command words to distinguish different ways of control. The protocol is based on strings following each other in separate messages. There are five command words. Some are stand-alone, basic commands, wile others must be followed by different values in a given order.
The commands are as follows:

- **HOME** (alternatively: home or Home) - The keyword HOME sends the instrument to its home position, defined as home positions for the motors. This command is not followed by any value.
Note that this position is not equivalent to sending 0 azimuth and elevation values to the instrument.
- **GPS** (alternatively: gps or Gps) - The keyword GPS must be followed by 3 numbers in the following order: geodetic latitude in degrees, geodetic longitude in degrees and altitude above reference ellipsoid in metres. Through this command the target of pointing is defined through its geodetic coordinates.
Note that the convention is negative values for East and South.
Example: GPS 78.15 -16.02 445
- **AZEL** (alternatively: azel, Azel or AzEl) - The keyword AZEL must be followed by five values representing the geodetic location of the sensor (latitude in degrees, longitude in degrees and altitude above reference ellipsoid in metres), and the azimuth and elevation angle for the direction vector to the target in degrees in the order described.
Example: AZEL 78.147686 -16.039011 523.161 12.6 25.1
- **AZELH** (alternatively: azelh, Azelh or AzElH) - The keyword AZELH must be followed by six values. It is the same five values as for the AZEL adding the height of the target above ground in km as the sixth parameter.
Example: AZELH 78.147686 -16.039011 523.161 12.6 25.1 200
- **END** (alternatively: end or End) - Ends the connection between the client and server application. There are no values following this command.

Any feedback from the servers is a single string that forms a message sentence directly displayed in the client application, without any interpretation.

4.2.4 Erroneous commands and feedback to client

To avoid cases in which the client sends commands too fast, the server application always waits for the execution of the motion by the instruments. An inquiry to the motion control unit is sent regarding the current position and the program is blocked in a 'while'-loop until the expected response (indicating that the instrument is in the desired position) is sent on the COM port. As soon as the response is satisfactory a feedback string is sent to the client, containing the 'Command executed.' sentence.
This mechanism relies on the assumption that all commands sent to the instruments are correct and possible to be executed. According to this method once the command is sent on the COM port the program is blocked until the motion is executed and the correct feedback is received. This is a hazard for program 'freeze', in consequence another condition is added to end the 'while'-loop: the expiration of a timer started when the control command was sent to the instrument. It is however not desirable for the while loop to end with the timer expiration (as it takes longer time than the execution of the maximum range of motion by the instrument), therefore it is made sure that all erroneous commands are filtered out.
Erroneous commands might appear in three situations: the command word is not recognized, the values following the command word are not correct or the instrument cannot execute the motion due to limitations in its motion range.

In case the command word is not recognized by the server, no action is taken. For other erroneous cases an appropriate feedback with the words 'Error in command.' is sent to the client. These cases are:

- *Number expected and something else received.* Some of the command words expect to be followed by numbers, for example the GPS word. An example for incorrect command is 'GPS 7k.5 -15,4 500'. Note that the decimal separator is the full stop.
- *Values not in range.* For each number the values are expected to fall in a defined range. The geodetic latitude value must fall in the [-90, 90) range, the longitude into [-180, 180), the altitude of the position and of the target must be positive, the azimuth and elevation must be between [0,360).
- *Resulting angles out of range.* The control units have a maximum and minimum value for the angles that can be set in the server applications. These values are different for the different instrument control units. For example the PTU D46 unit cannot set greater elevation angles than 47°. Therefore a command that results in such an elevation value in its server application returns the 'Error in command.' string to the client.

4.3 The current version of SvalPoint

The current version of the SvalPoint system is composed of the following programs:

- SvalTrack II as control interface;
- Sval-X Camera as control interface;
- SvalCast as client;
- Fs Ikea Tracker as server for the Fs-Ikea instrument;
- Keo Sky Scanner as server for the Narrow Field of view sCMOS Camera instrument;
- PTU46 Tracker as server for the Tracker Camera and the test system.

The client-server model adapted to the current version of SvalPoint is shown in Figure 4.3. It shall be noted that any of the applications may be modified as long as the interface requirements are kept, ensuring the robustness of the system.

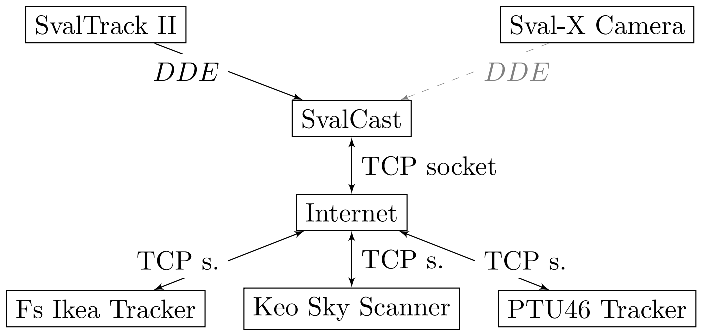


Figure 4.3: The client-server model of the SvalPoint System.

Chapter 5

Pointing algorithms

The pointing algorithms are all implemented on the servers side and they are responsible for the calculation of pointing direction for each instrument based on the information received from the client.
Before discussing the implementation of the different pointing modes, a set of reference frames is to be defined for the system:

- The **Sensor World Reference Frame (SWRF)** is defined as a Local Level Coordinate System associated with the sensor. This is the coordinate system in which the commands from the client application are received in case of direction based pointing command. It is a left handed coordinate system: X axis is aligned with True North (referenced to as N), Y axis with East (referenced to as E) and the Z axis with Up (referenced to as U). Its origin coincides with the centre of the sensor lens, or in case of a mathematical control interface, the origin is the point specified as point of observation in the application.

- Another Local Level Coordinate System, referred to as the **Instrument World Reference Frame (IWRF)** is defined. This reference system is identical with the SWRF in all aspects but one: its origin coincides with the centre of the instrument's lens. A vector in this reference frame shows the direction in World Coordinates where the instrument shall point to acquire an image of the target.

- The **Pointing System Reference Frame (PSRF)** is the reference frame in which the instrument is controlled. The origin of this coordinate system is in the intersection of the two axis along which the instrument is controlled by the motors, translated by a vector $\underline{\mathrm{T}}$, and rotated by the Tait-Bryan angles (see *Appendix B*), compared to the IWRF. The XOZ plane in this reference frame is the plane in which the elevation control motor motion takes place. The YOX Plane is defined by the motion plane of the azimuth control motor. The X axis of the frame is defined by the intersection of the two planes, pointing in the direction of the instrument's optical axis when the control motors are in their home position. The Z axis points upwards, in the plane of the elevation control motor, perpendicular to X. Y axis complements the coordinate system to a left handed reference frame. The control of the instrument is done in the PSRF, therefore all pointing algorithms calculate the target's direction in this coordinate system.

In *Chapter 4* the principles of the two different target-based pointing modes have been presented. The **geo-pointing** mode takes as input the geodetic latitude, longitude and altitude of the target. Knowing the geodetic coordinates of the instrument the vector between the two points is to be calculated. The vector is calculated in IWRF, then transferred to the PSRF.

In the **direction based pointing** mode the target is defined by its direction in the Sensor World Reference Frame. The server application, in addition to the direction, receives the geodetic information about the location of the sensor, making it possible to calculate the direction of pointing for the instrument in Instrument World Reference Frame. The result then is transformed it into the Pointing System Reference Frame.

5.1 Geo-Pointing

The geo-pointing control of the instrument is based on an algorithm that finds the vector between two points defined by their global level (geodetic) coordinates. The input for this type of control is the geodetic coordinates of the target, requiring the coordinates of the instrument to be known. Finding the coordinates for the system is a calibration step, see *Chapter 6*.
The control of the motors is based on azimuth and elevation angles. Therefore all vectors calculated in Cartesian coordinates shall be represented in spherical coordinates for the commands of the system. The steps for the geo-pointing control are the following:

1. Calculate vector between the two given points in IWRF, in Cartesian coordinates.

2. Represent the same vector in PSRF.

3. Transform the representation of the vector from Cartesian coordinates to spherical coordinates.

5.1.1 Calculating the vector between points defined by geodetic coordinates

Two points are considered, noted by P_i and P_j, defined by their global level coordinates (φ_i - ellipsoidal latitude, λ_i - ellipsoidal longitude, h_i - altitude). The aim is to calculate the vector between these two points, noted by $\underline{x}_{ij}$, expressed by the $\underline{n}_i$, $\underline{e}_i$ and $\underline{u}_i$ in the IWRF: a local level reference frame.

The approach used is to first calculate the vector in global level Cartesian coordinates, in the coordinate system XYZ, with its origin in the centre of the Earth. Finding this vector is a subtraction operation once we know the coordinates of the two points in Cartesian representation.
The vector found then is to be represented in the local reference frame, IWRF.

Therefore the first problem that needs to be solved is the finding of the Cartesian coordinates of a point based on the geodetic coordinates (see Figure 5.1).

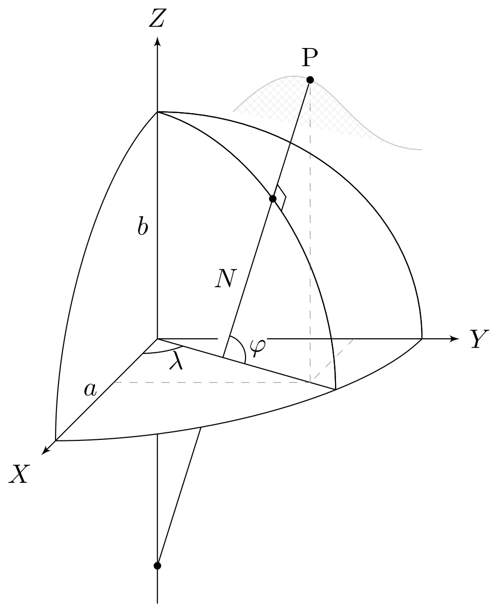


Figure 5.1: Illustration of Cartesian coordinates X, Y, Z and geodetic coordinates φ, λ, h. Figure adapted from Hofmann-Wellenhof et al. (2001).

According to Hofmann-Wellenhof et al. (2001) the relations for these calculations are:

$$\begin{aligned} X &= (N+h)\cdot cos(\varphi)\cdot cos(\lambda), \\ Y &= (N+h)\cdot cos(\varphi)\cdot sin(\lambda), \\ Z &= \left(\frac{b^2}{a^2}N+h\right)\cdot sin(\varphi), \end{aligned} \tag{5.1}$$

where N is the radius of curvature in prime vertical, obtained by the relation:

$$N = \frac{a^2}{\sqrt{a^2cos^2(\varphi)+b^2sin^2(\varphi)}}, \tag{5.2}$$

and a, b are the semi-axes of the ellipsoid.

The parameters of the ellipsoid that models Earth are defined in numerous different standards, such as the World Geodetic System, WGS 84, the European Terrestrial Reference System, ETRS 89 or the North American Datum, NAD 27 and NAD 83, just to mention but a few. These standards use different reference ellipsoids. The coordinates for the target might be acquired with two methods: using maps or using GPS system. The mapping in Europe is done based on the ETRS 89, while GPS systems use WGS 84. The two standards use different reference ellipsoids: the ETRS 89 uses GRS1980, while the WGS 84 has its own reference ellipsoid named WGS 84. Since the coordinates of the target points are most likely to be obtained by the use of a map, ETRS 89 is used for the definition of the ellipsoid.
According to *Geographic information - Spatial referencing by coordinates* (2003) the parameters for a and b defined by the GRS1980 reference ellipsoid are:

$$\begin{aligned} a &= 6378137.0 \text{ m - semimajor axis of ellipsoid}, \\ b &= 6356752.3 \text{ m - semiminor axis of ellipsoid}. \end{aligned} \tag{5.3}$$

$\underline{X}_i$ and $\underline{X}_j$ are defined as the vectors from the centre of Earth to P$_i$ and P$_j$ expressed with Cartesian coordinates. Then the vector between the two points expressed in Cartesian coordinates is $\underline{X}_{\text{ij}} = \underline{X}_i - \underline{X}_j$.

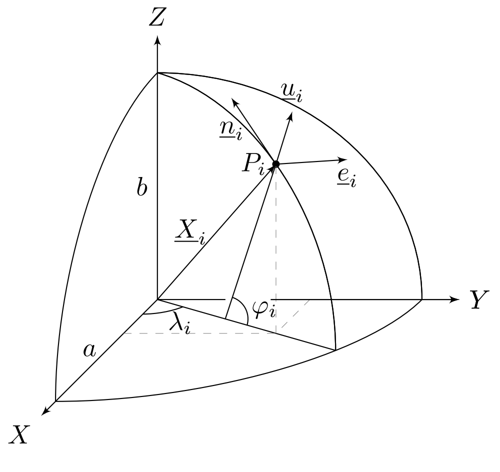


Figure 5.2: Global and local level coordinates. Figure adapted from Hofmann-Wellenhof et al. (2001).

To transfer the vector into the local level reference frame first the axes of the IWRF shall be found in global level Cartesian coordinates (XYZ). The point of origin for IWRF in global level geodetic coordinates is known (the position of instrument). The vectors $\underline{\mathrm{n}}_i$, $\underline{\mathrm{e}}_i$ and $\underline{\mathrm{u}}_i$ need to be found that define the directions true North ($\underline{\mathrm{n}}_i$), East ($\underline{\mathrm{e}}_i$) and Up ($\underline{\mathrm{u}}_i$). (See Figure 5.2)
Illustrated in Figure 5.2 and according to Hofmann-Wellenhof et al. (2001) these vectors are defined by:

$$\underline{n}_i = \begin{bmatrix} -sin(\varphi_i) \cdot cos(\lambda_i) \\ -sin(\varphi_i) \cdot sin(\lambda_i) \\ cos(\varphi_i) \end{bmatrix}, \underline{e}_i = \begin{bmatrix} -sin(\lambda_i) \\ cos(\lambda_i) \\ 0 \end{bmatrix}, \underline{u}_i = \begin{bmatrix} cos(\varphi_i) \cdot cos(\lambda_i) \\ cos(\varphi_i) \cdot sin(\lambda_i) \\ sin(\varphi_i) \end{bmatrix}. \tag{5.4}$$

Finally, according to Hofmann-Wellenhof et al. (2001), the expression for the vector $\underline{\mathrm{x}}_{\mathrm{ij}}$ in IWRF can be found by the scalar multiplication of its Cartesian coordinates with the vectors expressing the axes of the local level reference frame. Therefore the last piece of the series of expressions we have been looking for is:

$$\underline{x}_{\mathrm{ij}} = \begin{bmatrix} n_{\mathrm{ij}} \\ e_{\mathrm{ij}} \\ n_{\mathrm{ij}} \end{bmatrix} = \begin{bmatrix} \underline{n}_i \cdot \underline{X}_{\mathrm{ij}} \\ \underline{e}_i \cdot \underline{X}_{\mathrm{ij}} \\ \underline{u}_i \cdot \underline{X}_{\mathrm{ij}} \end{bmatrix}. \tag{5.5}$$

5.1.2 Transferring vector from IWRF into PSRF

To find a vector in the PSRF ($\underline{\mathrm{x}}_{\mathrm{PSRF}}$) based on its representation in the IWRF ($\underline{\mathrm{x}}_{\mathrm{IWRF}}$) a translation and a rotation shall be performed.

The spatial offset between the centre of the IWRF and PSRF is $\underline{\mathrm{T}}$, as discussed in *Section 4.1* (represented in IWRF). The translation is done by subtraction of the vector components of $\underline{\mathrm{T}}$.

After the translation, the rotation of the coordinate system around the vector shall be performed, with use of rotation matrices for the transformation. The rotation angles are defined as yaw, pitch and roll, rotations around the three axis of the coordinate system. Given that the reference systems are left-handed the following matrices are used for the rotations around axes for the different axes:

$$\begin{aligned} Q_x(\sigma) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\sigma) & -sin(\sigma) \\ 0 & sin(\sigma) & cos(\sigma) \end{bmatrix}, \\ Q_y(\theta) &= \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}, \\ Q_z(\psi) &= \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{5.6}$$

The order of the transformation is roll, pitch, followed by yaw. The corresponding rotation matrices to these angles are Q_x for the roll (σ), Q_y for the pitch (θ) and Q_z for the yaw (ψ). Including the translation with $\underline{\mathrm{T}}$, the final form of the equation for the transformation is:

$$\underline{x}_{\mathrm{PSRF}} = Q_z(\psi) \cdot Q_y(\theta) \cdot Q_x(\sigma) \cdot (\underline{x}_{\mathrm{IWRF}} - \underline{T}) \tag{5.7}$$

5.1.3 Representing a vector in spherical coordinate system

After calculating the vector $\underline{x}_{\mathrm{PSRF}}$, the vector representation shall be transformed into spherical coordinates, as the commands to the pointing system are azimuth and elevation values. See Figure 5.3 for the illustration of the problem. The axes of the PSRF are named $\underline{n}'$, $\underline{u}'$ and $\underline{e}'$ for the optical axis, up and the axis complementing it to a left handed system.

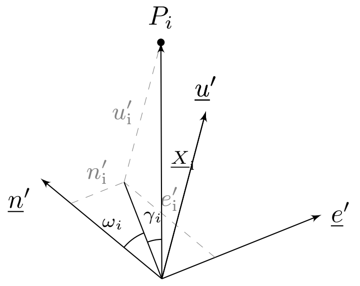


Figure 5.3: Measurement quantities in the PSRF. Adapted from Hofmann-Wellenhof et al. (2001).

As described in Hofmann-Wellenhof et al. (2001), given the components n_i', e_i' and u_i' the azimuth (ω) and elevation (γ) angles can be calculated as follows:

$$\begin{aligned} \omega_\mathrm{i} &= arctan\left(\frac{e_\mathrm{i}'}{n_\mathrm{i}'}\right), \\ \gamma_\mathrm{i} &= \frac{\pi}{2} - arccos\left(\frac{u_\mathrm{i}'}{\sqrt{n_\mathrm{i}'^2 + e_\mathrm{i}'^2 + u_\mathrm{i}'^2}}\right). \end{aligned} \tag{5.8}$$

Analysing the equations in 5.8 furthermore it can be noticed that in case both the $\underline{n}'$ and $\underline{e}'$ components of the vector are negative, the azimuth angle is incorrect: the ratio of e_i' and n_i' is positive, and the calculated angle is the azimuth for the mirrored image of the vector. Similar problem arises when the n_i' is negative, and e_i' is positive. The same angle results as if the signs are the other way around.
It can be concluded that these equations give correct results only in case n_i' is positive. Therefore the cases when the vector has a negative $\underline{n}'$ component are differentiated.
In case of negative n_i' and positive e_i', the equations are:

$$\begin{aligned} \omega_\mathrm{i} &= arctan\left(\frac{e_\mathrm{i}'}{n_\mathrm{i}'}\right) + \pi, \\ \gamma_\mathrm{i} &= \frac{\pi}{2} - arccos\left(\frac{u_\mathrm{i}'}{\sqrt{n_\mathrm{i}'^2 + e_\mathrm{i}'^2 + u_\mathrm{i}'^2}}\right). \end{aligned} \tag{5.9}$$

In case both n_i' and e_i' are negative the equations are:

$$\begin{aligned} \omega_\mathrm{i} &= arctan\left(\frac{e_\mathrm{i}'}{n_\mathrm{i}'}\right) - \pi, \\ \gamma_\mathrm{i} &= \frac{\pi}{2} - arccos\left(\frac{u_\mathrm{i}'}{\sqrt{n_\mathrm{i}'^2 + e_\mathrm{i}'^2 + u_\mathrm{i}'^2}}\right). \end{aligned} \tag{5.10}$$

5.2 Direction based pointing

The direction based pointing is controlling the instrument based on target identified and located by a sensor. The location of the sensor and instrument are generally different, and therefore setting the azimuth and elevation angle from the sensor points the instrument at a different location. This problem is shown in Figures 5.4 and 5.5. The figures are done on large scale to illustrate the effect of the displacement in both azimuth and elevation angles, the system is not intended to be used with such large spatial displacements between instruments and sensor.
One can easily see that an algorithm is needed to determine the direction of pointing for the instrument, based on the direction identified at the sensor. This algorithm shall take into consideration the place of the instrument and the sensor.

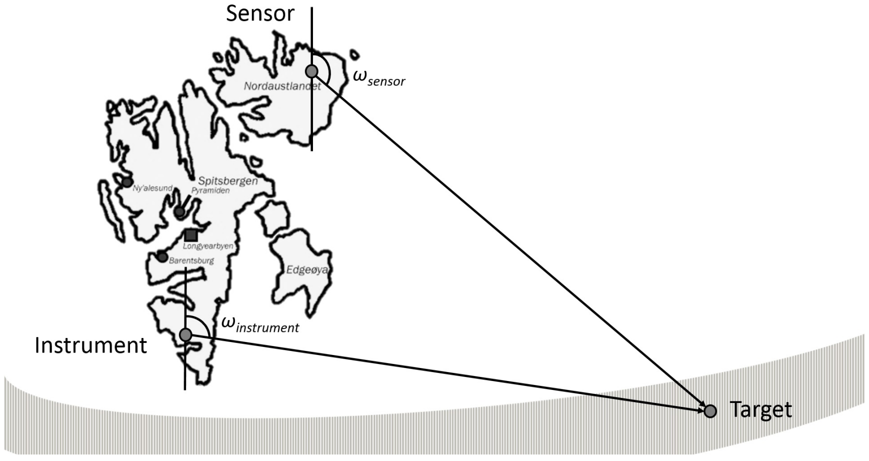


Figure 5.4: Top view illustration of the system, showing effect of spatial displacement between the instrument and sensor: for different locations the azimuth angle (ω) that points towards the same target is different.

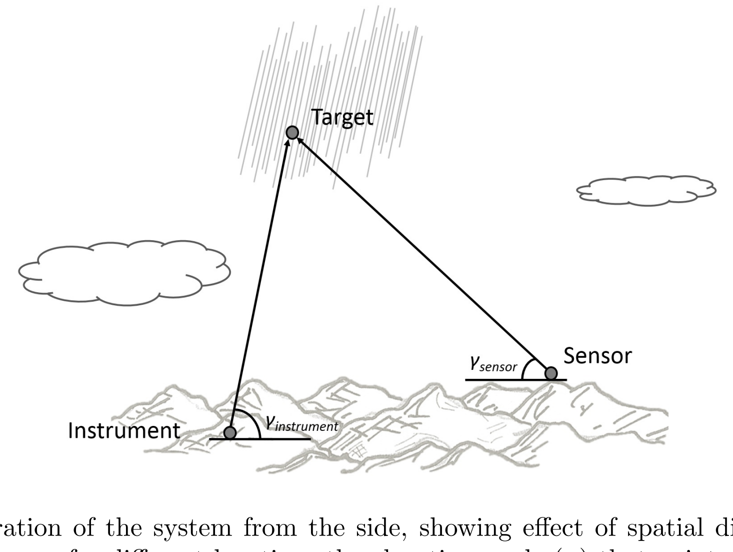


Figure 5.5: Illustration of the system from the side, showing effect of spatial displacement between the instrument and sensor: for different locations the elevation angle (γ) that points towards the same target is different.

Then the steps for the direction based pointing control are the following:

1. Calculate vector between instrument and target in IWRF, based on the vector from the sensor.
2. Represent the same vector in PSRF.
3. Transform the representation of the vector from Cartesian coordinates to spherical coordinates.

Steps 2 and 3 have already been covered in the previous section, describing geo-pointing (see *Sections 5.1.2* and *5.1.3*).

5.2.1 Calculating the vector from the instrument to the target

To calculate the direction of pointing for the instrument we shall know the vector from the sensor to the target and the vector between the sensor and instrument. Knowing them, both represented in the same reference frame, the vector between the instrument and the target can be determined by subtraction. However the in some cases the sensor determines the azimuth and elevation of the target, but not its range. Therefore sometimes only the direction of the vector is known, not its magnitude.

To calculate the direction of the vector, let's consider a unit vector $\underline{v}$, as shown in picture Figure 5.6, defined by the angles azimuth (ω) and elevation (γ). The components x (n_i) ,y (e_i) and z (u_i) are sought, supposing that $|\underline{v}| = 1$.

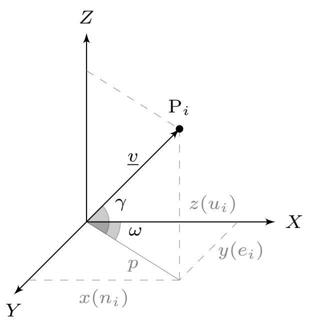


Figure 5.6: Calculating the components (x or n_i, y or e_i, z or u_i) of a vector ($\underline{v}$) defined by azimuth (ω) and elevation (γ).

Considering Figure 5.6 the following equations can be written:

$$\begin{aligned} sin(\gamma) = \frac{z}{v}, cos(\gamma) = \frac{p}{v}, \\ sin(\omega) = \frac{y}{p}, cos(\omega) = \frac{x}{p}, \\ v = 1. \end{aligned} \tag{5.11}$$

Based on the expressions in Equation 5.11, the equations expressing the components of the unit vector are the following:

$$\begin{aligned} n_i = x = cos(\omega) \cdot cos(\gamma), \\ e_i = y = sin(\omega) \cdot cos(\gamma), \\ u_i = z = sin(\gamma). \end{aligned} \tag{5.12}$$

Returning to the problem of the vector magnitude, the two control applications offer different targets.
In the case of the SvalTrack II software the targets are celestial bodies and satellites. In the case of the satellites, based on the orbital data, the range of the vector from instrument to target can be calculated. As

for the celestial bodies, due to their large distance from Earth and due to the relatively very short distance between instrument and sensor, it can be safely assumed that they are infinitely far away, and parallel pointing is sufficient. Therefore the same azimuth and elevation are set on the instrument as given by the sensor. It shall be noted that in this case, to minimize error, the geodetic position of the software shall be set to either the exact same coordinates as the instrument, in the case of one controlled instrument, or in the centre of the formation in case of multiple instruments.
For targets such as the aurora the height determination is more complicated (see *Appendix C*), nevertheless possible. However, as it is very probable to do only a rough estimation of it for simplifying the software usage, *Section 7.1* is dedicated to calculate the effect of a wrong estimation on the pointing accuracy.

Once the height of the target has been identified, the magnitude of the vector pointing from the sensor to the target shall be calculated. The problem is illustrated in Figure 5.7.

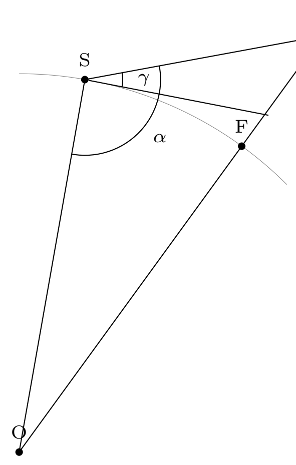


Figure 5.7: Calculating the magnitude of vector based on the height of the target. The lengths are not to scale for the purpose of better illustration of the problem.

In Figure 5.7 ST is the length sought, TF is the height of the target and the angle marked with γ is the elevation angle identified by the sensor. Since the location of the target is not possible to identify, the radius of curvature in prime vertical cannot be calculated. Therefore to minimise the error induced the following calculation has been made: considering that the aurora visible at KHO is the one appearing between the latitudes of 65.961° and 82.136° (values from the program SvalTrack II), the radius of curvature in prime vertical for those values is calculated and their mean value (6385992.9039 m) is used for the lengths of OS and OF. This results in assuming a sphere instead of an ellipsoid that further simplifies the calculations by the relation $\alpha = \gamma + 90°$. The final equation for the vector magnitude therefore is:

$$ST = OS \cdot cos(\alpha) + \sqrt{OT^2 - OS^2 \cdot sin^2(\alpha)} \tag{5.13}$$

Chapter 6

Calibration

For the correct functioning of the system each unit playing active role in SvalPoint shall be calibrated. The following information shall be acquired about each of them:

- geodetic location (latitude, longitude and altitude),
- attitude in Tait-Bryan angles (yaw, pitch and roll).

In addition to these values it must be ensured that the sensors provide correct data, hence all their parts contributing to target location is calibrated. The scientific data from the instruments does not play active part in the pointing system, therefore their calibration is not a part of the SvalPoint calibration process.

6.1 Geodetic positioning of the instruments and sensors

The positioning of the instruments is done by Differential GPS (DGPS) measurements for the centre of the instrument domes. Since all positions in the software are required in geodetic coordinates and the DGPS measurements are performed in UTM system, transformation from it into geodetic coordinates is required. See more information about the UTM in *Appendix D*.

6.1.1 Differential GPS

Differential GPS or DGPS is an enhancement to the Global Positioning System that increases the location accuracy up to 10 cm in contrast to the 15-25 m accuracy of GPS in normal operating conditions. See also Kee & Parkinson (1996).
DGPS is based on computation and correction of range error for each GPS satellite in the view of a single reference station, placed at a known location. In the case of Svalbard these locations are marked by a concrete platform with a metallic bullet in its centre. Then the corrections are transmitted to the nearby users, improving the accuracy of the measurements made by them. It shall be noted however, that as the distance between reference station and user increases, the lines of sight through the ionosphere and troposphere changes, resulting accuracy degradation as the delay to the station and user are different.
See Figure 6.1 for photos of the process on sight.

6.1.2 The measurements

The formulas that transform from Transverse Mercator Projection into geodetic coordinates are named the inverse Gauss-Krüger projection. To perform the transformation a MATLAB® function is used from the toolbox Wasmeier (2006). The function is named utm2ell, denoting transformation from UTM to ellipsoidal coordinates, taking into consideration the irregular sections (see *Appendix D* for more information about the irregular sections). In the input there is an option of entering the standard for the ellipsoid, that is implemented in the same toolbox as data structures. The measurements are done in ETRS89 that uses

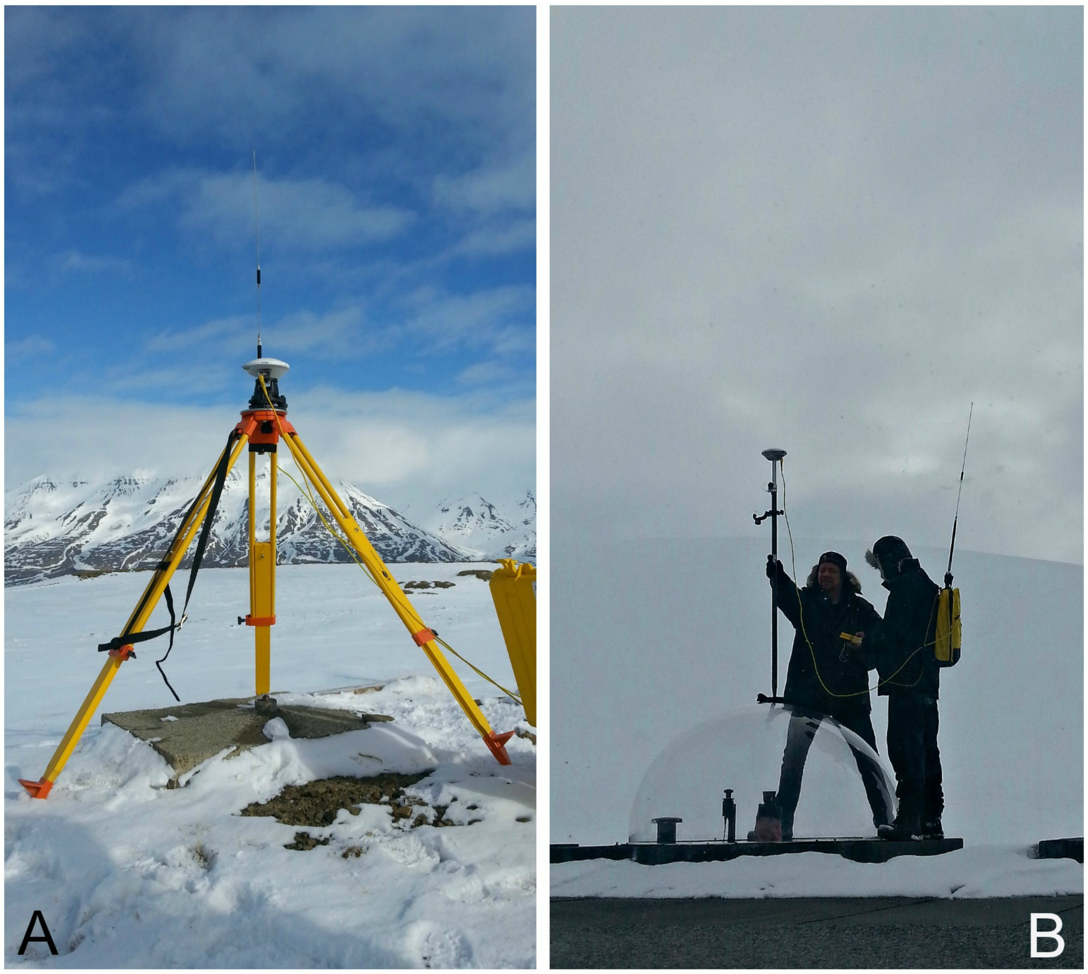


Figure 6.1: Panel A: The DGPS station placed over the marked location in Svalbard, nearby KHO. Panel B: The centre of each dome on the roof of the observatory is measured with the user equipment communicating with the DGPS reference station.

the GRS1980 ellipsoid, therefore that is the standard used for this transformation.
Each dome has been numbered and measured during the process. The measurements of the domes of interest in this project are the ones housing the Narrow Field of view sCMOS camera (referenced as Keo in the followings), the Fs-Ikea and the all-sky camera - PTU D46 unit pair. See the values and their equivalent in Geodetic coordinates in Table 6.1.

Due to delays in the delivery of the measurement data all tests of the system has been done based on data acquired with a hand-held high precision GPS device. Table 6.2 shows the values for each instrument. These measurements have also served as a verification on the transformation results.
In conclusion the measurements are fairly consistent between the DGPS and hand-held GPS methods. The largest error is for the Fs-Ikea instrument East coordinate, however it is fairly straightforward from the pattern of the values, the hand-held device measurement was erroneous.
It shall be noted, that the DGPS measurements may be further refined considering that all domes have the same height, the difference in it coming from the position of the user device that might have been not perfectly vertical at times or might have been placed not exactly in the centre of the domes. Calculating the mean value of all heights the result is 523220 mm (value used in the calculation of error for the measurements with the hand-held GPS device).

Table 6.1: DGPS measurements and their equivalent in geodetic coordinates.

Instrument	**UTM coordinates**			**Geodetic coordinates**		
	North (m)	East (m)	Height (mm)	North (°)	East (°)	Height (mm)
Keo	8675070.729	523835.847	523470	78.14775385	16.03963733	523470
Fs-Ikea	8675068.108	523831.084	523455	78.14773113	16.0394276	523455
PTU D46	8675062.921	523821.632	523161	78.14768616	16.03901141	523161

Table 6.2: Geodetic coordinates of instruments measured with hand-held GPS device.

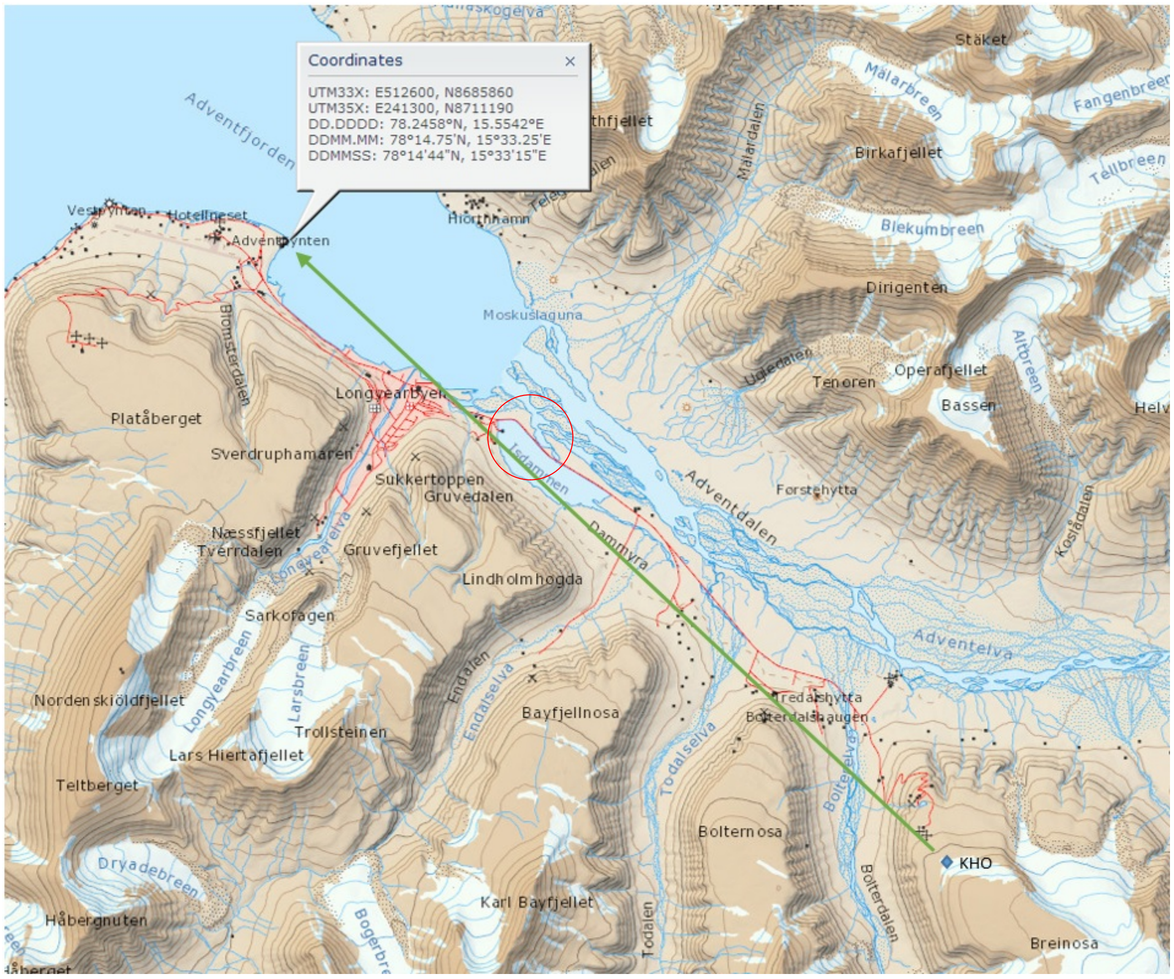
Instrument	**Geodetic coordinates**			**Error**		
	North (°)	East (°)	Height (mm)	North (°)	East (°)	Height (mm)
Keo	78.14776	16.0396075	520000	0.00000615	0.00002983	3220
Fs-Ikea	78.1477	16.0389	520000	0.00003113	0.0005276	3220
PTU D46	78.147686667	16.03900667	520000	0.000000507	0.00000474	3220

6.2 Attitude determination of the instruments and sensors

For the calibration of the pitch and roll angles there is dedicated equipment available, a high precision level. For the yaw calibration however the geo-pointing is used.
It shall be noted early on that as the coordinate systems used are left handed the sign of the angles is as follows: yaw angle is positive from North towards East, pitch angle is positive in the direction from horizontal to down, and roll is positive in the direction from right to left.

6.2.1 Yaw calibration

To define the yaw the offset from North shall be determined. The Geo-pointing is used for this procedure. Assuming that the location of the instrument is correctly determined and that the pitch and roll angles are correctly measured, by pointing at known locations the yaw angle input can be adjusted until the picked points appear in the centre of the image taken by the instrument. The calibration points shall be picked in a way that the features are easily recognised in the image, e.g. mountain tops or distinctive features of the shore. The process shall be done for several points, followed by the calculation of a mean value for the resulting yaw angles. In case the difference between the calibration points is unacceptably large (e.g. in the order of 10^{-1} °), the location, pitch and roll calibrations shall be revisited.
In the followings the calibration for the Tracker Camera is shown. It shall be noted that at the time of the calibration only hand-held GPS measurements have been available about the location of the instruments. The pitch and roll angles have been measured with a digital level as accurately as possible, resulting in the values +0.4°, respectively +0.7°. However, at later stages of testing it has turned out that these values have not been precise enough. Moreover, at the time of the calibration there has been a mistake in the code that limits the precision of the camera movement. These are the possible reasons for offset appearing in pictures. Despite these problems the results for the yaw are fairly correct, as shown later, in *Chapter 8*.
Figure 6.2 shows the target upon which the first calculation has been made. The yaw of the instrument has been initially approximated to be 32°. Pictures of the target have been taken with different settings for initial yaw, with the aim of finding the angle that results in the target being in the centre of the image upon the execution of command. See Figure 6.3 for the images with 26°, 28° and 30° initial yaw settings. The second and third targets are chosen as shown in Figure 6.4 and Figure 6.5. As the targets are in the centre of the pictures, no further adjustments are needed.


Coordinates

UTM33X: E512600, N8685860
UTM35X: E241300, N8711190
DD.DDDD: 78.2458°N, 15.5542°E
DDMM.MM: 78°14.75'N, 15°33.25'E
DDMMSS: 78°14'44"N, 15°33'15"E

Figure 6.2: The first target of pointing for the yaw angle calibration, marked by the red circle. The green arrow marks the direction of pointing. The map used in the image is the property of the Norwegian Polar Institute. (Norwegian Polar Institute n.d.)

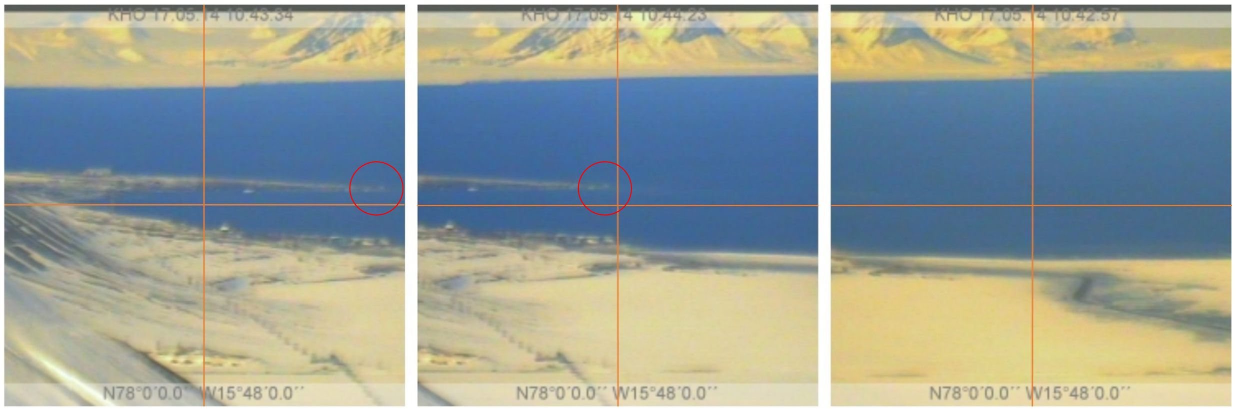


Figure 6.3: Pictures taken of the first target (see fig. 6.2) during the yaw calibration in the order -26°, -28° and -30° for initial yaw. The target is marked with a red circle in each picture it is visible in.

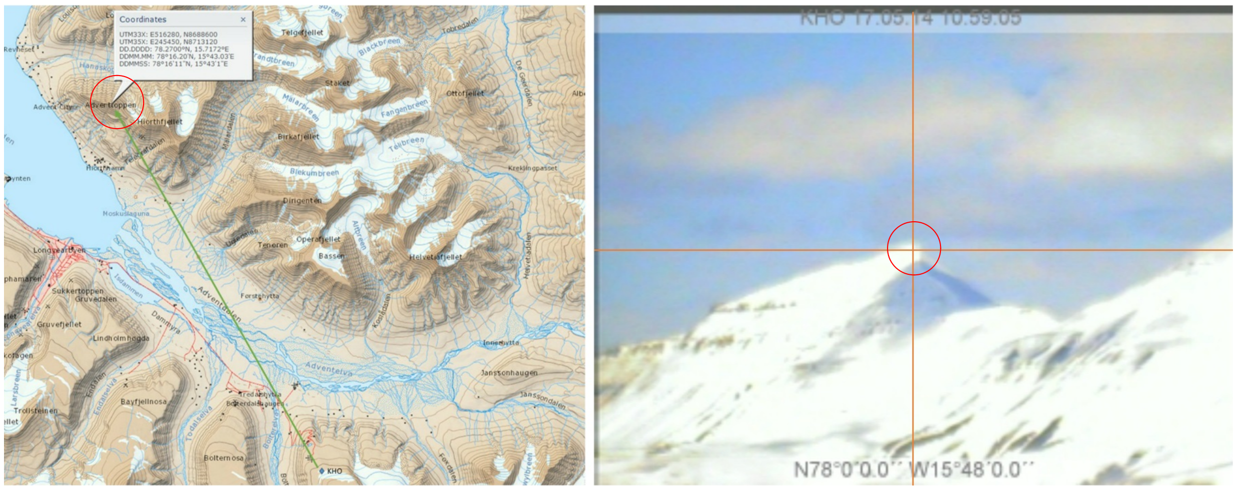

Coordinates
UTM33X: E516280, N8688600
UTM35X: E245450, N8713120
DD.DDDD: 78.2700°N, 15.7172°E
DDMM.MM: 78°16.20'N, 15°43.03'E
DDMMSS: 78°16'11"N, 15°43'1"E

Figure 6.4: The second calibration target: Adventstoppen. The centre of the image taken by the Tracker Camera is marked by the orange cross, while the targets on both images are marked by the red circles. The map used in the image is the property of the Norwegian Polar Institute. (Norwegian Polar Institute n.d.)

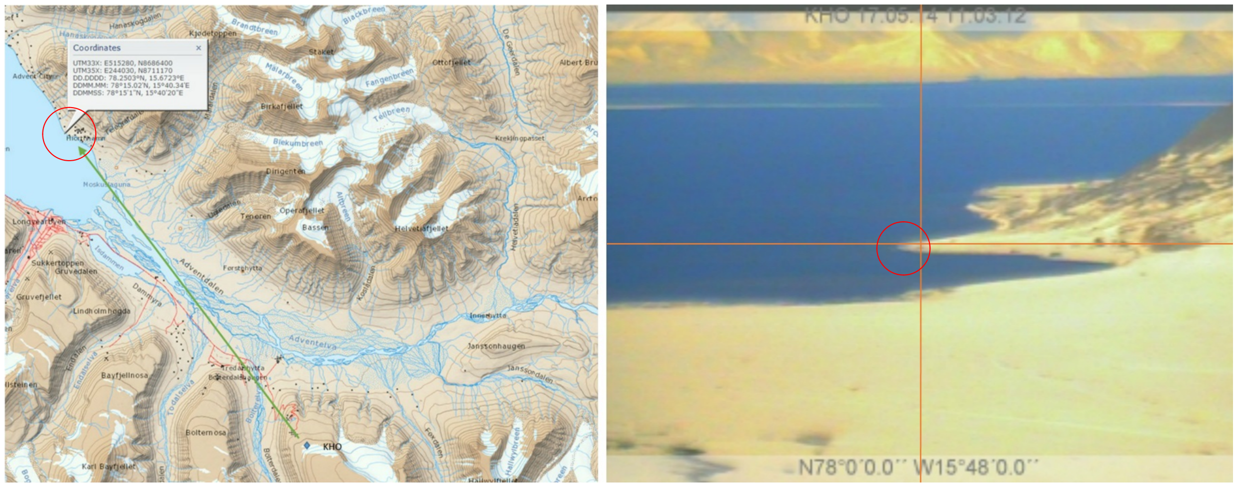

Coordinates
UTM33X: E515280, N8686400
UTM35X: E244030, N8711170
DD.DDDD: 78.2503°N, 15.6723°E
DDMM.MM: 78°15.02'N, 15°40.34'E
DDMMSS: 78°15'1"N, 15°40'20"E

Figure 6.5: The third validation target: Hiorthhamn. The centre of the image taken by the Tracker Camera is marked by the orange cross, while the targets on both images are marked by the red circles. The map used in the image is the property of the Norwegian Polar Institute. (Norwegian Polar Institute n.d.)

6.2.2 Pitch and Roll calibration

The roll and pitch can be measured with the use of a level on the instrument platform. The precision required for the calibration is based on the accuracy of the motion for the instruments, as it shall not be lower. The accuracy for both the Narrow Field of view sCMOS camera and the Fs-Ikea is ±0.05°, while for instruments based on PTU D46 unit no accuracy is available however the resolution of motion is known to be: 0.0129°. Based also on the availability of the digital levels, one with 0.01° resolution is chosen. Furthermore the equipment is factory-calibrated, ensuring the most precise measurements possible.
In case the level is placed on the instrument platform it is assumed that the platform and the instrument are perfectly aligned, inducing possible errors in the calibration. However, as the instrument rests on the platform in the case of the Narrow Field of view sCMOS camera and the Fs-Ikea, this error is minimum. Special attention shall be paid to the alignment of the level with the instrument. The level however has

very convenient flat surfaces on all sides therefore it can be easily aligned with the edges of the platform. In both cases the yaw and pitch error turned out to be 0.02°, indicating that perhaps these angles are induced by the building itself.
In the case of the Tracker Camera and the all-sky camera however the assumption of their alignment with the platform induces errors in the scale of entire degrees. This is due to the fact that the platform is constructed of raw worked wood and metallic brackets fixed with screws. Moreover there is no space for placing the level accurately. Therefore a different approach is taken for the measurements, that similar to the Yaw calibration's.
For the Tracker camera after making an initial measurement of the pitch and roll the camera is pointed at different targets and with trial and error method the angles are adjusted until the target is in the centre of the picture. However, as there are two angles to change and their effect on the pointing is not straightforward, it is not an effective or very precise process. However, as the camera is not used for gathering scientific information, the requirement for the precision of pointing is not that high. As long as the target is in the picture it is considered satisfactory.
In the case of the all-sky camera the same approach is used, only that a calibrated instrument is pointed by the use of the Sval-X Camera control interface. The angles are adjusted until the instrument records images of the target selected in the sensor's picture.

6.3 All-sky lens calibration

The lens of the all-sky camera is a Fujinon FE185C046HA-1 fish-eye lens. According to Fujinon Fujifilm (2009) it has an equidistant projection, a focal length of 1.4 mm and a field of view (FOV) of 185°. However, when calculations performed with these parameters the resulting FOV is over 200°.
In consequence two verifications are done for the lens: verification of the FOV, especially in the light of Krishnan & Nayar (2008) stating that after calibration the FOV for this lens has been found to be 194°, and the equidistant projection equation is suspected to have a scaling factor that needs to be determined (please refer to *Section 2.2* for more details on the equidistant projection equation).

The calibration is done with a small light source mounted on an arm of 1 m, that in turn mounted on a tripod with a panning module showing the degrees of rotation. The alls-sky camera is fixed to a platform and the following coordinate system is considered: the x axis coincides with the optical axis and y is tangent to it in the centre of the lens. The xy plane is parallel with the plane in which the arm moves.
The tripod is placed under the lens, aligning the centre of the rotation of the arm with the centre of the lens as precisely as possible. The spatial offset is measured with the help of a pendulum and measuring tape, giving an offset on the x axis of 0.75 cm, and 0 cm on the y axis. The levelling of the tripod and camera platform has also been adjusted to the best possible, and measured with a high resolution (0.01°) level. The measured offsets of the tripod are 0.57° around the y axis and 0.04° around the x axis, while the same measures for the platform are 0.06° around y axis and 0.2° around x axis. The height of the tripod is adjusted to have the light source shining exactly at the middle of the lens, adjustment made using the pictures captured by the camera.

To calculate the FOV the arm is rotated around to the point when the light source disappears on the horizons. On both horizons the angle value on the panning module is read. From these values not only the FOV but also the angle value at the middle of the lens is determined.
To determine the projection equation for the lens images are recorded with 640×512 pixels resolution while turning the arm from the centre towards the horizons with 10° steps, except for when the source reaches the horizon, that step is only 7.5°. The process is repeated two times to compensate for adjustment errors, and the values are averaged. All angle values measured are adjusted according to the 0.75 cm spatial offset of the rotation point of the tripod from the centre of the lens.

Figure 6.6 shows some of the calibration images for the fish-eye lens. The images start with the light source in the centre (image A), then it moves towards the left horizon. Image B is taken with the light source moved 10° to the left. Images C and D are the last two steps: right before and on the horizon.

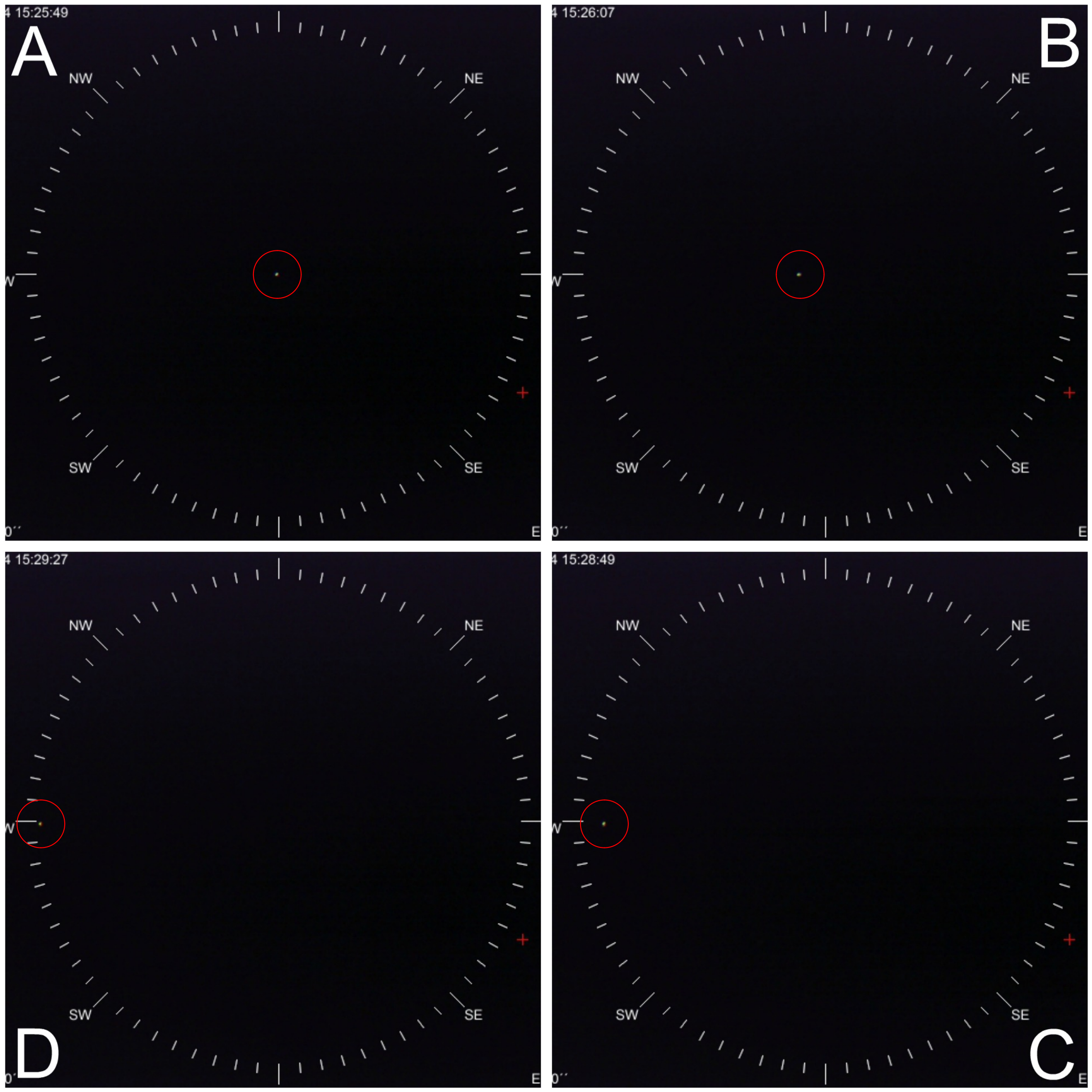


Figure 6.6: Examples for images recorded during the calibration of the fish-eye lens. The light source is marked with a red circle in each picture.

6.3.1 Compensation for spatial displacement

Figure 6.7 is a sketch of the calibration equipment set-up: the centre of the lens is placed in point O, the rotational axis of the tripod is placed in point B. The light source is in point C, moved there from point A. It is clearly visible that the angle displacement read from the panning module of the tripod (α) is different from the angle displacement for the lens (β), the two angles having the relationship $\alpha > \beta$.

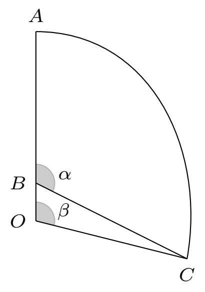


Figure 6.7: Illustration of the effect of a spatial offset of the rotational axis of the arm and the centre of the camera (OB) on axis x of the lens calibration system.

To calculate β we consider the triangle △BOC, as in Figure 6.8. In this triangle the lengths CB and OB are known: CB is the length of the arm and OB is the displacement on axis x in the system. Also, note that from Figure 6.7 angle α is the measured angle on the panning unit of the tripod.

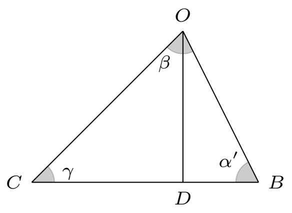


Figure 6.8: Calculating the angle for the fish-eye lens based on the angle measured in the calibration system. △BOC.

The relations in △BOC are:

$$\begin{aligned}
\alpha' &= 180^\circ - \alpha, \\
sin(\alpha') &= \frac{OD}{OB}, \\
cos(\alpha') &= \frac{DB}{OB}, \\
sin(\gamma) &= \frac{OD}{OC}, \\
cos(\gamma) &= \frac{CD}{OC}, \\
CD + DB &= CB, \\
\alpha' + \beta + \gamma &= 180^\circ.
\end{aligned} \tag{6.1}$$

From Equation (6.1) follows

$$\begin{aligned} OD &= OB \cdot sin(\alpha'), \\ DB &= OB \cdot cos(\alpha') \end{aligned} \tag{6.2}$$

and

$$\begin{aligned} CD &= CB - DB, \\ \frac{OD^2}{OC^2} &+ \frac{CD^2}{OC^2} = 1. \end{aligned} \tag{6.3}$$

Based on the trigonometric equation $sin^2(\gamma) + cos^2(\gamma) = 1$:

$$OC = \sqrt{OD^2 + CD^2}. \tag{6.4}$$

Therefore:

$$\begin{aligned} \gamma &= arcsin\left(\frac{OD}{OC}\right), \\ \beta &= 180^\circ - \gamma - \alpha'. \end{aligned} \tag{6.5}$$

6.3.2 The measurements

The measurements, considering the angle adjustments show that the horizon is on 97° from the centre, resulting in a FOV of 194°, the same as the value found in Krishnan & Nayar (2008).

To calculate the projection equation the pictures taken with 10° steps has been analysed. On the pictures the light source represents approximately 3 pixels in width and 3 pixels in height. Therefore the central pixel has been used to determine the place of the light source, in a 2-axis Cartesian coordinate system (x and y) on the picture. Knowing the centre of the picture the distance of the light source from it in pixels can be determined.
The pixel size of the DDC1240C-HQ camera used in the mount is $5.3\mu m \times 5.3\mu m$ (see Thorlabs (2011)), and knowing that the full resolution of the camera is 1280×1024, while the recorded image is in 640×512 resolution, the distance from the centre can be determined.
Assuming that the projection is equidistant and only a scaling factor is unknown then the following equation can be written:

$$\begin{aligned} r &= f_e \cdot \theta, \\ f_e &= a \cdot f, \end{aligned} \tag{6.6}$$

where r is the distance of point from the centre of image, f_e is the effective focal length, f is the focal length, θ is the angle between optical axis and the incoming ray and a is the scaling factor.

Using the angles measured, then adjusted, and the distance of the source determined from the images an f_e for each angle can be calculated. The largest difference between two f_e values is 0.0157 mm, therefore it can be concluded that the projection is indeed equidistant.
Calculating the mean value for the effective focal length gives $f_e = 1.42463427mm$, resulting in $a = 1.017595907$.

Chapter 7

The accuracy of pointing

The system is planned to be used for various observations, for various targets, the main focus being aurora. Due to the very dynamic nature of the phenomenon and the large variety of its shape, size and location in the sky, also due to the differing nature of scientific observations there could not be a well defined accuracy requirement from the start, ultimately aiming for having the target in the field of view of the instrument. The system involves many processes that contain both human factor and a multitude of possibly standing cases contributing to the accuracy of pointing, noting that errors may propagate ultimately and they shall be reduced as much as possible in all cases. Therefore the problem is discussed in terms of absolute limitations and conditions that change the final accuracy.

The first and most important factor in the accuracy is the resolution of the pointing control motors. This limits the finery of the motion, and therefore sets an absolute technical limit on the accuracy of the pointing. The values for the different instruments are to be found in *Chapter 3*.
On the other hand, since the instruments are not sensing with a single pixel, but have a field of view, some displacement may be permitted, considering the size of the target and nature of observation, as the image acquired might still contain the target. One can draw the conclusion that chance for the success of the pointing decreases with the increase in the size of the target.

The second large factor limiting the accuracy to be considered is the pointing algorithms of the instruments. Three major limitations shall be considered: calibration inaccuracies, the accuracy in the target location and the accuracy of target identification.
The inaccuracy in the initial instrument attitude (coming from calibration errors) affects all modes of instrument control. When transforming the vector from its representation in the Instrument World Reference Frame (IWRF) to a representation in the Pointing System Reference Frame (PSRF), as the PSRF is calculated by the yaw, pitch and roll angles, the direction will be offset with as much as the error in these angles resulting in effects and further considerations identical to the motion resolution problem, already discussed.
In the case of geo-pointing, considering that the positioning of the instrument and target is exact, the pointing is more accurate than the resolution of the motors, limited only by the accuracy of real number representation in the computer. The source of inaccuracy lies in the exact positioning of both instrument and target. Considering inaccurate acquisition of location data for target and/or instrument, the direction of pointing will experience larger error at shorter instrument-target distances.
In the case of direction based pointing location calibration errors for the sensor and instrument introduce a decrease in accuracy that is also more visible at close targets. This method has however further limitations to consider. In the case of the Sval-X Camera, when an all-sky camera is used for target location and no information can be acquired about the range of the target, the correctness of the height estimation limits the accuracy. See more about it and its effects in *Subsection 7.1.* Furthermore, calculating the range based on the height of target is done with an estimated curvature in prime vertical of the ellipsoid that reduces the accuracy of the calculations. The effect of this is the smallest when the target is at 90° elevation angle.

Errors in target identification may occur in the case of the Sval-X Camera, when an all-sky camera is used for sensor. The image often reveals less details close to the horizon, increasing the chance for wrong identification of target. Considering the aurora, supposing that its horizontal feature is of interest, its details will be more visible close to zenith than at low elevations.

7.1 Calculation of pointing accuracy in relation to error in the height of target

As previously discussed in *Chapter 5* and in this chapter the determination of the height of target is necessary in case of direction based pointing. To be able to estimate and visualize its importance and effect on pointing accuracy a MATLAB® program has been written, consisting of two functions.

The first function gives the error in azimuth and elevation angles. It takes as inputs the vector direction between sensor and instrument, the vector direction between target and sensor and the height of the target above the reference ellipsoid (an array of different heights). It has hard-coded the height set in the SvalPoint for the target and the height of the sensor above the reference ellipsoid. The function calculates the magnitude for the sensor-target vector for each height input, followed by the calculation of azimuth and elevation angles for the instrument-target vector for each height. Then, subtracting from these values the azimuth and elevation calculated for the height of target set in the SvalPoint the error in pointing is achieved.
The second function generates different possibilities and plots the graphs. The plots shall show the function between the angle error (separately for azimuth and elevation), the distance between the sensor and instrument and the height of target above reference ellipsoid. The hard-coded parameters in this function are the limits for the height of target, the limits for the distance between sensor and instrument, the direction vectors both for the sensor-instrument and sensor-target. In the case of the latter one all possibilities are investigated by generating all unit vectors on half a sphere with the origin in (0,0,0). For this the following equations are used:

$$\begin{aligned} x &= cos(\theta) \cdot sin(\varphi), \\ y &= sin(\theta) \cdot sin(\varphi), \\ z &= cos(\varphi), \\ \text{where: } 0 \leq \theta \leq \frac{\pi}{2} &\text{ and } 0 \leq \varphi \leq \pi. \end{aligned} \tag{7.1}$$

Out of the angle errors for all possible target directions the maximum error is chosen.

Both most probable and worst case scenarios are investigated for being able to estimate the affect of error on system accuracy. The height of sensor is set to be 520 m, the height of the KHO station. The distance between sensor and instrument varies between 0 to 22.5 m, these distances being the most realistic estimations for future situations: the span of 10 adjacent rooms. The height of the target is calculated for the values 70 and 550 km, based on Strømer (1955), presenting the height distribution of a large number of auroral observations between 1911-1944 (See Figure 7.1). The height measurements have been done by triangulation.

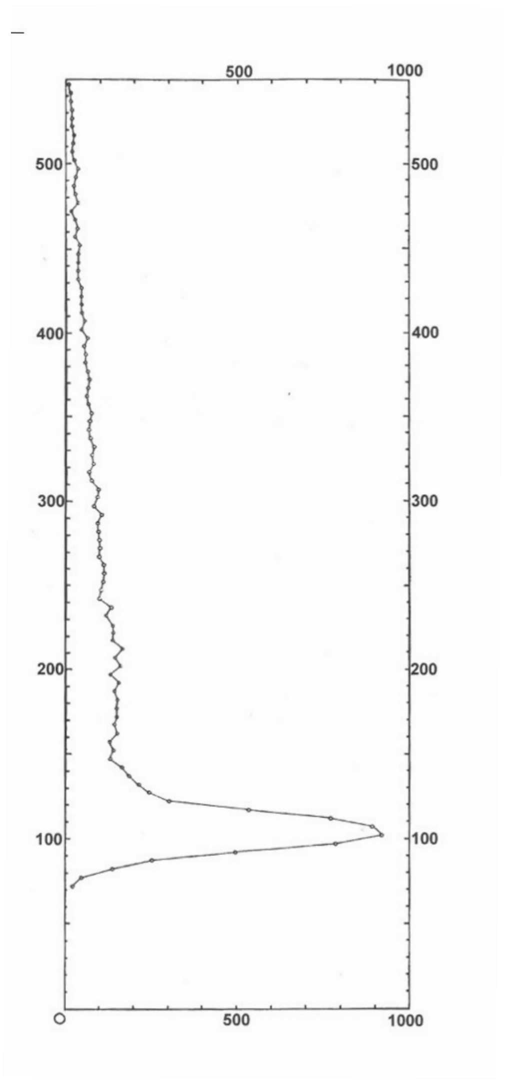


Figure 7.1: The distribution of auroral heights in Strømer (1955). The plot shows the number of events on axis x versus the height of the occurrences. Image adapted from Brekke (2012).

First the worst case calculations are presented, with the sensor-instrument direction vectors being: (1,0,0), (0,1,0) and (0,0,1) in North-East-Up coordinates. See Figure 7.3 and Figure 7.2 for the results. Based on these images one can see that the best configuration for sensor and instrument would be to be atop of each-other (Figure 7.3), however that is physically impossible due to view-field problems. One can draw the conclusion that if the sensors are at the same level further errors in elevation are eliminated. Furthermore, the placement of sensors in the North-East plane do not make any difference. The maximum error in angles are 0.1179° for azimuth and 0.0024° for elevation. These are worst case scenarios, without any attempt of determining the height of the aurora, having a constant 110 km set (the most frequent height of aurora according to Kaila (1987)).

Considering a more realistic scenario, the sensor and instrument are placed at $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0)$ direction. Note that the North-East coordinates do not make difference, as concluded earlier, however they are levelled for the best possible results. The height of target is determined to be 200 km with 50 km uncertainty. For the graph in Figure 7.4 the height of the target set in the SvalPoint is 200 km, resulting in the errors 0.024° in azimuth and $4.8035 \cdot e^{-4}$ ° in elevation. As one may see, the error in pointing if the target is lower than the estimated height is larger, therefore one may conclude that performance may be maximized if a lower value is entered for the height in the SvalPoint system, to balance the errors. It has been found that in this case the value 187 km is the height of target value that results in the lowest error in pointing for the 150 - 250 km range: 0.0194° in azimuth and 3.8792·e^{-}4° in elevation. See Figure 7.5.

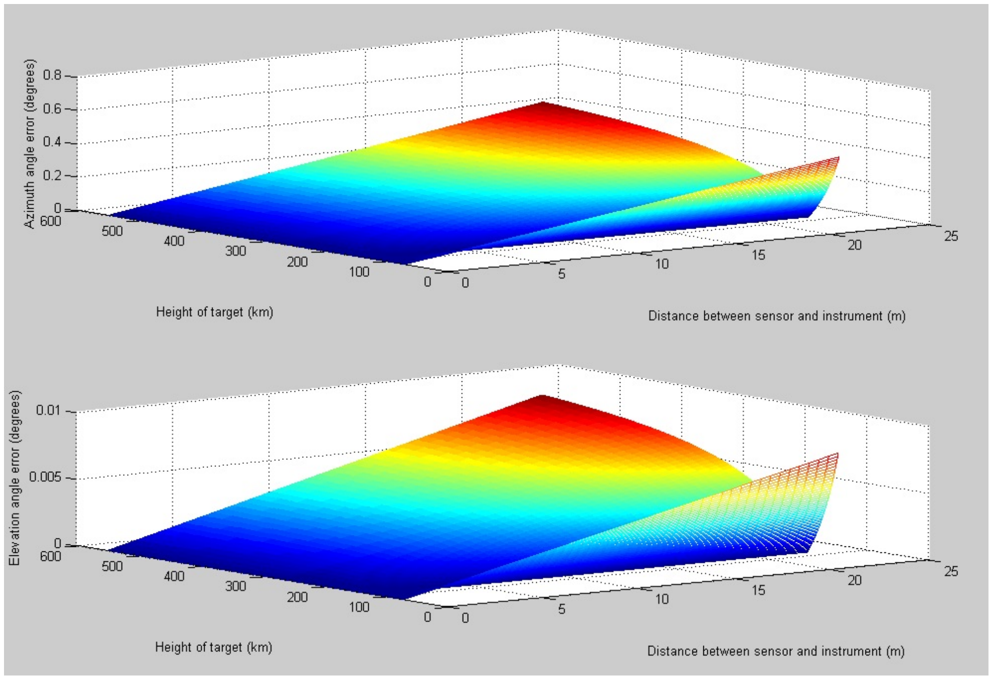


Figure 7.2: The pointing error in the function of height of target and distance between sensor and instrument if the direction of instrument is the (1,0,0) or (0,1,0) unit vector from sensor and the set value in SvalPoint for the height of target is 110 km. The maximum value for azimuth and elevation errors are 0.4714° and 0.0094°. Graph plotted with MATLAB®.

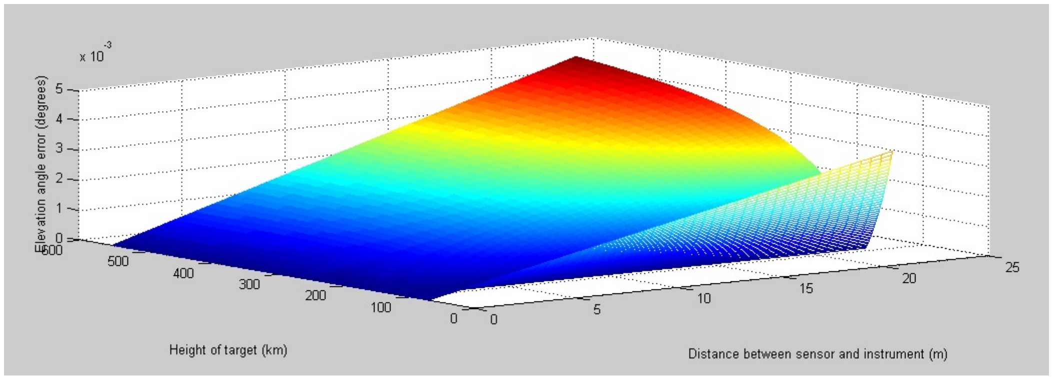


Figure 7.3: The pointing error in the function of height of target and distance between sensor and instrument if the direction of instrument is the (0,0,1) unit vector from sensor and the set value in SvalPoint for the height of target is 110 km. The maximum value for azimuth and elevation errors are 2.5444·e^{-14} ° (approximately 0°) and 0.0047°. Graph plotted with MATLAB®.

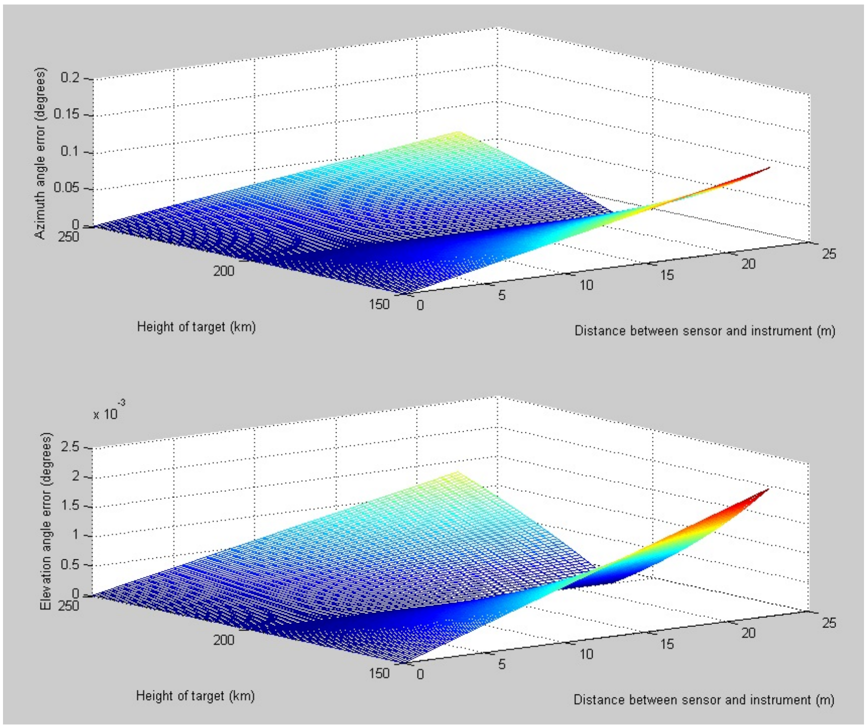


Figure 7.4: The pointing error in the function of height of target and distance between sensor and instrument with the height of target 200 km, estimated with 50 km uncertainty. Height set in SvalPoint: 200 km. The maximum value for azimuth and elevation errors are 0.1081° and 0.0022°. Graph plotted with MATLAB®.

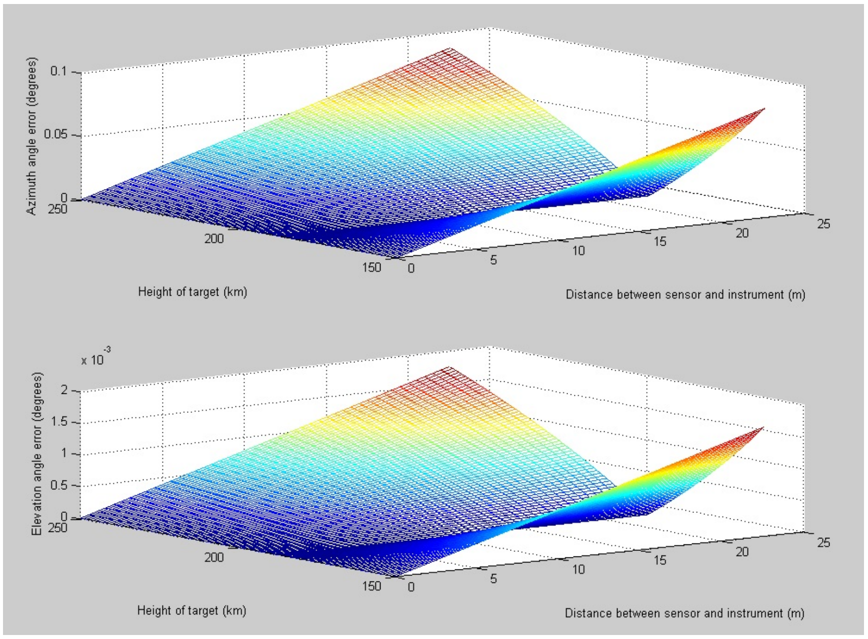


Figure 7.5: The pointing error in the function of height of target and distance between sensor and instrument with the height of target 200 km, estimated with 50 km uncertainty. Height set in SvalPoint: 187 km. The maximum value for azimuth and elevation errors are 0.0873° and 0.0017°. Graph plotted with MATLAB®.

Chapter 8

Validation

The complete validation of the system includes the testing of the connections and the testing of the pointing itself. It is done in order to ensure that the system can be used to perform the tasks that it was designed for, fulfilling all aims of the project.
For validating that the SvalPoint system is working from the connection's point of view the following tests shall be done:

1. connect to all combinations of the possible servers (e.g. connect to Fs-Ikea Tracker, and Keo Sky Scanner but not the PTU46 Tracker), making sure that they respond correctly,
2. try connection with the firewall blocking all attempts to connect, then from a computer that is not in the exception list, making sure that the connection is declined in both cases,
3. connect and disconnect repeatedly with the client application, making sure that new connection is possible after each disconnection,
4. try connecting the client to servers that are not connected to the internet, making sure that the failure to connect is properly indicated in the client,
5. and finally attempt connection to several servers, some of them connected to the internet, others not, making sure that the ones connected react to the commands, while the disconnected ones indicate the lack of connection.

Each control possibility with all interfaces shall be validated. During the tests all instruments shall be pointed, and the data collected shall be analysed to confirm that the target was indeed acquired.

1. pointing at geographical features identified in a map, using geo-pointing, making sure that each target is found in the centre of the pictures,
2. pointing at targets identified in the sensor, covering both targets close to the horizon and at zenith position, making sure that all instruments acquire the same target,
3. pointing at auroral structures, making sure that the acquired target is consistent for all instruments, but also assessing the speed of the operation of SvalPoint versus the dynamics of the aurora,
4. tracking the moon with all instruments, making sure that the target stays in the field of view of the instruments after an extended time,
5. tracking of stars with instruments, verifying that the targets are not lost after extended time,
6. tracking of satellites shall be done on a selection that are visible in the night sky and it shall be verified that the instruments keep the satellite in their field of view during its visibility on the horizon.

SvalPoint has been validated entirely from the connection's point of view and all tests have passed. The validation of the pointing on the other hand has been performed only partially. This is due to both the light conditions in the time frame in which the thesis work took place and for a technical failure on sight that has not been solved in the time limit. One of the motor controllers on the Fs-Ikea instrument have failed, therefore its pointing could not be tested; textual output about the control angles has been verified in its case. The other reason for the partial validation is that dark sky returns in September (a time until which this project does not extend), therefore stars, satellites and the aurora are not visible. Also, at the time of the validation there was no suitable time for testing on the moon as it has always been very close to the horizon. Until the present moment test 1 and a part of test 2 has been conducted.

Test 1. Due to the mathematical nature of the method, the geo-pointing is the most accurate pointing and the test associated to it is rather a validation of the calibration values than of the method (the geo-pointing is also actively used in the calibration, see *Chapter 6*). Actual test on the method were done only in order to make sure that all formulas are correct in the program. This has been done by approximative pointing at a number of different targets. The targets chosen have been geographical features, easily identifiable from a picture, such as mountain tops and features of the sea-shore. The determination of their geodetic coordinates and height has been done based on the maps provided by the Norwegian Polar Institute (Norwegian Polar Institute n.d.).

Test 2. The pointing methods based on the all-sky camera have been tested after calibration.
The testing equipment is the following: the Tracker Camera (instrument 1) placed at approximately 30 cm distance from the alls-sky camera (sensor), both of them being located in the same dome; and the Narrow Field of view sCMOS camera (instrument 2) located at 5 m distance from the sensor, two rooms away. The chosen target is a mountain showing on the horizon, for which due to its proximity the system is expected to perform worse than for the primary target.
It shall be noted that the tests have been conducted with a fixed 110 km height of target. This induces for the Narrow Field of view sCMOS camera a maximum error of 0.2629° in azimuth and 0.007° in elevation, however in the specific scenario it is much smaller: 0° in azimuth and 0.0104° in elevation, according to mathematical calculations similar to the ones presented in *Section 7.1*. The Tracker Camera being at approximately 30 cm distance from the sensor, is not affected by this error: the first noticeable difference (0.05°, the pointing accuracy of the instruments) appears at 75 cm distance from the sensor (in case of a 1 km target altitude, the approximate height of the surrounding mountains), calculated with MATLAB®️ similarly to the calculations in *Section 7.1*.
The pictures recorded during this test are shown in Figure 8.1. The total error in pointing for the Tracker Camera is 0.7725° whilst for the Narrow Field of view sCMOS camera it is 0.0259°.

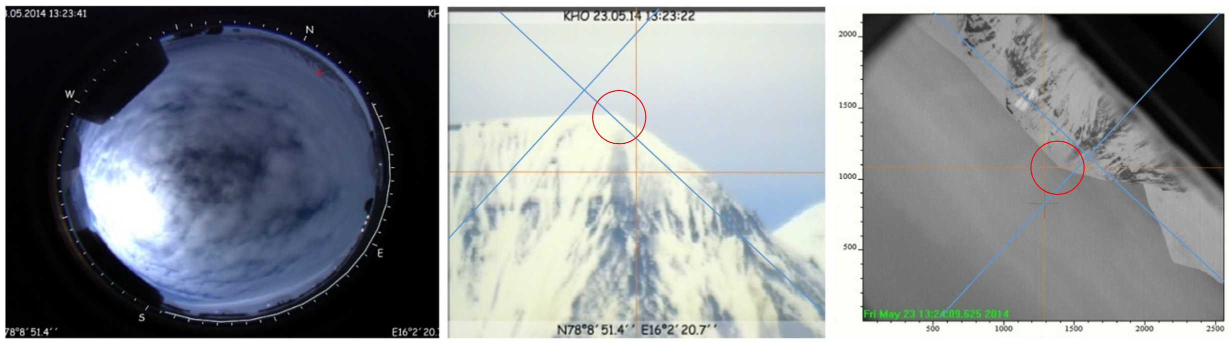


Figure 8.1: Images taken during the validation of the system. The images are recorded as follows (from left to right): image from the all-sky camera, as sensor, the red cross marking the target of pointing; image recorded by the Tracking Camera; image acquired by the Narrow Field of view sCMOS camera. On both images acquired by the instruments the orange cross marks the centre, the blue lines mark the centre of the image from the other camera and the red circle marks the target of pointing.

As one might notice in Figure 8.1, the target on the all-sky camera is at a very low elevation angle, appearing as just a few pixels on the image, with barely identifiable features. Indeed, after selecting the target on the picture usually naked eye test has also been preformed to see better how does the feature look like and hence be able to identify it on the images acquired by the instruments. The erroneous identification of targets shall induce the same magnitude error in all instruments: an offset from the intended target of pointing.
Considering the fact that the two instruments have different error in pointing, an error not justified by wrong target selection and larger in case of the Tracker Camera, further error sources shall be considered. The most probable cause is identified to be calibration error in the initial attitude measurements. As described in *Chapter 6* the calibration method has been different and more inaccurate for the Tracker Camera, that took the second image in Figure 8.1.
As mentioned in the same chapter, at the time of the calibration there has also been a rounding mistake in the geo-pointing feature of the software, used as the basis of the calibration, that has limited the accuracy of the pointing, effecting the initial yaw measurement on both instruments. In conclusion the calibration values have been more inaccurate in this instance then they will be in the future. However, as the Tracker Camera unit is not an instrument of interest but has been used only for testing, and the target is still in the field of view, and since the Narrow Field of view sCMOS camera has a pointing accuracy considered satisfactory, the re-calibration has not been done, and the test has been considered passed.

Summarizing the sources of errors in both units, the following factors are relevant, in decreasing order of their estimated magnitude:

- Narrow Field of view sCMOS camera
  - yaw calibration error,
  - height of target error,
  - target identification error;
- Tracker Camera
  - yaw, pitch and roll calibration error,
  - target identification error.

Chapter 9

Conclusions

The system developed during this thesis enables a large range of scientific observations. Some of the observations that can be done for the first time at KHO are: multiple instruments pointing at the same target, acquiring different data in their nature complementing each other, the same target can be observed by a satellite and ground station with the satellite tracking feature. Also, for the first time at the observatory target acquisition became very intuitive and easily done. However, most importantly this system has accomplished the control of the instruments over the internet, opening up a wide range of possibilities for future development, representing their corner-stone.

The current version of SvalPoint consists of 6 software applications categorized as follows by their role in the system: server, client and control interface. Each instrument is associated with a server application, running on its control computer controlling the instruments, commanded through the client with data about the target from one of the control interfaces. The control interface applications have already been developed at the start of the thesis and only minor changes has been required in them. The client and the server applications have been implemented during the time-span of the thesis work.

The current system is capable of pointing 4 instruments installed at KHO at the same time, counting the test instrument used for development not installed on a fixed platform. The instruments may be picked in any combination, and only the selected ones are commanded. The scientific data collection is not part of the system, standalone programs are developed for them.

The theoretical accuracy of SvalPoint is limited only by the motion accuracy of the individual instruments. However, in practice the following factors influence the pointing accuracy: precision of calibration, human factor at the selection of target in the control interface and the accuracy of the target's height determination.

During the time of the thesis only partial validation has been possible due mainly to light-conditions but also to technical difficulties at KHO. The complete validation and trial of the competence of the system for the required observation is to be done in the up-coming auroral season. After the first trial observations the ease of use for the system is to be assessed and modifications may be done to the user interface or software interactions.

One of the most important features of the system is that its design leaves room for further development. After the real test of the system new programs may be developed, especially on the client side, for target identification. Features such as image processing for the automatic identification of aurora, and the scanning of its entire area, or the mounting of instruments or sensors on truck, aircraft or other moving platforms, with sensors to determine their changing attitude or mobile phone application based remote control are all possible to be developed. Also, further sensors or instruments may be added with little effort, placed at very different locations, extending all the way to the mainland.

Further helpful software applications for making the use of SvalPoint easier would be the ones that help calculating the height of the target and determining the value that shall be the input for the system. As discussed in *Appendix C* the determination of the height of aurora is a complex problem in itself, therefore to be able to do it at a time-scale suitable for the observations an automated software is absolutely necessary. Furthermore, as shown in *Section 7.1* when the height is estimated with a given uncertainty, in order to have the smallest possible errors a different value shall be entered in the system than the estimation. This value can be calculated for each estimation, therefore an option for it in the control interface would further enhance accuracy without causing the operation to be significantly more complicated.

Appendix A

The Kjell Henriksen Observatory

Figures A.1, A.2 and A.3 show details of the KHO building. The pictures are presented to help the better understanding of the need for the system and the motivation for different aims and desired features of SvalPoint.

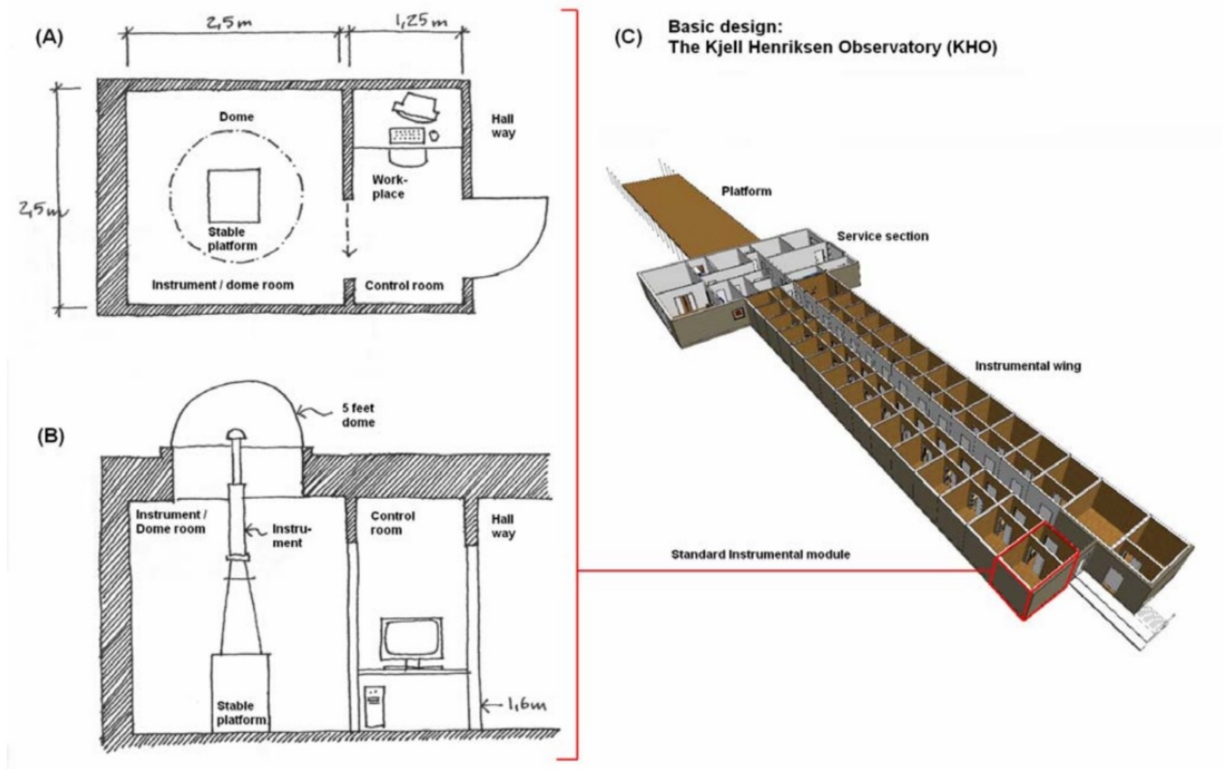


Figure A.1: Basic design of the observatory with a drawing of the standard instrument module. Image from KHO (n.d.*d*).

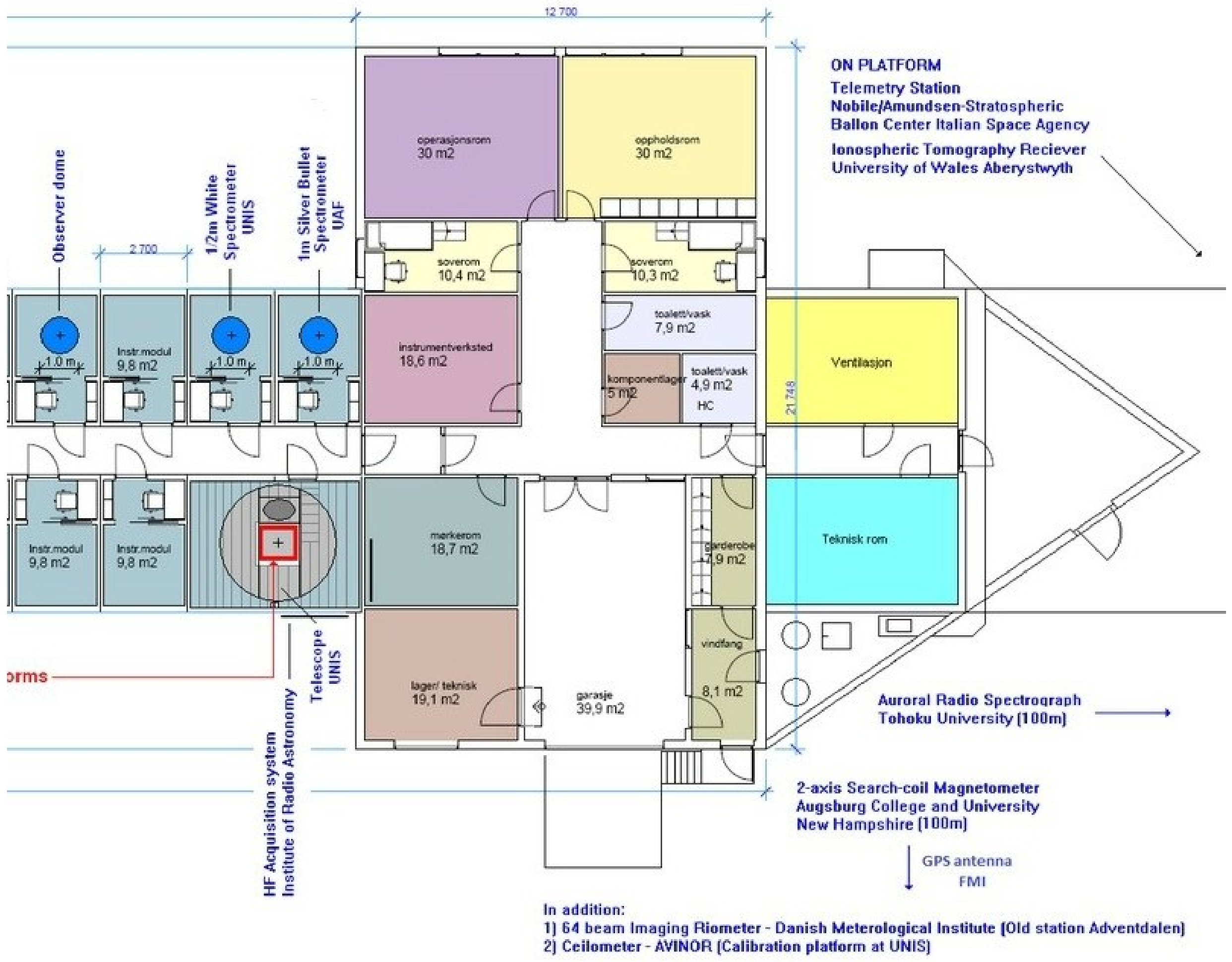

12 700
ON PLATFORM
Telemetry Station
Nobile/Amundsen-Stratospheric
Ballon Center Italian Space Agency
Ionospheric Tomography Reciever
University of Wales Aberystwyth
operasjonsrom 30 m2
oppholdsrom 30 m2
Observer dome
1/2m White Spectrometer UNIS
1m Silver Bullet Spectrometer UAF
2 700
soverom 10,4 m2
soverom 10,3 m2
Instr.modul 9,8 m2
1.0 m
instrumentverksted 18,6 m2
toalett/vask 7,9 m2
komponentlager 5 m2
toalett/vask 4,9 m2
HC
21 748
Ventilasjon
Instr.modul 9,8 m2
Instr.modul 9,8 m2
mørkerom 18,7 m2
garderobe 7,9 m2
Teknisk rom
orms
Telescope UNIS
HF Acquisition system Institute of Radio Astronomy
lager/ teknisk 19,1 m2
garasje 39,9 m2
vindfang 8,1 m2
Auroral Radio Spectrograph
Tohoku University (100m)
2-axis Search-coil Magnetometer
Augsburg College and University
New Hampshire (100m)
GPS antenna
FMI
In addition:
1) 64 beam Imaging Riometer - Danish Meterological Institute (Old station Adventdalen)
2) Ceilometer - AVINOR (Calibration platform at UNIS)

Figure A.2: Fragment of the map of KHO showing allocation of instruments with the focus on the facilities located in the service section of the building. Note the operational room marked with 'operasjonsrom'. Image from KHO (n.d.*d*).

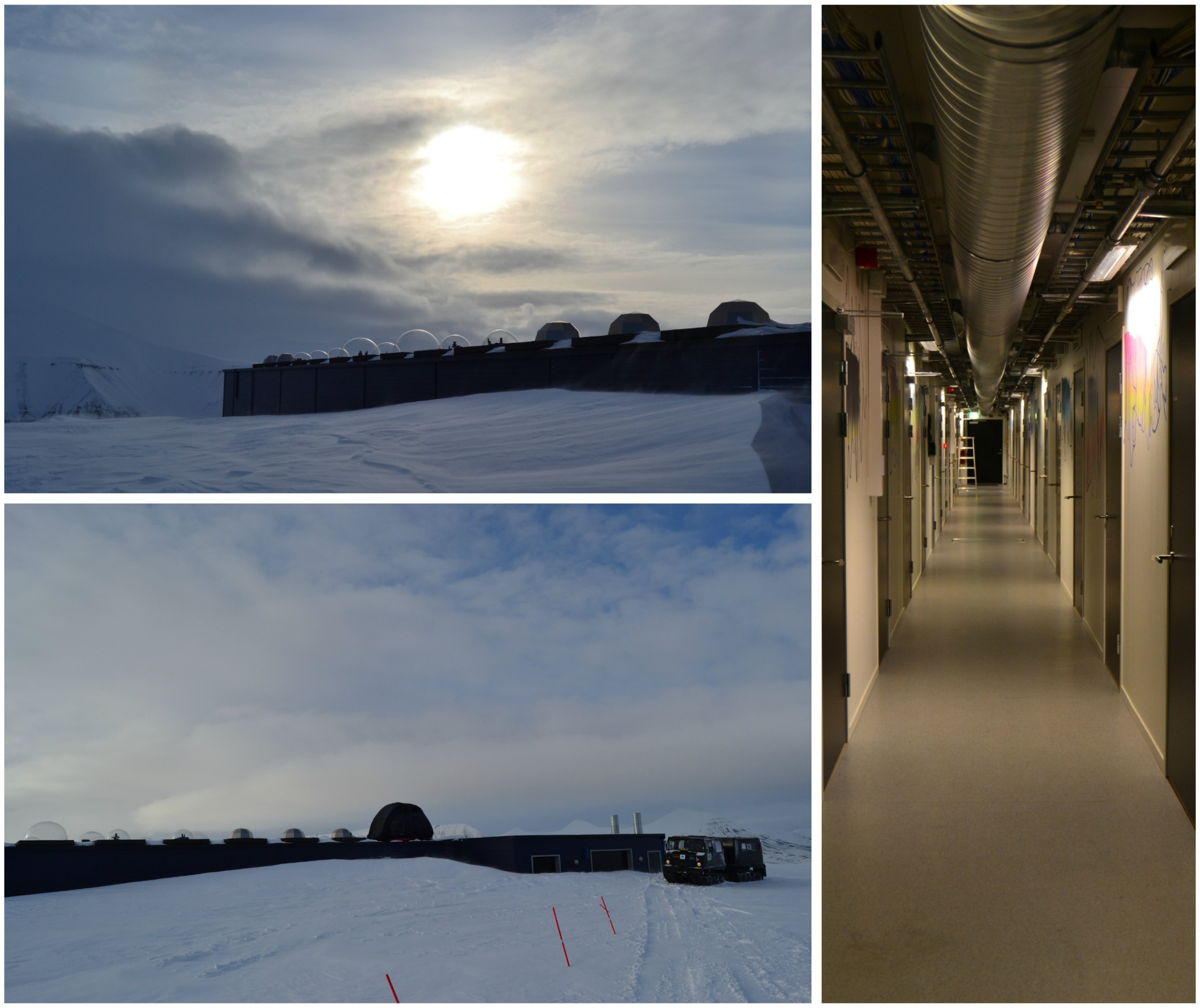


Figure A.3: Photos taken at KHO showing the domes on the roof, housing the instruments, the building with the road leading to it, and the corridor of the instrumental wing with the instrument modules on both sides.

Appendix B

Tait-Bryan angles

The Tait-Bryan angles, also named as asymmetrical Euler angle sets or 3-2-1 Euler angles, represent rotations around three distinct axes: x, y and z. The angles are named pitch, roll and yaw, and in the case of a ship or aircraft they are defined as follows (Berlin 2007):

- pitch is the up-down movement of the longitudinal axis, in general rotation around axis y,
- roll is the rocking movement around the longitudinal axis, in general rotation around axis x,
- and yaw is the sideways movement of the longitudinal axis, in general rotation around axis z.

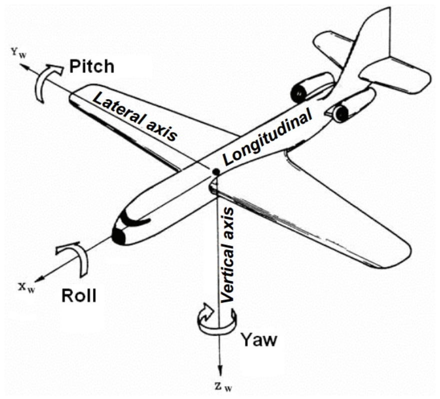


Figure B.1: Illustration of the principal axes of an aircraft. Figure adapted from Mahoney (n.d.)

In the case of an aircraft Figure B.1 illustrates the three angles. The same angles are adapted for the different reference frames associated with optical instruments: the rotations around the axes follow the convention described. For a left handed reference frame placed in the centre of the lens see Figure B.2.

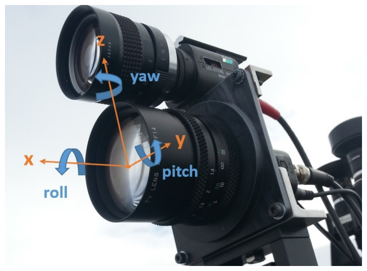


Figure B.2: Illustration of the principal axes of a camera.

Appendix C

Calculation of the height of aurora

The determination of the height of aurora is an issue that has been studied since the 1950-s, yet still a problem today, resulting in numerous scientific articles. The earliest studies use triangulation in general based on different cameras (narrow field of view, all-sky), with a spatial displacement from 4 to 100 km. An early study of auroral heights is done in Strømer (1955), using narrow field of view camera photography placed at distances of only a couple of kilometres, calculating the altitude of a single point in the auroral arc. Studies on the determination of the height and position of the aurora based on a single sensor has been conducted in Rees (1963) followed by Dandekar (1974) further improving the method. Kaila (1987) shows a method for determining the height along the entire auroral form with a ±1 km precision if certain criterion are met, using two all-sky cameras placed at 100-200 km apart from each other. More recent articles on the subject are Sigernes et al. (1996) determining the height of midmorning aurora with photometers and Whiter et al. (2013), presenting a method for finding the peak emission height of the aurora based on a pair of all-sky camera images, bringing improvement in speed, accuracy and level of automation compared to the earlier works.

The presented methods for locating the place of aurora involve visual or photometric data. However, due to the nature of the phenomenon other measurements can be used as well by understanding the basic process of the aurora.
The cause of the aurora is protons and electrons accelerated at thousands of kilometres height. The large flux of electrons moving downwards into the ionosphere (called electron precipitation) collide with atoms and molecules, getting them in an excited state. The excess energy during the de-excitation creates optical emission (auroral arcs) and increases conductivity by creating secondary charged particles. The optical emission and conductivity profiles are strongly dependent on the flux and energy spectrum of the precipitating electrons. Therefore models can be used to determine the height of the aurora (among other parameters) in case information about the precipitating electrons is available.
To determine these parameters different approaches may be used. One approach is to collect in-situ data from satellites or sounding rocket measurements. An example for the latter one presented in Sharp et al. (1979), showing height distribution of emission rates, ion densities, auroral electron densities and flux.
One of the EISCAT radars, located in close vicinity of KHO can also be used to determine the height. EISCAT determines the height distribution of the electron density and electron temperature among its four basic parameters. These parameters can be also combined with optical observations making complex observations possible. In Frey et al. (1998) a three dimensional model of the auroral emission is constructed with the use of optical tomography and EISCAT data.

In conclusion, after presenting different methods and options for determining the heigh of the aurora it shall be noted that all these observations are complex, therefore take time, possibly not comparable with the dynamics of the phenomena. In case any of the options is to be used, a well automated process is at need.
On the other hand, depending on the precision needed for the observations a fairly good guess about the height might be sufficient. See the induced angle errors by imprecision in height in *Section 7.1.*

Appendix D

Universal Transverse Mercator coordinate system

The Universal Transverse Mercator (UTM) coordinate system uses two-dimensional Cartesian coordinates to define locations on the surface of Earth. It is the preferred system for large scale mapping (1:250000 and larger scales), giving the advantage of simple numbers over the complex degrees, minutes and seconds of latitude and longitude. For more details consult Terry Jr (1996).

As described in Hager et al. (1992) as well the Mercator Projection can be visualised as an ellipsoid projected into a cylinder, tangent at the equator and the polar axis coinciding with the axis of the cylinder. Opening and flattening the cylinder results in distortion at the polar regions, while the showing true distances along the Equator.
The Transverse Mercator Projection is a Mercator Projection with the cylinder rotated by 90°, resulting in the ellipsoid and cylinder being tangent along a meridian. In this case when the cylinder is flattened the east and west extremities appear distorted, however there is no distortion along the meridian that is tangent to the cylinder, showing true distances. The zone around the meridian considered still sufficiently little distorted is 3° in each direction, resulting in 6° wide portions for the system, dividing the Earth in 60 longitudinal mapping zones. See also Hager et al. (1992).

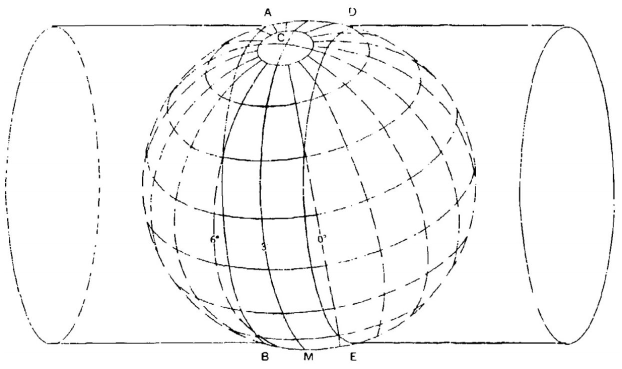


Figure D.1: Illustration of the UTM. CM - Central Meridian; AB, DE - Lines of secancy formed by the intersection of cylinder and ellipsoid. Image from Hager et al. (1992).

The Universal Mercator Projection modifies the cylinder of projection by reducing its elliptical dimensions and making it secant to the ellipsoid. It results in intersecting the ellipsoid along lines parallel to the central meridian. (See Figure D.1) In one 6° zone the lines of secancy establish approximately 180000 m east and west from the central meridian. The effect of these lines of secancy is that of allowing a more consistent relationship between ellipsoid and map distances, resulting in a scale factor of 0.9996 for the conformal coordinates in the plane. (See Hofmann-Wellenhof et al. (2001), Hager et al. (1992).)

The grid zones are defined uniform over the globe, however as shown in Figure D.2 there are exceptions to this rule in the Scandinavian region: the south-west coast of Norway and the area of Svalbard. Around Svalbard there are four grid zones 31X, 33X, 35X and 37X are extended to widths of 9°, 12°, 12° and respectively 9°, reducing the number of zones from seven to four, resulting in the fact that the grid zones 32X, 34X and 36X are not in use.

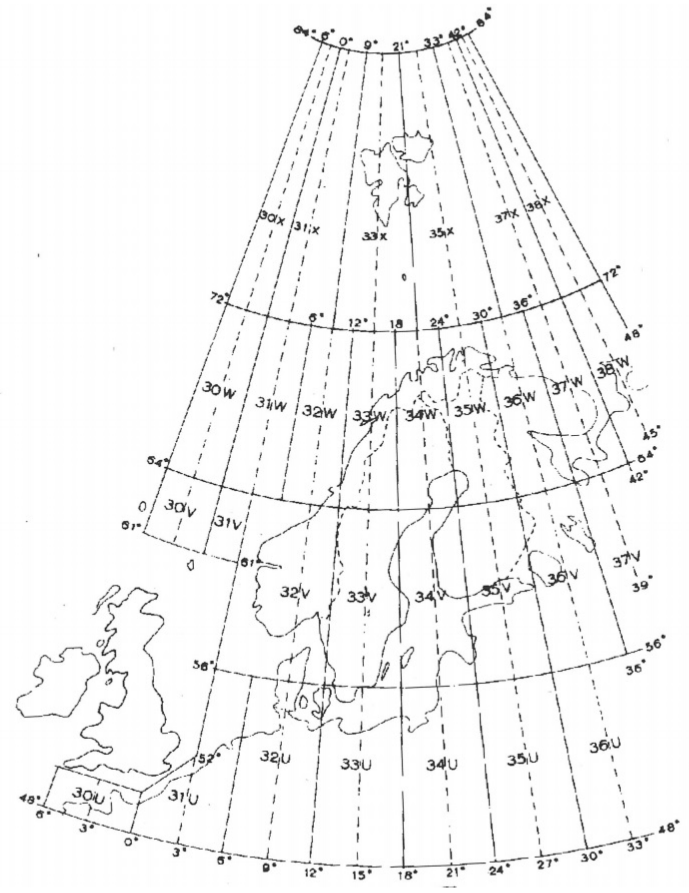


Figure D.2: UTM grid zones in Scandinavia. Figure from Bernhardsen (1992).

Appendix E

Fragments from the user manual to the SvalPoint system

Due to the complexity of the system a user manual is provided to it that contains practical information for the operator of the system. It also describes the structure of the software and the algorithms used for the developer maintaining and extending it. This appendix shows fragments of the user manual, showing examples of the practical information included in it.

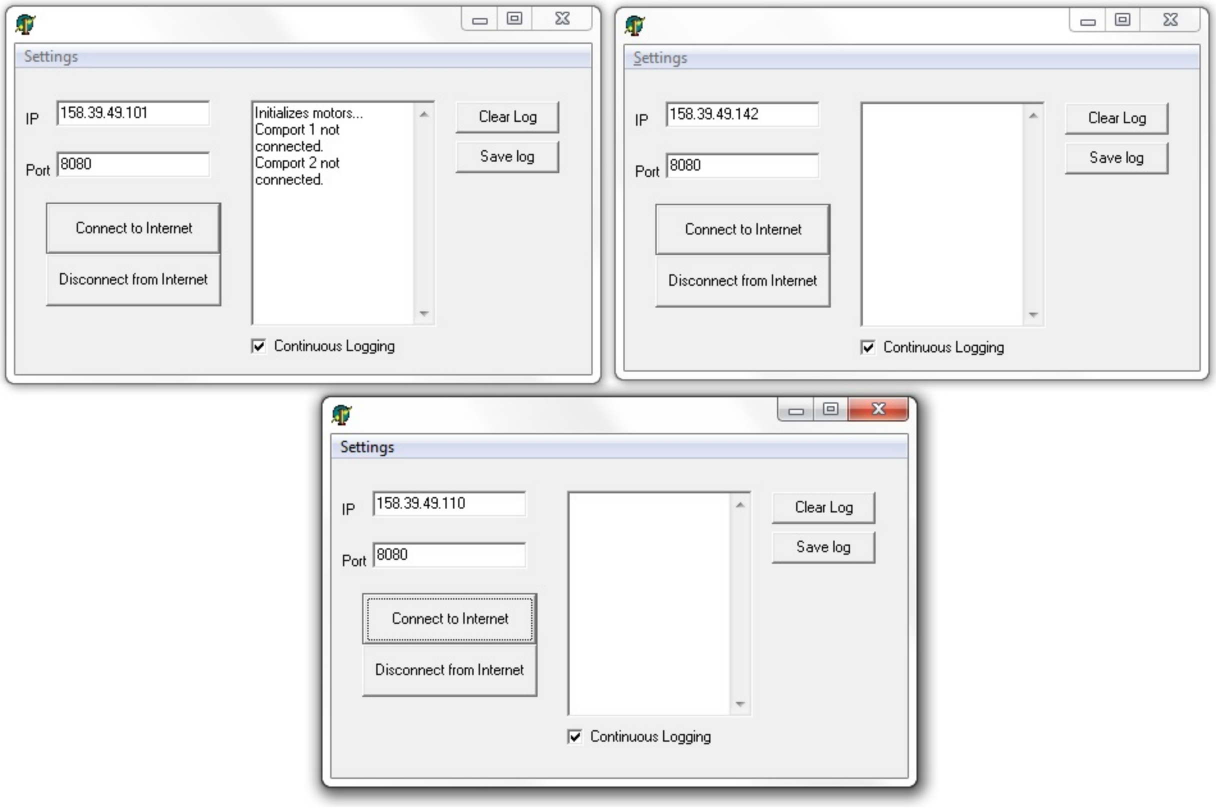
E.1 What is SvalPoint?

The SvalPoint system is a collection of software programs that control a number of optical instruments at the Kjell Henriksen Observatory (KHO). The instruments are controlled with internet protocol, therefore they can be controlled from the building of the University Centre in Svalbard (UNIS). It shall be noted however that the Firewall settings in Windows on the computers at KHO do not allow access from other than PC-s registered as part of the KHO network.

E.2 How to use SvalPoint?

On the computers connected directly to the instruments, at KHO, the server programs shall run: **Fs Ikea Tracker** (for the Fs-Ikea instrument), **KeoSkyScanner** (for the Narrow Field of view sCMOS camera) and **PTU46 Tracker** (for any instrument fixed on PTU D46 platform). The server applications are ought to be installed already on the computers. In case they are not, copy the folder of each server directly to the C: (in case of the PTU46 Tracker and KeoSkyScanner) and to the D: directory (in the case of the Fs Ikea Tracker).
When the programs are launched they look as in Figure E.1. Verify the IP address of the computer, also note the Port. The same Port number shall be used in the client applications. After all parameters are set correctly, connect it to the internet by clicking the 'Connect to Internet' button. If continuous logging is not desired, remove the tick from the check-box. In that case you can save the content of the dialogue window by clicking 'Save log'. For emptying the log file click 'Clear Log'. The log file can be accessed in the directory of the program. The files are named FS_IKEAScanner_log.txt, KeoSkyScanner_log.txt, and PTU46Tracker_log.txt for the three different instruments.
Also make sure that the COM-ports are set correctly for the instruments, being one of the most common troubles. Access it through the 'Settings' tab, then 'Port Settings'. Verify the rest of the settings through the other tabs. Use the 'Help' tabs for more information.

The second part of the system is the client. That can be launched on any computer registered as part of the KHO network. The software is named **SvalCast**. Launch it and select the instruments from the list that you wish to control, then click 'Connect'. Feedback from the servers appear in the 'Info' field. If the



IP 158.39.49.101
Port 8080

```
Initializes motors...
Comport 1 not
connected.
Comport 2 not
connected.
```

☑ Continuous Logging

IP 158.39.49.142
Port 8080

☑ Continuous Logging

IP 158.39.49.110
Port 8080

☑ Continuous Logging

Figure E.1: The user interface of the server applications.

IP address or Port number is incorrect, edit them in the Tracker_Console_Clients.txt file. Use this file if you wish to add new instrument as well. The format is the following, each instance on new row: Displayed name, IP address, Port number, selection status (0 for not selected and 1 for selected). There shall be no empty row between the different instruments.
This client connects to a command interface. The available command interfaces currently are: **SvalTrack II** and **Sval-X Camera**. Launch one of these programs. *Note: it is possible to launch both of them at the same time, however it is counter-advised.* In the SvalTrack II select any satellite or celestial body for tracking. In the Sval-X Camera click on the image acquired by the all-sky camera for instrument control. On the settings of these programs please consult the dedicated documentation or contact the person in charge.
Alternatively Telnet can be launched and the commands can be sent in raw format. See section *Format of commands* for more details.

E.3 Format of commands

There are five different commands that can be sent to the servers. Two of them are routine commands: the homing command (home command word) and the end of connection command (end keyword). The fist target acquisition command is for geo-pointing, with the keyword gps. The second and third one are for direction based pointing when the height of the target is known (command word: azelh), and when the target is considered to be infinitely far away compared to the distance between sensor and instrument (command word azel).
The format of the commands is as follows (each line is sent as a separate message):

- Send instrument into home position
  - keyword 'HOME', 'Home' or 'home'
- Geo-pointing
  - keyword 'GPS', 'Gps' or 'gps'
  - the value of latitude in degrees (defining the target)
  - the value of longitude in degrees (defining the target)
  - the value of altitude in metres (defining the target)
- Direction based pointing, azel
  - keyword 'AZEL', 'AzEl', 'Azel' or 'azel'
  - the value of latitude in degrees (defining the place of sensor)
  - the value of longitude in degrees (defining the place of sensor)
  - the value of altitude in metres (defining the place of sensor)
  - the value of azimuth in degrees (measured on the sensor in North-East-Up reference frame)
  - the value of elevation in degrees (measured on the sensor in North-East-Up reference frame)
- Direction based pointing, azel
  - keyword 'AZELH', 'AzElH', 'Azelh' or 'azelh'
  - the value of latitude in degrees (defining the place of sensor)
  - the value of longitude in degrees (defining the place of sensor)
  - the value of altitude in metres (defining the place of sensor)
  - the value of azimuth in degrees (measured on the sensor in North-East-Up reference frame)
  - the value of elevation in degrees (measured on the sensor in North-East-Up reference frame)
  - the height of target in kilometres
- End connection
  - keyword 'END', 'End' or 'end'

E.4 Known issues and quick trouble shooting

E.4.1 SvalCast is not responding after the first command was sent

The most probable cause of this issue is rooted in COM port settings in the server. The program does stop responding in some cases, related to lack of COM port connection. Please verify the COM port settings from the server application, but also check that all RS232 cables are connected to the computer, that the motor control unit is on and all motors are powered. Please restart the server programs, then kill the SvalCast from the Task Manager and try it again.

E.4.2 SvalCast is not connecting to the servers

First make sure that the computer that you launched SvalCast on is a KHO computer. Check that the SvalCast is launched only once on the computer, as if another instance is running, that occupies the port. If this was not the cause of the problem check that the Server programs are all running and connected to the internet. This might be the most probable cause if only some of the servers fail to respond. Also, verify the IP address and the Port number both in the Server applications and the SvalCast. Verify firewall settings on the server computers to make sure that connections are allowed. If all else failed try restarting the server programs.

E.4.3 Target acquisition failed

In case the target acquired is incorrect, the most probable issue is calibration. Check the *Calibration* section of the User Manual for more information on how to calibrate each instrument and update the parameters.

E.4.4 Multiple instruments pointing at different targets

The first thing needed to be done is to check that the height of target information is set in this case. In the case of the Sval-X Camera software 0 km height means infinite distance, therefore the instruments point parallel. In case you have information about the height of the target, enter it in km in the dedicated dialogue box. If this does not solve the problem verify the calibration data of the instrument that points in the wrong direction.
Note: The accuracy of pointing is determined by many factors. It is worse at lower altitudes and at lower accuracy of determination of height. If everything is shown to be correct and the instruments still do not point at the same target, the system might have reached its accuracy limit. See more about it in the section Accuracy of pointing.

E.4.5 Target acquisition stops unexpectedly

In case during the operation of the system it stops, first restart both SvalCast and the control interface used. Then, if the problem still persists verify that the servers are still connected to the internet, alternatively restart them.

Appendix F

Code fragments from server programs

This appendix shows fragments from the code of the server applications.

Listing F.1 shows the type definitions used throughout the server programs for better readability. Please refer to this listing for the understanding of variables in the other code fragments.

Listing F.1: Type definitions used in the server programs

```
type
    {all angles in radians!!!}
    Cartesian = Record
                  X, Y, Z : Real;
                End;

    Geodetic = Record
                  lat, lon, alt : Real;
                End;

    LocalLevel = Record
                  az, el : Real;
                End;

    RotMatrix = array[0..2, 0..2] of Real;
    Vector = array[0..2] of real;
```

Listing F.2 is an example for the implementation of basic functions, not related directly to the application, however necessary due to their lack in the language Pascal by Borland.

Listing F.2: Example for a basic function

```
{
------------------------------------------------------------------
--------------------Arccos function-------------------------------
------------------------------------------------------------------
}
function TPTU46TrackerF.arccos(a:real):real;
begin
  arccos := ArcTan (sqrt (1-sqr (a)) /a);
end;
```

Listing F.3 shows an example for error detection in the information coming from the client. The procedure that executes when a string arrived in the TCP socket calls this function to extract the information from the string received. First it is checked that the received string indeed contains a number, followed by checking that the number is within the requested boundaries, [-90, 90) in this case. If it is not in the boundary, a global boolean variable (errorinCmd : **boolean**;) is set that indicates that there has been an error in the command.

Listing F.3: Example for error detection. The algorithm that reads the latitude value from the client

```
{————————————————————————————————————————————————————————
————————————Get latitude input function—————————————————
————————————————————————————————————————————————————————}
function TPTU_D46_Serv.getlat(sInCommand:String): Real;
var
   {Variables for interpreting commands}
   Code: Integer; {string to int conversion help variable}
   Value: Real; {string to int conversion help variable}
   lat: real;

begin
lat:=0;
{Saving command in log file if continuous logging is on}
if(ContLog.State = cbChecked) then
   begin
      Append (Log);
      Write (Log,  DateTimeToStr(Now) + ': ' + sInCommand + String(#13#10));
      CloseFile(Log);
   end;

   {Check if the command is a number.
   If it is, save it in lat.}
   Val(sInCommand, Value, Code);

   if (Code = 0) then
     begin
        lat := Value;
        {if angle is not out of range}
        if ((lat < -90) and (lat >= 90)) then
           begin
               errorinCmd := True;
           end;
        end
           else
               errorinCmd := True;    {Command not a valid number}
   getlat := lat;
end;
```

Example for an application specific function may be seen in **Listing F.4**. This function transforms geodetic coordinates into Cartesian coordinates.

Listing F.4: Function for the change of geodetic coordinates into cartesian coordinates.

```
{
------------------------------------------------------------------------
---------Change geodetic to cartesian coordinates----------
--P.279 from B. Hofmann-Wellenhof, H. Lichtenegger, and-
--J. Collins: GPS, Theory and Practice. Fifth edition---
-----------------SpringerWienNewYork, 2001-----------------
------------------------------------------------------------------------
}
function TPTU46TrackerF.ToCartesian(input:Geodetic):Cartesian;
var
  a, b: real; {Semi-major and semi-minor axis of ellipsoid}
  N: real; {Radius of curvature}
begin
  {WGS 84 parameters}
  a := 6378137.0;
  b := 6356752.3;
  {Calculate radius of curvature}
  N := a*a / (sqrt(a*a * cos(input.lat)*cos(input.lat) +
                           b*b * sin(input.lat)*sin(input.lat)));
  {Calculate Cartesian coordinates}
  ToCartesian.X := (N+input.alt) * cos(input.lat) * cos(input.lon);
  ToCartesian.Y := (N+input.alt) * cos(input.lat) * sin(input.lon);
  ToCartesian.Z := (((b*b)/(a*a))*N + input.alt) * sin(input.lat);
end;
```

The function labelled with Command from Client, shown in **Listing F.5** is the procedure called when a new string appears in the TCP socket. This is the part of the code that differs from server to server. In the followings a short walk-through of the procedure and explanations about one of the commands are presented.

The procedure first verifies whether the string is one of the command words. If it is not, no action is taken. If the string matches a command word, depending on each case other strings might be read from the client. The cases are marked by visible comments in the code for readability. In this fragment the *Home* and *End* commands are shown, together with a direction based pointing command: *AzEl*. The other pointing commands acquiring target are similar, therefore not explained. (See *Section 4.2.3* for more details on the command protocol.)
Upon the *Home* command the 'TP' and 'PP' strings are sent to the unit, the keywords that move the tilt and pan motors in their home position. On the *End* command the TCP connection between the client and server is ended.
The actions on *AzEl* command are the calculation of the pointing direction, the control of the instrument and the sending of feedback to client. All steps for the transformation of input information into degrees of azimuth and elevation for the instrument in the PSRF are clearly marked. (See *Chapter 5* for further details.) The next step is to adjust these angles to the values that need to be sent on the COM port, taking into account the command format of each instrument.
In the case of the PTU D46 the sign of the number defines the direction of motion, referenced from the home position of the motors. A negative number moves the unit to the left and up, while the positive number moves it to the right and down. Moreover, the motors are commanded not by angle values but step numbers. The transformation is done by the help of the PR: **real**; and TR: **real**; variables, set from the setting tab to the motor resolution. Also, the boundaries of the unit are checked before sending the command to the unit, by comparing the values to the minimum and maximum positions of both motors, saved in the PN: **Integer**; PX: **Integer**; TN: **Integer**; and TX: **Integer**; variables.
When the command is being sent to the instrument the **procedure** SendCmd(S:String); is called. This sends the string on the COM port. The string contains the word marking which motor shall move ('PP' for the pan motor, controlling elevation and 'TP' for the tilt motor, controlling azimuth) followed by the number of steps that shall be executed. The *while* loop after the sending of the command shall be noted. The donea, donee: **boolean**; global variables mark whether the command has been executed or not. The flags are set on **true** when the feedback from the unit, marking its position, contains the expected values. While the command has not been executed the current position of the unit is requested.
Last, upon the execution of the command feedback is sent to the client.

Listing F.5: Fragments of the function called when a string is received in the TCP Socket.

```
{
------------------------------------------------------------------
--------------------COMMAND FROM CLIENT---------------------------
------------------------------------------------------------------
}
procedure TPTU46TrackerF.IdTCPServer1Execute(AContext: TIdContext);
var
    {String to read command from client}
    sInCommand: String;
    {Integers to calculate azimuth and elevation angles}
    nAzimuthPSRF: Integer;
    nElevationPSRF: Integer;
    {variables read from command}
    lat: real;
    lon: real;
    alt: real;
    az: real;
    el: real;
    h: real;
    {Angles based control variables}
    v1: real;
    {geodetic coordinates based control variables}
    HomeC: Geodetic;
    Target: Geodetic;
    PointingDir: Vector;
    {Variables for the SWRF - IWRF - PSRF transformation}
    neuIWRF: Vector;
    neuSWRF: Vector;
    neuPSRF: Vector;
    neuSI: Vector;
    Sensor: Geodetic;

begin
    AContext.Connection.Socket.WriteLn(DateTimeToStr(Now) +
                                       '␣PTU␣D46:␣Connection␣set␣up.');
    repeat    {While end command is read.}

      {Read New Command}
      sInCommand := AContext.Connection.Socket.ReadLn;

      {Saving command in log file if continuous logging is on}
      if(ContLog.State = cbChecked) then
        begin
          Append (Log);
          Write (Log,  DateTimeToStr(Now) + ':␣' + sInCommand + String(#13#10));
          CloseFile(Log);
        end;

      Memo1.Lines.Add(sInCommand);

      {Open COM port}
      if not ComPort1.Connected then
       try
        ComPort1.Open;
       except
        on E:Exception do begin
          memo1.lines.add('Comport␣not␣connected.');
        end;
       end;


      {-------------------------
            Home command
      -------------------------}
      if ((sInCommand = 'Home') or (sInCommand = 'home') or (sInCommand = 'HOME')) then
        begin
          home := false; {New home command recieved}
          GoHome;
          if Comport1.Connected then begin
            while(home = false) do
            begin
              timer1.enabled:=true;
              sleep(10);
              sendcmd('TP');
              sendcmd('PP');
            end;
          timer1.enabled:=false;
      end
        else errorinCmd := True; {COM port not connected}

       if(home = true) then
           AContext.Connection.Socket.WriteLn(DateTimeToStr(Now) +
                                                 ' PTU D46: Command executed.')
       else if (errorinCmd = True) then
           AContext.Connection.Socket.WriteLn(DateTimeToStr(Now) +
                                                 ' PTU D46: Error in command.');
  end;

{------------------------------------
     Azimuth and Elevation command
-------------------------------------
    Receives lat, lon, alt, az, el
-------------------------------------}

if ((sInCommand = 'AzEl') or (sInCommand = 'AZEL')
            or (sInCommand = 'azel') or (sInCommand = 'Azel')) then
  begin

    errorinCmd := False;{New commands}

    {Read commands in order}
    sInCommand := AContext.Connection.Socket.ReadLn;
    lat:=getlat(sInCommand);
    sInCommand := AContext.Connection.Socket.ReadLn;
    lon:=getlon(sInCommand);
    sInCommand := AContext.Connection.Socket.ReadLn;
    alt:=getalt(sInCommand);
    sInCommand := AContext.Connection.Socket.ReadLn;
    az:=getaz(sInCommand);
    sInCommand := AContext.Connection.Socket.ReadLn;
    el:=getel(sInCommand);


    {If all inputs are correct}
    if (errorinCmd = False) then
        begin

          {Set the new azimuth and elevation value in the AzElWRF}
          AzElWRF.az := DegToRad(az);
          AzElWRF.el := DegToRad(el);

          {Set sensor coordinates}
          Sensor.lat := DegtoRad(lat);
```

```
 Sensor.lon := DegtoRad(lon);
 Sensor.alt := alt;

 {Set instrument coordinates}
 HomeC.lat := DegtoRad(lath);
 HomeC.lon := DegToRad(lonh);
 HomeC.alt := alth;

 {Calculate vector based on n, e and u components.}
 neuSWRF := AzElToNEU(AzElWRF);


 {Target infinitely far away: IWRF = SWRF}
 neuIWRF[0] := neuSWRF[0];
 neuIWRF[1] := neuSWRF[1];
 neuIWRF[2] := neuSWRF[2];

 {Transform from IWRF to PSRF - translate and rotate vector}
 neuIWRF[0] := neuIWRF[0] - xoff;
 neuIWRF[1] := neuIWRF[1] - yoff;
 neuIWRF[2] := neuIWRF[2] - zoff;

 neuPSRF := RotateToPSRF(neuIWRF);

 {Calculate azimuth and elevation from neu
 - not real elevation, zenith.}
 AzElPSRF := NEUToAzEl(neuPSRF);

{Calculate azimuth command}
AzElPSRF.az := RadToDeg(AzElPSRF.az);
{Calculate degrees in direction}
if (AzElPSRF.az > 180) then begin
   AzElPSRF.az := 360 - AzElPSRF.az ;
   end
else
   AzElPSRF.az := - AzElPSRF.az;

Str(AzElPSRF.az,sAzimuthPSRF);
memo1.Lines.add('azimuth:␣' + sAzimuthPSRF);
{Calculate number of steps}
nAzimuthPSRF := round(AzElPSRF.az/PR);

{Calculate elevation command}
AzElPSRF.el:=RadToDeg(AzElPSRF.el);
{Calculate degrees in direction}
if (AzElPSRF.el >= 0) then
begin
  AzElPSRF.el := -90+AzElPSRF.el ;
  end
else
begin
  AzElPSRF.el := 90+AzElPSRF.el ;
  end;
```

```
        Str(AzElPSRF.el,sElevationPSRF);
        memo1.Lines.add('Elevation:␣' + sElevationPSRF);
        {Calculate number of steps}
        nElevationPSRF := round(AzElPSRF.el/TR);

        {Set az and el on instrument}
        if Comport1.Connected then
         if ((nAzimuthPSRF <= PX) and (nAzimuthPSRF >= PN) and
                   (nElevationPSRF <= TX) and (nElevationPSRF >= TN)) then
           begin
            Str(nAzimuthPSRF,sAzimuthPSRF);
            Str(nElevationPSRF,sElevationPSRF);
            {Send elevation command}
            donea := false; {New command}
            SendCmd('PP'+sAzimuthPSRF);
            while (donea = false) do
                begin
                  timer1.enabled:=true;
                  sleep(100);
                  SendCmd('PP');
                end;
            timer1.enabled:=false;
            {Send azimuth command}
            donee := false; {New command}
            SendCmd('TP'+sElevationPSRF);
            while (donee = false) do
               begin
                 timer1.enabled:=true;
                 sleep(100);
                 SendCmd('TP');
             end;
            timer1.enabled:=false;
         end
        else
         errorinCmd := True {Angles out of range}
      else
         errorinCmd := True; {COM port not connected}
  end;

  {Send feedback when command executed.}
  if ((donee = True) and (errorinCmd = False)) then
       AContext.Connection.Socket.WriteLn(DateTimeToStr(Now)
                + '␣PTU_D46:␣Command␣executed.')
  else
     if (errorinCmd = True) then
       AContext.Connection.Socket.WriteLn(DateTimeToStr(Now)
                + '␣PTU_D46:␣Error␣in␣command.');

Memo1.Lines.Add('Azimuth␣=␣' + sAzimuthPSRF + ',␣Elevation␣=␣' +
                                                sElevationPSRF);

end;
{End of azimuth and elevation command}
```

```
(...)

    {——————————————————————————
            End command
    ——————————————————————————}
    until ((sInCommand = 'End') or (sInCommand = 'END') or (sInCommand = 'end'));
    AContext.Connection.Socket.WriteLn('PTU_D46:␣Connection␣ended.');
    AContext.Connection.Disconnect;
end;
```

Listing F.6 shows the *AzEl* command code for the Fs-Ikea instrument. It is shown in order to illustrate the nature of the differences between server programs. In contrast with the same command implementation for the PTU D46 one may notice that the way of calculating the direction vector is identical, only in the determination of the actual commands changes.

The Fs-Ikea instrument has a different protocol for the commands than the PTU D46. The azimuth motor moves only 90° to each side, while the elevation motor moves 180° to ensure whole sky visibility.
Another difference is that the control of this instrument is not absolute, but always referenced to its current position, making it necessary to keep track of it in the CAs, CEs: **Integer**; variables.
Furthermore, there is no option on this instrument to make an inquiry of the current position. Therefore a waiting function is used: **Procedure** WaitOnMotors(Steps:**integer**); implements a delay based on the number of steps that the unit will take. It takes into consideration the acceleration, nominal speed and deceleration of the motor, calculating the time needed for the motion to be executed.

Listing F.6: Function fragment illustrating the difference between the different server programs.

```
      {————————————————————————————————
          Azimuth and Elevation command
      ————————————————————————————————
          Receives lat, lon, alt, az, el
      ————————————————————————————————}

      if ((sInCommand = 'AzEl') or (sInCommand = 'AZEL') or (sInCommand = 'azel') or
                                                      (sInCommand = 'Azel')) then
        begin

          errorinCmd := False;{New commands}

          {Read commands in order}
          sInCommand := AContext.Connection.Socket.ReadLn;
          lat:=getlat(sInCommand);
          sInCommand := AContext.Connection.Socket.ReadLn;
          lon:=getlon(sInCommand);
          sInCommand := AContext.Connection.Socket.ReadLn;
          alt:=getalt(sInCommand);
          sInCommand := AContext.Connection.Socket.ReadLn;
          az:=getaz(sInCommand);
          sInCommand := AContext.Connection.Socket.ReadLn;
          el:=getel(sInCommand);


          {If all inputs are correct}
          if (errorinCmd = False) then
              begin
```

```
 {Set the new azimuth and elevation value in the AzElWRF}
 AzElWRF.az := DegToRad(az);
 AzElWRF.el := DegToRad(el);

 {Set sensor coordinates}
 Sensor.lat := DegtoRad(lat);
 Sensor.lon := DegtoRad(lon);
 Sensor.alt := alt;

 {Set instrument coordinates}
 HomeC.lat := DegtoRad(lath);
 HomeC.lon := DegToRad(lonh);
 HomeC.alt := alth;

 {Calculate vector based on n, e and u components.}
 neuSWRF := AzElToNEU(AzElWRF);

 {Target infintely far away: IWRF = SWRF}
 neuIWRF[0] := neuSWRF[0];
 neuIWRF[1] := neuSWRF[1];
 neuIWRF[2] := neuSWRF[2];

 {Transform from IWRF to PSRF - translate and rotate vector}
 neuIWRF[0] := neuIWRF[0] - xoff;
 neuIWRF[1] := neuIWRF[1] - yoff;
 neuIWRF[2] := neuIWRF[2] - zoff;

 neuPSRF := RotateToPSRF(neuIWRF);

 {Calculate azimuth and elevation from neu
 - the result is not real elevation, viewangle.}
 AzElPSRF := NEUToAzEl(neuPSRF);

 {Change angles into degrees}
 {Note: the elevation is actually the viewangle.
     For the FS-ikea the command is based on the viewangle.}
 AzElPSRF.az := RadToDeg(AzElPSRF.az);
 AzElPSRF.el:=RadToDeg(AzElPSRF.el);

{-----Calculate azimuth and elevation commands-----}

{Calculate elevation (viewangle) degrees in direction}
nElevationPSRF := round(AzElPSRF.el);
if (nElevationPSRF >= 0) then
  AzElPSRF.el := -AzElPSRF.el
else
  errorinCmd := True; {Instrument is incapable of pointing downward}

{Calculate azimuth degrees for instrument with direction}
nAzimuthPSRF := round(AzElPSRF.az);
if (nAzimuthPSRF > 90) then
   begin
```

```
            AzElPSRF.az := -180 + AzElPSRF.az;
            AzElPSRF.el := -AzElPSRF.el;
          end
       else if (nAzimuthPSRF < -90) then
          begin
            AzElPSRF.az := 180 + AzElPSRF.az;
            AzElPSRF.el := -AzElPSRF.el;
          end
       else
          AzElPSRF.az := AzElPSRF.az;

       {Print angles for instrument}
       Str(AzElPSRF.az,sAzimuthPSRF);
       Str(AzElPSRF.el,sElevationPSRF);
       memo1.lines.add('Azimuth␣angle:␣'+sAzimuthPSRF);
       memo1.lines.add('Elevation␣angle:␣'+sElevationPSRF);

       {Calculate number of steps elevation}
       nElevationPSRF := round(AzElPSRF.el/TR);
       {Calculate number of steps azimuth}
       nAzimuthPSRF := round(AzElPSRF.az/PR);

       {Set az and el on instrument}
       if Comport1.Connected then
        if ((nAzimuthPSRF <= PX) and (nAzimuthPSRF >= PN) and
                     (nElevationPSRF <= TX) and (nElevationPSRF >= TN)) then
          begin
           {Calculate the motion needed, save current spot}
           nAzimuthPSRF_C := nAzimuthPSRF - CAs;
           nElevationPSRF_C := nElevationPSRF - CEs;
           CAs := nAzimuthPSRF;
           CEs := nElevationPSRF;
           Str(nAzimuthPSRF_C,sAzimuthPSRF);
           Str(nElevationPSRF_C,sElevationPSRF);

           {Send azimuth command}
           donea := false; {New command}
           sendcmd2('FL'+inttostr(nAzimuthPSRF_C));
           WaitOnMotors(abs(nAzimuthPSRF_C));
           donea := true;
           {Send elevation command}
           donee := false; {New command}
           sendcmd1('FL'+inttostr(nElevationPSRF_C));
           WaitOnMotors(abs(nElevationPSRF_C));
           donee := true;
        end
       else
        errorinCmd := True {Angles out of range}
    else
        errorinCmd := True; {COM port not connected}
end;

{Send feedback when command executed.}
```

```
        if ((donee = True) and (errorinCmd = False)) then
             AContext.Connection.Socket.WriteLn(DateTimeToStr(Now) +
                                                    ' FS Ikea: Command executed.')
        else
          if (errorinCmd = True) then
             AContext.Connection.Socket.WriteLn(DateTimeToStr(Now) +
                                                    ' FS Ikea: Error in command.');
      end;
      {End of azimuth and elevation command}
```

Listing F.7 shows the procedure that is called upon the reception of a character on the COM port. All characters received on the port are part of the feedback from the PTU D46 unit in this case. Since the characters arrive with a high rate to the port, they are read as strings, as there is quite a high chance that by the time the procedure is called there are several characters available for reading. All read character-groups are assembled into one string until the phrase sought is contained in the string. The phrase is characteristic to each instrument, and represents the current location of the unit. Once it has been found the variables donea, donee: **boolean**; are set to 'true'.

Listing F.7: Function for obtaining feedback from the motor control unit.

```
{
------------------------------------------------------------------
--------------------FEEDBACK FROM PTU-----------------------------
------------------------------------------------------------------
}
procedure TPTU46TrackerF.ComPort1RxChar(Sender: TObject; Count: Integer);
var
   newlinepos: Integer;
begin
   ComPort1.ReadStr(ComStr, Count);
   TotalCount := TotalCount + Count;
   Reply := Reply + ComStr;
   {Find new lines and delete them from the read sequence}
   newlinepos := Pos(String(#13#10), Reply);
   while ( newlinepos > 0 ) do
     begin
       Delete(Reply, newlinepos, 1);
       newlinepos := Pos(String(#13#10), Reply);
       TotalCount := TotalCount - 1;
     end;
   {If the correct feedback is in the sequence of replies,
   indicate that final position is reached.}
   if ((Pos('Current Pan position is '+sAzimuthPSRF,Reply)>0) and (home=True))
     then
       begin
         donea := True;
         Reply := '';
         TotalCount := 0;
       end;

   if ((Pos('Current Tilt position is '+sElevationPSRF,Reply)>0) and (home=True))
    then
       begin
         donee := True;
         Reply := '';
```

```
          TotalCount := 0;
        end;

    if (((Pos('Current Pan position is 0',Reply)>0) and
                (Pos('Current Tilt position is 0',Reply)>0)) and
                (donea=True) and (donee=True))
      then
        begin
          home := True;
          Reply := '';
          TotalCount := 0;
        end;

end;
```

Bibliography

Berlin, P. (2007), *Satellite platform design*, Department of Space Science, University of Luleå, Kiruna, Sweden.

Bernhardsen, T. (1992), *Geographic Information Systems*, Viak IT/Norwegian Mapping Authority.

Brekke, A. (2012), *Physics of the upper polar atmosphere*, Springer.

Dandekar, B. (1974), 'Simplified formula for the determination of the altitude of an auroral arc from a single station', *Journal of Atmospheric and Terrestrial Physics* **36**(5), 829–834.

Directed Perception (2007), 'Ptu-d46 family: Miniature pan-tilt units'.

Frey, H., Frey, S., Lanchester, B. & Kosch, M. (1998), Optical tomography of the aurora and EISCAT, *in* 'Annales Geophysicae', Vol. 16, Springer, pp. 1332–1342.

Fujinon Fujifilm (2009), 'Cctv lens catalogue'.

Geographic information - Spatial referencing by coordinates (2003), Technical Report ISO 19111:2003(E), Swedish Standard Institute.

Hager, J. W., Fry, L. L., Jacks, S. S. & Hill, D. R. (1992), Datums, ellipsoids, grids, and grid reference systems, Technical report, DTIC Document.

Hofmann-Wellenhof, B., Lichtenegger, H. & Collins, J. (2001), *GPS, Theory and Practice. Fifth, revised edition.*, SpringerWienNewYork.

Hower, C. Z. & Indy Pit Crew (2006), 'Internet direct (indy) - an open source suite of internet components.'. Version 10.1.5.

Kaila, K. (1987), 'An iterative method for calculating the altitudes and positions of auroras along the arc', *Planetary and space science* **35**(2), 245–258.

Kannala, J. & Brandt, S. (2004), A generic camera calibration method for fish-eye lenses, *in* 'Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on', Vol. 1, IEEE, pp. 10–13.

Kee, C. & Parkinson, B. W. (1996), 'Wide area differential gps (wadgps): Future navigation system', *Aerospace and Electronic Systems, IEEE Transactions on* **32**(2), 795–808.

KHO (n.d.*a*), 'Hyperspectral tracker (fs-ikea)', http://kho.unis.no/FS_IKEA_Spesification.htm. Accessed: 2014-07-01.

KHO (n.d.*b*), 'Narrow field of view scmos cameray', http://kho.unis.no/sCMOS_Tracker_Spesification.htm. Accessed: 2014-07-01.

KHO (n.d.*c*), 'Practical information about the observatory', http://kho.unis.no/nordlysstasjon_info.htm. Accessed: 2014-07-01.

KHO (n.d.*d*), 'Related documents', http://kho.unis.no/nordlysstasjon_doc.htm. Accessed: 2014-07-22.

KHO (n.d.*e*), 'Tracker camera', http://kho.unis.no/Track_Camera.htm. Accessed: 2014-07-01.

Krishnan, G. & Nayar, S. K. (2008), Cata-fisheye camera for panoramic imaging, *in* 'Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on', IEEE, pp. 1–8.

Mahoney, M. (n.d.), 'Pointing an instrument on an airborne platform', http://mtp.mjmahoney.net/www/notes/pointing/pointing.html. Accessed: 2014-05-09.

Muller, R., Lehner, M., Muller, R., Reinartz, P., Schroeder, M. & Vollmer, B. (2002), 'A program for direct georeferencing of airborne and spaceborne line scanner images', *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences* **34**(1), 148–153.

Norwegian Polar Institute (n.d.), 'Toposvalbard'. http://toposvalbard.npolar.no/.

Rees, M. (1963), 'A method for determining the height and geographical position of an auroral arc from one observing station', *Journal of Geophysical Research* **68**(1), 175–183.

Sharp, W., Rees, M. & Stewart, A. (1979), 'Coordinated rocket and satellite measurements of an auroral event, 2. the rocket observations and analysis', *Journal of Geophysical Research: Space Physics (1978–2012)* **84**(A5), 1977–1985.

Sigernes, F. (2000), 'Direct georeferencing'. (Fragment of lecture notes from UNIS course AGF-331 Spectroscopy and remote sensing).

Sigernes, F., Dyrland, M., Brekke, P., Chernouss, S., Lorentzen, D. A., Oksavik, K. & Deehr, C. S. (2011), 'Two methods to forecast auroral displays', *Journal of Space Weather and Space Climate* **1**(1), A03.

Sigernes, F., Lorentzen, D. A., Heia, K. & Svenøe, T. (2000), 'Multipurpose spectral imager', *Appl. Opt.* **39**(18), 3143–3153.

Sigernes, F., Moen, J., Lorentzen, D., Deehr, C., Smith, R., Øieroset, M., Lybekk, B. & Holtet, J. (1996), 'Scifer-height measurements of the midmorning aurora', *Geophysical research letters* **23**(14), 1889–1892.

Strømer, C. (1955), *The Polar Aurora*, Oxford University Press.

Swan, T. (1998), *Delphi 4 Bible*, IDG Books Worldwide, Inc.

Terry Jr, N. (1996), 'How to read the universal transverse mercator (utm) grid', *GPS World* **32**.

Thorlabs (2011), 'V21 photonics catalog'.

Wasmeier, P. (2006), 'Geodetic transformation toolbox', http://www.mathworks.se/matlabcentral/fileexchange/9696-geodetic-transformations-toolbox. Accessed: 2014-05-09.

Whiter, D. K., Gustavsson, B., Partamies, N. & Sangalli, L. (2013), 'A new automatic method for estimating the peak auroral emission height from all-sky camera images', *Geoscientific Instrumentation, Methods and Data Systems* **2**(1), 131–144.