Recovering Data with Digital Photogrammetry and Image Analysis Using Open Source Software

1936 Aerial Photographs of Svalbard

Niels Ivar Nielsen

Thesis Submitted for the Degree of Master in Geosciences (Physical Geography, Hydrology and Geomatics) 60 credits

Department of Geosciences
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2017
Recovering Data with Digital Photogrammetry and Image Analysis Using Open Source Software

1936 Aerial Photographs of Svalbard

Niels Ivar Nielsen
© 2017 Niels Ivar Nielsen

Recovering Data with Digital Photogrammetry and Image Analysis Using Open Source Software

Supervisors:
1. main supervisor: Christopher Nuth*
2. co-supervisor: Luc Girod*
3. co-supervisor: Andreas Kääb*

* University of Oslo, Faculty of Mathematics and Natural Sciences, Department of Geosciences, Section for Geography and Hydrology

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo
Abstract

Old images are a valuable source of information in Geosciences. They provide information on the state of an object or phenomenon at a given time. Datasets of the past are produced according to old data standards optimized for analog management and analysis. Modern analysis and processing methods in digital systems are able to extract more information than what was possible earlier, but old formats and data standards make the transition to digital systems somewhat complicated.

This thesis will present a workflow for creating orthophotos and digital elevation models from a set of high-oblique aerial images of Svalbard. Digital image analysis methods are applied to the images to locate fiducial marks and mask terrain features. Fiducial mark locations are used to calculate inner orientation for the images while terrain feature masks remove areas of the image that contain water, clouds or sky. Structure from Motion is applied to the images to generate dense georeferenced point clouds. The last processing step generates orthophotos and digital elevation models that are useful for geoscientific analysis.

Through the use of well known and transparent methods, with solid foundations in scientific literature, and by using only open source software, the work can be tested, reviewed and used by others.
Acknowledgment

Creating this thesis would not have been possible without help and support from others. My supervisors at UiO, Chris, Luc and Andreas have helped me throughout the process with everything from providing data to advice on methods. My employer, Forsvaret, has given me the flexibility and freedom to pursue my interest alongside my normal tasks. My life companion, Veronica, has supported me through the process and not complained about my habit of working at night.

Thank you!
Contents

I Introduction  
1 Background 2  
2 Problem Formulation 4  
3 Study Area 5  
4 Data 7  
  4.1 Aerial Photographs from 1936 and 1938 . . . . . . . . . . . . . . . . . . . . 7  
  4.2 Map Data from The Norwegian Polar Institute . . . . . . . . . . . . . . . 10  
  4.3 TanDEM-X . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11  

II Computational Theory and Software 13  
5 Theory 14  
  5.1 Thresholding . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14  
  5.2 Histogram Equalization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15  
  5.3 Image Filtering . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16  
  5.4 Corner Detection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22  
  5.5 Machine Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24  
  5.6 Structure from Motion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27  
6 Software 33  
  6.1 Python . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33  
  6.2 Jupyter Notebook . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33  
  6.3 SciPy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34  
  6.4 OpenCV . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34  
  6.5 dlib . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35  
  6.6 MicMac . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35  
  6.7 QGIS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35  
  6.8 GDAL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35  
  6.9 Docker . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36  
  6.10 PDAL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36  
  6.11 CloudCompare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36  
  6.12 ImageMagick . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
### III Processing Svalbard 1936 Images

7 Processing Workflow 38

8 Find Fiducial Marks 40
   8.1 Svalbard 1936 Aerial Photographs 40
   8.2 Detection 41
   8.3 Prediction 45
   8.4 Correction 57
   8.5 Output and Performance 60

9 Sparse Point Cloud Creation 65
   9.1 Georeferencing and Bundle Adjustment 66
   9.2 Adjusted Camera Positions 71

10 Masking Terrain Features 73
   10.1 Pre-processing 73
   10.2 Entropy Mask 75
   10.3 Post-processing 78
   10.4 Mask Quality 81

11 Dense Point Cloud Creation 84

12 Point Cloud Processing 89
   12.1 Point Cloud Cleaning 89
   12.2 Raster Creation 93
   12.3 Raster Mosaicing 94

### IV Results 97

13 Complete Models of Barentsøya 98

14 Quality of the Results 102
   14.1 Errors and Artifacts 102
   14.2 Accuracy and Precision 104
   14.3 Overall Quality Assessment 111

### V Conclusion 113

15 Workflow Review 114

16 Software Review 117

17 Conclusion 120
Part I

Introduction
1 Background

In Geosciences accurate terrain data over the same area at different times is important. It enables researchers to accurately describe the situation at a given time and establish time series to investigate how conditions change. This gives valuable insight into the past and may enable prediction on how things will evolve in the future.

The methods for gathering data and establishing time series vary depending on the field of research and what data is available. Some collect rock samples and apply advanced dating technology while others drill into glaciers and examine the properties of ancient ice. Others use photography to document the world. Images can contain a lot of information and effectively describe the visual characteristics of an object or phenomenon. In addition to providing visual descriptions, images can be analyzed and processed to extract additional or more specific information.

One such method is photogrammetry or the science of making measurements from photographs (Dick 2015). A photography is a two-dimensional representation of a three-dimensional scene. Having two or more images of the same scene, taken from different angles, makes it possible to project the two-dimensional images into the three-dimensional scene giving a stereo model which enables measurements in three dimensions. In Geosciences photogrammetry is useful for creating digital elevation models (DEM) and images with orthogonal projection, or map properties, known as orthophotos or orthoimages (Ortofoto 2015).

![Figure 1.1: The worlds first photography, View from the Window at Le Gras, taken by Joseph Nicéphore Niépce (Niépce, c.1826) (Ahlsen 2017) Creative Commons Public Domain]
In the nearly 200 years that have passed since the first photography was taken (Ahlsen 2017) photography has evolved into a popular way of documenting events and phenomenon. Cameras and the quality of the photos have evolved a lot since they were first introduced. The bulky cameras of the past required good conditions and often produced poor results compared to modern cameras that can be mounted anywhere and produce images with extremely high quality and resolution.

Along with cameras the data standards also evolved. The images and data standards developed for analog processing are not always easy to use in digital systems. Modern photogrammetry methods are able to extract more information from images than in the past. Being able to efficiently use old data with modern methods to produce DEMs and orthophotos give valuable insight into the past and extend the time series further back in time.

In Geosciences old photographs are a valuable source of information and have been used in numerous studies. As an example a set of oblique aerial images from the 1930s were used to map the extent of glaciers along the southeastern margin of Greenland using photogrammetry (Bjørk et al. 2012, Kjeldsen et al. 2015, Korsgaard et al. 2016).

This thesis will focus on a set of high-oblique aerial images of Svalbard taken in 1936 and 1938. They were originally intended for creating topographic maps of Svalbard in scale 1:50 000 (Luncke 1936). In addition to creating topographic maps they have been used in numerous other studies over the years (Lefauconnier and Hagen 1991, Nuth 2006, Nuth, Kohler, et al. 2013), but the methods applied are often manual and time consuming.

In 2016, the internship report 1936 oblique aerial images : Spreading the time series of Svalbard glaciers (Couderette 2016) was produced at the Department of Geosciences, UiO. The report presents a workflow for creating orthophotos and digital elevation models from the 1936 and 1938 photographs of Svalbard. The workflow produces good results, but it involves some manual and time consuming steps. This thesis will build upon the internship report and present a more automated processing workflow.
2 Problem Formulation

The 1936 and 1938 images of Svalbard have been studied extensively, but little effort has been put into automatic processing. The internship report *1936 oblique aerial images: Spreading the time series of Svalbard glaciers* (Couderette 2016) is a good starting point, but the manual processing steps involved are time consuming. Robust and well known methods in digital photogrammetry and image analysis combined with sufficient processing power have the potential to make the process more effective. The problem statement for this thesis is therefore:

> Derive an automated workflow for creating orthophotos and digital elevation models from the 1936 and 1938 high-oblique aerial images of Svalbard

The method will be tailored to this specific dataset, but it should be general enough so it can be applied to other datasets with some modification. Through the use of well known and transparent methods, with solid foundations in scientific literature, and by using only open source software, the work can be tested, reviewed and used by others.
3 Study Area

Barentsøya, in the south east part of Svalbard, is chosen as the study area for this thesis. It lies in the coldest and most isolated part of Svalbard and is one of the least studied parts of the archipelago. The island, Barentsøya, has a total area of $1298 \text{ km}^2$.
where $610km^2 (47\%)$ is covered by the glacier Barentsjökulen. The large and complex Barentsjökulen is drained by a number of outlet glaciers which are known to surge. Freemanbreen, Duckwitzbreen and Besselbreen are the largest. (Dowdeswell and Bamber 1995).

The area not covered by ice consists of different types of landscape. Along the coast and up to approximately 100 meters above sea level there is coastal lowland which is accumulated sediment from the Pleistocene glaciations. In some areas along the coast there are also well-developed raised beaches formed by isostatic uplift. Further inland there is platou mountainous landscape with scattered moraine fields. (Dallmann 2015). Barentsøyas highest point is Solveigdomen at 666 meters (Norwegian Polar Institute 2016a).

Barentsøya has very few man made features. The only man made feature identified is the Würzburger hut$^1$ at Sundneset on the southwestern tip of Barentsøya.

---

$^1$UTM 33X E638100 N8688540
4 Data

4.1 Aerial Photographs from 1936 and 1938

Figure 4.1: Examples of the 1936 high-oblique aerial images of Svalbard showing Duckwitzbreen on the east coast of Barentsoya. The slight difference in angles make it possible to apply photogrammetry methods.

In July and August of 1936 a Norwegian expedition lead by Docent Adolf Hoel went to Svalbard to photograph the area from the air. Bernhard Luncke was in charge of the planning and acquisition of aerial photographs. Their goal was to collect images suitable for creating topographic maps of Svalbard in 1:50 000 scale and contour interval of 50m. Good weather conditions and few problems resulted in a total effective flight time of 89 hours. 3300 images covering an area of 40 000km$^2$ or nearly $2/3$ of the total area of Svalbard were taken. The areas covered were Barentsoya, Edgeoya and most of western Spitsbergen. Figure 4.2 shows where aerial photographs were taken during the 1936 expedition. (Luncke 1936)

In the summer of 1938 a second expedition lead by Docent Adolf Hoel and with Bernhard Luncke as aerial photographer went to Svalbard. Their goal was to photograph the remaining areas that were not covered in 1936. Bad weather conditions and technical problems prevented them from completing all the planned acquisition, but 48 hours of flight time resulted in 2178 photographs covering a total area of about 25 000km$^2$. (“Norwegian Expedition to Svalbard, 1938” 1939)

In total the two expeditions in 1936 and 1938 resulted in 5478 high-oblique aerial photographs that cover most of Svalbard. The photographs are owned and manged by...
the Norwegian Polar Institute.

4.1.1 Technical specifications

The Norwegian Navy provided the expedition with a M F 11 scout plane with crew. A Zeiss camera was mounted in a stabilizing cradle at the side of the plane. More information about the camera is given in table 4.1. The photographs were taken at an altitude of 3000m along the coast and 3500m inland. The optical axis of the camera was pointed 20° below the horizon, or 70° off the vertical axis, making them high-oblique. (Luncke 1936)

In addition to the camera specifications from table 4.1 it is necessary to know some information about the position of the fiducial marks on the image negative. Figure 4.3 gives us the distances between them while table 4.2 gives their position in millimeters on the film.
4.1.2 Available data

A total of 5478 images of Svalbard were taken during the missions in 1936 and 1938. A subset of 857 scanned images covering Barentsøya, Brøggerhalvøya, Edgeøya and Prins Karls Forland were available for this thesis. The main focus in this thesis is the images covering the study area Barentsøya (chapter 3).

Along with the images there is also a rough description of camera position \((x, y, z)\), camera orientation and flight direction.
Table 4.2: Position of fiducial marks in Svalbard 1936 image negatives. Coordinate system origin is top-left corner and coordinates are given as (horizontal, vertical) in millimeters. (Luc Girod 2016)

<table>
<thead>
<tr>
<th>Fiducial mark</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td>89.80, 3.10</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>89.47, 177.90</td>
</tr>
<tr>
<td>RIGHT</td>
<td>176.74, 90.40</td>
</tr>
<tr>
<td>LEFT</td>
<td>2.20, 90.20</td>
</tr>
</tbody>
</table>

4.2 Map Data from The Norwegian Polar Institute

The Norwegian Polar Institute\(^1\) offers an extensive list of map data and services that are free to use as long as their terms of use\(^2\) are read and accepted. All datasets and services are available through their web portal\(^3\).

These datasets are used in this thesis:

- Svalbard digital elevation models with 5, 20 and 50 meter resolution. Different resolutions are available in different areas. Only the 20 meter model is available for Barentsøya. It has an expected standard deviation of 25 meters in the vertical direction. The 20 meter model was used to extract elevation for Ground Control Points (GCPs) and to compare the digital elevation model produced from Svalbard 1936 imagery. (Norwegian Polar Institute 2016b) (S0_DTM20 version: April, 2016)

- Svalbard 1:100 000 basemap themes. Topographic maps in vector and raster formats covering Svalbard in 1:100 000 scale. It is used for presentation and navigation in GIS tools as well as collection of GCPs. (Norwegian Polar Institute 2016a) (S100 version: April, 2016)

- Svalbard Basemap WMTS. Web service with basemaps over Svalbard. It is mainly used for presentation and navigation in GIS tools. (Norwegian Polar Institute n.d.[a])

- Svalbard Satellite Imagery WMTS. Web service with satellite imagery over Svalbard. The service contains USGS Landsat 8 satellite imagery. They are pansharpened color images with 15 meter ground resolution. Landsat 8 meet the Level 1

\(^1\)http://www.npolar.no/en/
\(^2\)http://geodata.npolar.no/bruksvilkar/
\(^3\)https://geodata.npolar.no/
system requirement of 65 meters at a circular error with 90% confidence. The satellite imagery was used to pick out GCPs. (Norwegian Polar Institute n.d.(b), Landsat 8 (L8) Data Users Handbook - version 2.0 2016)

4.3 TanDEM-X

In 2010 the two twin satellites TanDEM-X and TerraSAR-X flew in a closely controlled formation with a baseline of between 100 and 500 meters to generate a global elevation dataset using single-pass SAR interferometry. (Wessel 2013)

A digital elevation model with 12 meter resolution covering Barentsøya was available through Department of Geosciences, UiO. According to the high error map (HEM) of the dataset it has a maximum error of 19.79857 meters, while the mean error is 1.05785 meters with a standard deviation of 1.00012 meters. The HEM is expressed as a linear error at 90% confidence. The TanDEM-X elevation model has ellipsoidal heights. (Wessel 2013)

This dataset became available very late in the process and has not been used for anything else than checking the result.
Part II

Computational Theory and Software
5 Theory

A brief overview of the main image analysis algorithms and methods used for data processing in part III. This chapter’s function is to serve as a theoretical baseline for part III. Basic theory is presented here while practical application and some specific theory is presented within the workflow in part III.

5.1 Thresholding

Thresholding is the process of classifying pixels in a digital image into two or more classes based on their intensity values. Thresholding can be performed globally on the entire image or locally by using a sliding window approach. The basic idea is the same for local and global approaches. A intensity threshold value $T$ is chosen and pixel values are classified according to this. For a simple threshold of the image $f$ into the two classes 0 and 1 a basic threshold formula is used. (Gonzalez and Woods 2010)

$$g(i,j) = \begin{cases} 1 & \text{if } f(i,j) > T \\ 0 & \text{if } f(i,j) \leq T \end{cases} \quad (5.1)$$

This basic thresholding formula can easily be extended with more classes by adding more intensity threshold values.

The most difficult part of thresholding is to find the correct intensity value $T$. It can be picked out manually or calculated based on the image. One popular approach for local or moving window approaches is to use the mean value of the moving window. This will for some images remove the effects of different lighting conditions in the scene. (Gonzalez and Woods 2010)

For global thresholding simply choosing the mean value of the image will not be good enough. There are simple and more complicated solutions to this problem. One very popular approach is to find the optimum global threshold using Otsu’s method. This is done with an iterative process to maximize the between-class variance based on the assumption that the optimum threshold is found when the separation between classes intensity values is largest (Otsu 1979). Otsu’s method can be used on two classes or extended to many classes, but for more than three classes or two threshold values other methods should be considered (Gonzalez and Woods 2010).
5.2 Histogram Equalization

Histogram equalization is a method for globally increasing the contrast of an image. This is done by spreading the intensity values in the image across all available intensity values by using the cumulative distribution function (cdf). The most frequent intensity values will be spread the most.

In a digital image $x$ the probability of encountering a pixel with intensity value $i$ is

$$p_x(i) = \frac{n_i}{n} \quad (5.2)$$

where $n$ is the total number of pixels and $n_i$ is the number of pixels with intensity value $i$. Based on the probability the cdf of the normalized histogram can be defined.

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j) \quad (5.3)$$

The cdf can be used to map intensity values to their equalized values creating a linear cdf in the equalized image, but the cdf returns values between 0 and 1 so the result must be multiplied with the number of intensity values $L$ in the image minus 1.

$$n_x(i) = (L - 1) cdf_x(i) \quad (5.4)$$

$n_x$ is the new intensity value in equalized image. (Gonzalez and Woods 2010, Histogram equalization 2017, Szeliski 2011)

Figure 5.1 illustrates how cdf is used to assign new intensity values to pixels in the equalized image. Observe that the cdf of the equalized image is linear.
Figure 5.1: Histogram Equalization using cumulative distribution function. Note that \( cdf \) and histogram share the x-axis, but have separate y-axis. Kongsbreen, Kronebreen and Kongsvegen, 1936.

5.3 Image Filtering

Image filters are techniques to enhance or suppress certain properties of an image. Usually filters are divided into low-pass, high-pass and band-pass filters based on the type of frequencies that are allowed through the filter. Low-pass filters preserve low frequencies and will have a blurring effect on the image. High-pass filters will preserve high frequencies and therefore highlight edges and other steep gradients in intensity. Band-pass filters are designed to only let certain frequencies through and is useful for filtering out or keeping only certain frequencies in an image. (Gonzalez and Woods 2010)

Filters come in many forms and are often specially tailored to extract certain properties from a image. This section will present some basic concepts that are relevant to
5.3.1 Linear Filters

The linear filter is a simple neighborhood operator which uses a fixed kernel or mask to calculate the sum of products of the pixel neighborhood. Calculating the filtered image \( g(i,j) \) from an image \( f(i,j) \) and a kernel \( h(k,l) \) is done by using the correlation or the convolution operator. Correlation is the sum of products in the neighborhood defined by the kernel which slides across the image.

\[
g(i,j) = f(i,j) \ast h(k,l) = \sum_{k,l} f(i + k, j + l)h(k,l)
\]  

(5.5)

Convolution is very similar, but the kernel is flipped.

\[
g(i,j) = f(i,j) \bigstar h(k,l) = \sum_{k,l} f(i - k, j - l)h(k,l) = \sum_{k,l} f(k,l)h(i - k, j - l)
\]  

(5.6)

For symmetric kernels the result will be exactly the same with correlation and convolution. Equation 5.7 shows a simple average filter.

\[
h = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]  

(5.7)

Convolution is the most commonly used operator because it has some nice properties like being commutative, associative and the Fourier transform of two convolved images is the product of their individual Fourier transforms.

In image processing literature convolution often refers to the sliding window sum of products process and does not necessarily separate between correlation and convolution. (Szeliski 2011, Gonzalez and Woods 2010)

Linear filters are very useful for for many operations on images. They are often used for blurring, removing noise, sharpening details, calculating intensity gradient or finding edges. The filter properties depends on kernel design and are often separated into the three types high-pass, low-pass and band-pass filters which refer to the types of frequencies they preserve. Figure 5.2 shows image blurring with a low-pass filter and image edge detection with a high-pass filter. The cone filter shown in figure 8.12 is a linear band-pass filter because it searches for circles with a given radius (frequency).
5.3.2 Non-Linear Filters

While linear filters use fixed kernels non-linear filters performs various calculations within the sliding window. Morphological operators like the ones explained in section 5.3.3 are examples of non-linear filters that calculate maximum and minimum values. Median and bilateral (Tomasi and Manduchi 1998) filters are other examples. They are similar to their linear counterparts average and Gaussian blur, but they have slightly different properties and can sometimes perform better. Non-linear filters are more complex than linear filters and take more time to compute (Szeliski 2011).

5.3.3 Morphological Image Processing

Morphology is a branch of biology that deals with the form and structure of plants and animals. In image processing it is used to describe algorithms or transformations that are used to extract components that describe or represent region shape. (Gonzalez and Woods 2010)

Morphology in image processing is done by sliding a kernel or structuring element with specific properties across the image and calculating output pixel values. Structuring elements or kernels are arrays and can have any extent, shape and combinations of values to achieve the wanted result, but flat binary kernels with circular or rectangular shapes with origin in the center value (figure 5.3) are most common.

Morphology can be performed on binary or gray-level images. This section will explain the basic method that will work on both binary and gray-level images with the morphological operations erosion, dilation, opening and closing.
Erosion and Dilation

Erosion and dilation are the most basic morphological operations. For an image $f$ and a flat structuring element $b$, the erosion result in pixel $(i, j)$ is the minimum value within the the structuring element $b$ when it is convolved with the image $f$.

$$[f \ominus b](i, j) = min\{f(b)\} \quad (5.8)$$

The dilated result is the maximum value within the structuring element.

$$[f \oplus b](i, j) = max\{f(b)\} \quad (5.9)$$

Figure 5.4 shows erosion and dilation performed on a binary image. Erosion will reduce white areas while dilation will increase the white areas of a binary image. Erosion will remove white noise particles while dilation will remove black noise particles. (Gonzalez and Woods 2010)

Figure 5.3: Morphological kernels with origin in bold value

Figure 5.4: Morphological Erosion and Dilation performed on focal length annotation in the top-right corner of a Svalbard 1936 image.
Opening and Closing

Morphological opening and closing are algorithms that are combinations of erosion and dilation. The opening of an image \( f \) with a kernel \( b \) is a erosion followed by a dilation. Morphological opening can used to remove white noise or separate feature parts that are linked together.

\[
f \circ b = (f \ominus b) \oplus b
\]

Closing is a dilation followed by a erosion. It can remove black noise or connect feature parts that are separated.

\[
f \mathcal{Y} b = (f \oplus b) \ominus b
\]

Figure 5.5 shows how a input binary image responds to morphological opening and closing. One important attribute of opening and closing is that features without large artifacts or noise will not be altered in the process. This is not the case for erosion and dilation. (Gonzalez and Woods 2010)

![Image](image.png)

(a) Original image  (b) Opened image  (c) Closed image

Figure 5.5: Morphological Opening and Closing performed on focal length annotation in the top-right corner of a Svalbard 1936 image.

5.3.4 Fourier Transform

The Fourier transform states that all functions whose area under the curve is finite can be expressed as the integral of sines and/or cosines multiplied by a weighting function. The inverse Fourier transform can reconstruct the function again without loss of information. The Fourier transform works in continuous and discrete domains and in multiple dimensions. The 2D discrete Fourier transform (DFT) enables us to transform digital images from the spatial domain to the frequency domain and back again. (Gonzalez and Woods 2010)

The DFT is too computationally intensive for practical use, but the discovery of the
fast Fourier transform (FFT) (Brigham 1988) enables efficient transform of digital images between the spatial and frequency domains. The terms FFT and DFT are often used to describe the same operation of transforming finite discrete data into the frequency domain and back again.

When a digital image is transformed to the frequency domain with the FFT the result will be a matrix with the same size as the input image where the axis represent frequencies in horizontal and vertical directions. Every element in the matrix is a complex number representing magnitude and phase for the given combination of horizontal and vertical frequency. In the frequency domain frequency filters are applied as simple masking operations. The Fourier transform of a digital image have a lot of nice properties like the convolution theorem mentioned in section 5.3.1, but they will not be further discussed here.

Figure 5.6 is an example of a band-pass filter implemented with FFT by removing certain frequencies from the image in frequency domain.

![Figure 5.6: Input image (a) with banding effect from digital scanner is transformed into frequency domain (b) where horizontal line representing banding frequency is removed (c) before transforming back to spatial domain (d). Ny-Ålesund, 1936.](image)

**5.3.5 Entropy Filters**

In information theory entropy is the probability that a given element occurs in a population. It is a measure of disorder or uncertainty defined by Shannon 1948 and used to find the number of bits needed to transmit a message. The entropy $H$ of a population of $n$ elements is

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i , \text{ } i = 1, 2, ..., n , \sum_{i=1}^{n} p_i = 1$$

(5.12)

where $p_i$ is the probability that $i$ occurs.

In digital image processing entropy has been widely used in thresholding and object-
background segmentation (Pun 1980, Kapur, Sahoo, and Wong 1985, N. Pal and S. Pal 1989). When entropy measures is applied to a digital image the entropy of every pixel is given by the uncertainty or disorder within a neighborhood around the pixel given by a kernel (figure 5.3). The uncertainty or disorder of the neighborhood can be seen as the amount of texture and how ordered it is. Areas with little texture or ordered texture will give low entropy while areas with unordered texture will give high entropy. These properties are useful for segmenting background and foreground information in images. Figure 5.7 shows an entropy filter applied to an image. One can observe that the different responses to terrain types. Water, snow and clouds get low values while glaciers and mountains get high values.

Figure 5.7: Entropy filter applied to image. Different responses to different terrain types can be observed. Kronebreen, 1936.

5.4 Corner Detection

In digital images corners and edges can be defined as changes in intensity. For a edge intensity will change when moving perpendicular to the edge, but not when moving along the edge. Corners can be seen as intersections of edges and intensity will change when moving in any direction. Based on these assumptions and earlier work by other, Harris and Stephens 1988 introduced a corner detector. The algorithm finds the change in
intensity $E$ when a shift $(x, y)$ in all directions is performed.

$$
E(x, y) = \sum_{u,v} w(u, v)[I(x + u, y + v) - I(u, v)]^2
$$

(5.13)

Where $w_{u,v}$ is a Gaussian kernel

$$
w(u, v) = e^{-\frac{u^2 + v^2}{2\sigma^2}}
$$

(5.14)

Using Taylor expansion this can be written

$$
E(x, y) = \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix}
$$

(5.15)

$$
M = \sum_{u,v} w(u, v) \begin{bmatrix} I_xI_x & I_xI_y \\ I_xI_y & I_yI_y \end{bmatrix}
$$

(5.16)

$I_x$ and $I_y$ are the gradients or partial derivatives in $x$ and $y$ directions and can be found using a gradient filter like the Sobel operator (Sobel and Feldman 1968) which is the same method as Horn 1981 uses for finding gradients of surface models.

$$
I_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad I_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}
$$

(5.17)

Harris found that analyzing the eigenvalues $\alpha$ and $\beta$ of $M$ is sufficient to find corners. As shown in figure 5.8 one large eigenvalue is an edge while two large eigenvalues is a corner.

To avoid calculating the eigenvalues, a response function $R$ is formulated where $\kappa$ is a sensitivity parameter.

$$
R = |\text{det}(M)| - \kappa \text{trace}(M)^2
$$

(5.18)

The iso-response contours in figure 5.8 show what type of features can be expected from different values of $R$. This makes it easy to threshold corner responses of a given certainty.
5.5 Machine Learning

Machine learning is defined as an automated process that extracts patterns from data. In predictive data analysis a concept called supervised machine learning is applied to a training dataset to model it. The training dataset is a set of data instances that have one or more descriptive features and a target feature. Descriptive features are characteristics that describe some instance while the target feature is the correct answer for that instance. The training dataset is then fed to a machine learning algorithm which establishes a prediction model. Real data containing descriptive features, but no target feature, is then fed into the prediction model which will predict the target feature. (Kelleher, Mac Namee, and D’Arcy 2015)

Machine learning algorithms exist for many kinds of data. A simple example will briefly explain how a zebra finding machine can be trained using both tabular and image data. The tabular and image descriptions of a zebra are two independent ways to train a predictor and can be seen in figure 5.9.

The tabular data contains the descriptor features legs and stripes and the target feature zebra. Using the two descriptor features and the target feature a predictor is trained. When data with no target features are fed to the predictor, it will know that an instance (animal) with legs = 4 and stripes = true is a zebra. Unfortunately a tiger will have the same descriptors and is falsely classified as zebra. A robust predictor will need more descriptors in the training dataset.

For the image training dataset the target feature is the label ZEBRA on the yellow
(a) Tabular data

<table>
<thead>
<tr>
<th>legs</th>
<th>stripes</th>
<th>zebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

(b) Image data. Photo: Niels Ivar Nielsen, 2010

Figure 5.9: Zebra training data examples.

rectangle while everything inside the rectangle are descriptor features. The individual pixels themselves do not make good descriptors. They must be processed in some way. One descriptor can be a histogram containing the color or gray-level distribution of the normalized image in a color space. Another common descriptor is to calculate gradients and their orientation to create a Histogram of Oriented Gradients (HOG) descriptor. There are numerous different descriptors that can be implemented to represent specific characteristics of an image.

To train predictors with training data a prediction model must be applied. The common prediction models *k Nearest Neighbor*(kNN) and *Support Vector Machine*(SVM) will be explained in the sections below.

5.5.1 kNN

*k Nearest Neighbor*(kNN) is a prediction model which calculates similarity based on distance in a n-dimensional feature space. The descriptor features of the training dataset are converted into dimensions in feature space where the target feature is the label. Each instance in the training dataset will have a coordinate in the n-dimensional feature space. A feature space populated with the training dataset is the prediction model. When data is fed into the prediction model it will be assigned a coordinate based on its descriptors. Distance is measured between the new instance and all instances in the training dataset. The new instance is assigned the target feature from the majority of the k nearest instances in the training dataset. The k parameter controls the number of nearest neighbors used for finding the target feature. (Kelleher, Mac Namee, and D’Arcy 2015) Figure 5.10 is a example of a two-dimensional kNN prediction model.
Figure 5.10: A two-dimensional kNN prediction model with $k = 3$ and feature space axis $a$ and $b$. The training dataset containing the two target features blue squares and red triangles have descriptor features $a$ and $b$. The new instance green circle is placed in feature space based on its descriptors (a). Distance to all training instances is calculated and $k$ Nearest Neighbors are identified (b). The new instance is assigned the target feature from the majority of its $k$ Nearest Neighbors (c).

5.5.2 SVM

Support Vector Machine (SVM) is a error-based predictive model. It is based on finding a decision boundary between the target features in the training dataset. With two descriptive features it will be a line and with more than two descriptors it will be a hyperplane. The training instances closest to the decision boundary are known as support vector and the perpendicular distance from the decision boundary to the support vectors is known as the margin. The SVM algorithm searches for the decision boundary with the largest margin. The largest margin will give the best separation between target features and the best predictive model. New instances will be assigned a target feature based on the decision boundary. (Kelleher, Mac Namee, and D’Arcy 2015) Figure 5.11 is an example of a two-dimensional SVM prediction model.
5.6 Structure from Motion

Structure from Motion is the process of matching a series of independent photos and creating a 3D point cloud. The process uses feature matching and triangulation and does not require the 3D position or pose of the cameras or a series of control points like traditional photogrammetric methods (Westoby et al. 2012). This section will present a simplified explanation of the basic concept of SfM and how the processing is done. Szeliski 2011 gives a more thorough explanation of the concepts mentioned in this section.

5.6.1 Feature Matching

The first step in SfM is feature matching. Finding the same features in multiple images is the basis for all further calculations. Using the scale-invariant feature transform algorithm (SIFT) on every image it is possible to describe features in an image so that it is possible to find them in another image. The SIFT algorithm can be simplified into these steps (Lowe 2004):

1. Find keypoints in the image by convolving a DoG(Difference of Gaussian) filter (figure 5.2c) with the image at different scales and finding local extrema. The DoG filter is a simplified and more effective implementation of the LoG(Laplacian of Gaussian).
2. For every keypoint canonical scale is calculated by cycling through scales and picking the scale where the DoG function produces the highest score.

3. Extract a patch around each keypoint. The size of the patch is determined by the canonical scale of the keypoint.

4. Calculate orientation of the patch based on the gradients. The largest bin in the HOG (Histogram of Oriented Gradients) is the orientation of the patch.

5. Normalize the patch to scale and orientation.

6. Resize the normalized patch to 16x16 pixels and compute the gradients of the patch. Divide the patch into 16 4x4 squares and compute a 8 bin HOG for each square. Concatenate the HoG’s into one 128 dimensional feature vector.

SIFT is the most common descriptor, but there are numerous alternatives like Speeded Up Robust Features (SURF) (Bay, Tuytelaars, and Van Gool 2006) and Oriented FAST and Rotated BRIEF (ORB) (Rublee et al. 2011). These alternative descriptors will not be covered in this text. A simple example of SIFT in action is shown in figure 5.12.

After having calculated descriptors for every image the matching process can begin. For most scenes with an unordered collection of images it is normal to check every image against each other. For large scenes it might be necessary to group images before starting the matching process (Heinly et al. 2015, Crandall et al. 2013, Klingner, Martin, and Roseborough 2013).

Descriptors for points in one image are compared to descriptors in another. The descriptors are compared by calculating the euclidean distance between them in feature space. Simply picking the closest point or applying a global threshold will likely produce many false positive (FP) matches (Lowe 2004).

The number of false positives can be reduced by applying nearest neighbor matching. This is done by calculating the ratio between the distance to the closest point and the second closest point. All matches with ratio larger than a given threshold are rejected. According to Lowe 2004 a ratio threshold of 0.8 will discard 90% of the FP matches and remove only 5% of the correct matches. Figure 5.13 is an example of nearest neighbor matching.

5.6.2 Sequential Structure from Motion

Having a set of matching points between two images makes it possible to calculate relative pose between the two images. Knowing the pose the matching points can be triangulated
Figure 5.12: SIFT algorithm applied to an image. Keypoints are located in the center of each circle. The circle is the local patch around every keypoint, and its size is determined by the keypoints canonical scale. Patch orientation is the line from the keypoint to the circle outline.

to 3D points in the scene. These properties are used in a process called sequential SfM.

The sequential SfM process is initiated with two images. This will give some common points that can be triangulated into the scene. Then images are added one by one. For every image added some points will be matched to existing points while other are added to the scene and extending the model. A high number of matches for every point will make it possible to remove errors by performing cross checks in epipolar geometry. In this way, it is possible to build a sparse point cloud one image at a time. (Opsahl 2016b)

The output from sequential structure from motion is a sparse point cloud without scale or position. Due to image noise there will be errors in the model.
5.6.3 Bundle Adjustment

The result from sequential structure from motion contains errors due to noise. As every point in the sparse point cloud is seen from multiple images noise will project the points to slightly different positions. These errors can be minimized in a process called bundle adjustment. It is a large optimization problem where the goal is to minimize the sum of squared reprojection errors. Removing as many unknowns as possible will make the result better. Extracting parameters from the image EXIF data is one way to get good initial parameters. (Szeliski 2011) Figure 5.14 demonstrates the concept of bundle adjustment.

Figure 5.14: Minimizing the sum of squared reprojection errors with bundle adjustment (Opsahl 2016b).
Bundle adjustment can easily become an optimization problem with an extreme number of parameters. This is computationally expensive and can take a lot of time. Several solutions to this problem have been proposed. One solution is to split the images into subsets and perform the bundle adjustments on several subsets. Another solution is to fix points and then cameras in two different calculations to reduce the number of parameters. (Szeliski 2011)

5.6.4 Dense Point Cloud Creation

Figure 5.15: Sparse point cloud of Barentsøya before surface reconstruction using multiple-view stereo.

The sparse point cloud from bundle adjustment can be enhanced using multiple-view stereo methods to create more points in the point cloud. Plane sweep is a common method where all images are warped into a common plane using homography. The chosen plane can be any plane, but it is most common to choose the ground plane or the image plane of one of the images. The plane is moved along an axis perpendicular to the plane at given intervals. For every position the correlation between corresponding pixels in the warped images is calculated using a small patch around the pixel. The position giving the highest score is chosen for every pixel. Using multiple warped images will give you scores in different frequencies. The combined score will help resolve ambiguities in difficult areas. (Szeliski 2011, Haavardsholm 2016) Figure 5.15 and 5.16 show a point cloud before and after surface reconstruction using multiple-view stereo.
5.6.5 Position and Scale

Structure from Motion processing can be done without any reference points in the real world. This will produce a 3D point cloud without scale or position. A model without scale or position will not be very interesting for anything else than visualization since it is hard to make measurements or compare it to other data. Reference points can be added to the model as the last processing step or it can be done at an earlier stage. In the MicMac processing workflow it is done before bundle adjustment.

Since the relative positions of all points in the model are known, reference points can be added to any part of the model. One popular approach is to collect precise GPS-coordinates for the camera. Another approach is to put reference objects with known coordinates in the scene (GCPs). It is important that the objects are represented in enough images and are easy to distinguish from the rest of the scene. Examples of objects are metal bolts (Kääb, L. Girod, and Berthling 2014) or 1x1m yellow targets (Westoby et al. 2012).
6 Software

This section will present the software used in this thesis. One important aspect of this thesis is to only use open source software. Most of the tasks done can probably be done with closed source commercial software, but that would limit the usability to those who have a license.

The software listed below is free to use as long as licenses and terms of use are respected. Development of these software packages is usually done by volunteers, sometimes funded by donations from satisfied users and companies. The main software packages used are therefore listed to give these developers the credit they deserve.

6.1 Python

The Python programming language is a high-level programming language. It is a versatile and easy to read programming language that is very well suited for general-purpose programming in all scales. Python has a large standard library that contains tools for many tasks and is often described as “batteries included”. In addition to the standard library the user community provides a huge collection of packages for almost any task imaginable. (Python Software Foundation 2017)

Python is the backbone of all programming or scripting done in this thesis. Using both standard functions and external packages it is used for everything from complicated tasks like image analysis to simpler tasks like generating text files. Python 2.7.6 was used for this thesis.

The main principles of Python are summarized in the Zen of Python\footnote{https://www.python.org/dev/peps/pep-0020/}. It is the informal entry number 20 in the Python Enhancement Proposals (PEP). (Zen of Python 2017) To view Zen of Python enter the following line in the Python interpreter:

```
import this
```

6.2 Jupyter Notebook

Jupyter Notebook is a document format that combines easy to read markdown with executable code. This combinations enables you to write nicely formatted and complex
equations along side with executable code in over 50 programming languages. (Thomas et al. 2016)

The notebook is a great format prototyping ideas as well as sharing them with others. Markdown documentation between the lines of code removes the need to provide explanations in other document formats. Jupyter Notebook was used to test concepts and ideas.

6.3 SciPy

SciPy is a Python-based open source scientific computing environment. (Jones, Oliphant, Peterson, et al. 2001) It contains numerous packages, but only the ones used in the thesis are mentioned in this section.

SciPy 0.13.3 was used for this thesis.

6.3.1 NumPy

NumPy is the fundamental package for scientific computing with Python. It enables you to work with N-dimensional arrays and linear algebra. (Walt, Colbert, and Varoquaux 2011)

NumPy 1.8.2 was used for this thesis.

6.3.2 Matplotlib

Matplotlib is a powerful plotting library for Python. It enables you to produce all kinds of plots both in 2D and 3D. (Hunter 2007)

All plots presented in the thesis are produced with Matplotlib version 1.3.1.

6.3.3 scikit-image

scikit-image is a image analysis library. (Walt, Schönberger, et al. 2014) The main reason for using this library is its functions for calculating texture features like entropy in images. scikit-image 0.12.3 was used.

6.4 OpenCV

OpenCV is an open source computer vision library with a very large collection of functions. It has interfaces for C++, C, Python and Java. OpenCV has very good documentation and a very large and supporting user community. (Bradski 2000)
Most of the image analysis done in this thesis is done with OpenCV. OpenCV version 3.1.0 was used.

6.5 dlib

dlib is a open source machine learning library. It is written for C++, but it has bindings to Python. (King 2009) The implementation of histogram of oriented gradients support vector machine from dlib 19.1.0 was used for this thesis.

6.6 MicMac

Micmac is a photogrammetry software developed at the IGN (French National Geographic Institute) and ENSG (French national school for geographic sciences) (Deseilligny, Jouin, et al. 2017). The software enables and demands the user to set every imaginable parameter, and some additional ones. This gives the user a lot of possibilities, but it demands some previous knowledge from the user. In addition to the command line documentation there is a wiki\(^2\) that provides good documentation and some tutorials to get you started.

6.7 QGIS

QGIS is a Free and Open Source Geographic Information System (GIS). It is an official project of the Open Source Geospatial Foundation. (QGIS Development Team 2017)

QGIS has all the functionality that can be expected of a GIS. It enables viewing, analyzing, and creating plots composed from many data sources.

In addition to the built-in functionality of QGIS the vast user community supplies plugins for almost every imaginable GIS-related task.

QGIS 2.18 Las Palmas was used in this thesis.

6.8 GDAL

Geospatial Data Abstraction Library (GDAL) is a translator library for raster and vector geospatial data formats. It is released with an Open Source license by the Open Source Geospatial Foundation. (GDAL Development Team 2016)

\(^2\)http://micmac.ensg.eu/
GDAL supports reading and writing most geospatial raster and vector formats. Its Python integration makes it very convenient when creating automatic processing workflows.

GDAL 2.1.0 was used in this thesis.

6.9 Docker

Docker is a software container platform system. It enables you to package a piece of software and all dependencies into a container which can be moved around and run on any system without additional setup. Docker container will work cross platform which makes it possible to run Linux software on Windows and Mac and removes “works on my machine” problems. (Docker Development Team 2017)

6.10 PDAL

Point Data Abstraction Library (PDAL) is a C++ library for translating and manipulating point cloud data. PDAL has readers and writers for most point cloud data formats and functions for filtering and processing the data. (PDAL Development Team 2017)

PDAL 1.5 was used in this thesis.

6.11 CloudCompare

CloudCompare is an open source 3D point cloud and mesh processing software. It provides a nice graphic user interface for inspecting point clouds. (CloudCompare Development Team 2017)

6.12 ImageMagick

ImageMagick is a free and open source software that can create, edit, compose, or convert bitmap images. It reads and writes over 200 formats. ImageMagick has a command-line interface, but is also available with a graphic user interface through various applications. (ImageMagick Development Team 2017)
Part III

Processing Svalbard 1936 Images
7 Processing Workflow

Figure 7.1: The input and output of the data processing. A series of high-oblique aerial photographs from 1936 are converted into a digital elevation model and a orthophoto. Illustrations from eastern Barentsøya, 1936.

This part of the thesis will focus on the processing steps involved in generating orthophotos and digital elevation models from a series of high-oblique aerial photographs of Svalbard from 1936. Figure 7.1 illustrates the input and output of the process. Between input and output there are a series of processing steps which try to automate the process as much as possible. The complete workflow is shown in figure 7.2. Every step involved will be explained in detail in chapter 8 through chapter 12.

The programming language Python is the backbone of the workflow. It does most of the work with support from a series of other software packages and libraries. The most important ones are presented in chapter 6. In the following chapters all data processing will be explained. When processing is done outside Python it will be specified explicitly. Processing done in Python with supporting libraries it will not always be specified. Image analysis is mostly done in OpenCV through Python. When no software package is specified one can assume that Python is used with or without external libraries.
Figure 7.2: Workflow for processing Svalbard 1936 images into orthophotos and digital elevation models.

The workflow is designed to work on all the images in the Svalbard 1936 and 1938 datasets. In this thesis it will only be applied to 99 images within the study area of Barentsøya. The only exception is the algorithm for locating fiducial marks which is applied to all available images for testing purposes. While the workflow is designed for a specific dataset it is likely that a slightly modified version can be applied to similar datasets.

A more thorough review of the workflow and how the open source software performed can be found in chapters 15 and 16. Before the workflow can be discussed in detail all the elements must be presented.
8 Find Fiducial Marks

In order to use scanned aerial photographs for digital photogrammetry the exact dimensions of the original image and how it relates to the scanned image must be known.

For aerial photos fiducial marks are often used as reference points in the analog images. Fiducial marks are unique markers placed along the image or film boundary. Usually there are four or eight fiducial markers placed on edges-centers and corners of the image. To make them easy to distinguish they usually comprise lines, circles and crosses in high contrast to a uniform, usually black, background. Fiducial marks should represent one specific point with high precision.

Measuring the exact distance in millimeters between fiducial markers on the analog film and the distance in pixels in the digital scanned image enables us to establish a relationship between analog film coordinate space and digital pixel coordinate space to solve inner orientation.

8.1 Svalbard 1936 Aerial Photographs

The aerial photographs from Svalbard in 1936 have four fiducial marks placed on the edge-centers of the film. Figure 8.1 shows the bottom edge fiducial mark. The marks have three teeth pointing towards the center of the image and a small circle in the middle which is indicates the reference point.

A quick visual examination of the geometric properties of the fiducial mark in figure 8.1 reveals some properties that might be helpful when searching for the exact position of the reference point:

- Lines along the edges of the teeth roughly point towards the reference point. They are not perfectly aligned.
- The reference point is a circle.
- The teeth of the fiducial marks are large features with well defined corners.

There are also properties making it difficult to work with these marks:

- Scratches and other damages to the film create a lot of noise on the fiducial marks.
• The background is not uniform. There are large differences in background between the fiducial marks of one image and even behind one single mark. Snow, sky, water and rock create numerous histogram peaks making thresholding difficult.

• The circles indicating the reference point of the fiducial mark are transparent and not a white spot. This means that the background image is what is seen through the small hole. The non-uniform nature of the background is a problem.

The algorithm for finding the reference point must be designed with these positive and negative characteristics in mind.

![Bottom fiducial mark from image 2922 in the 1936 aerial photography mission](image)

**Figure 8.1: Bottom fiducial mark from image 2922 in the 1936 aerial photography mission**

### 8.2 Detection

The first step in determining the exact reference point of fiducial marks is confirm the presence of fiducial marks and narrow down the search area. Narrowing down the search area is essential to get good results in a reasonable amount of time. The large size of the images means that processing will take a long time, but the amount of information is a larger concern. More information or features in the images increases the chance of false positive results. Reducing the search size will therefore reduce processing time and likely produce more accurate results.

#### 8.2.1 Position based detection

Initial narrowing is done based on knowledge about the positions of the fiducial marks. As mentioned in section 8.1 the four marks of every image is located on the edge-centers. Due to slight inaccuracies in the scanning process the position varies a bit, but the position is roughly the same.
Figure 8.2: Narrowing down fiducial mark search area. Kronebreen, 1936

Figure 8.2 illustrates the process of narrowing down the search area. The image centers are simply picked out by dividing image width and height by two (8.2a). This center value in pixels represents the image edge-centers, but not necessarily the edge centers of the original photo. To make the process robust against slight inaccuracies in the scanning process a search box is created around the edge-centers (8.2b). Testing has shown that search boxes that are 10% of the original image size is sufficient to make the process robust against scanning inaccuracies. The output image (8.2c) contains the fiducial mark while a lot of other features are removed. Processing time will also be heavily reduced. Reducing the size to 10% in horizontal and vertical directions results in a 99% reduction in the amount of pixels as illustrated in equation 8.1.

\[
\begin{align*}
\text{Original image} & : 13000px \times 13000px = 169000000px \\
\text{Reduced image} & : \frac{13000px}{10} \times \frac{13000px}{10} = 1690000px \\
\text{Ratio} & : \frac{1690000px}{169000000px} \times 100\% = 1\%
\end{align*}
\]  

8.2.2 Feature based detection

The position-based result from section 8.2.1 provides a fairly good assumption that the result contains a fiducial mark. It does not test for presence of fiducial marks or give any indication of where they are located. This section will discuss computer vision methods to accurately test the presence of fiducial marks and provide a more accurate position in the image.
Feature descriptors

When the goal is to math find the same object in two images, SIFT (Lowe 2004) is a popular method. Superficial testing on the fiducial marks did not give good results. Consisted matching was not achieved. There may be many reasons for this. One theory is that the fiducial marks are quite different because of noise. SIFT was discarded as detection method.

Template matching

Another approach to feature-based detection is to calculate correlation with a template feature in a window moving across the image. Before calculating correlation the template and the image are thresholded into a binary image representing the shape of the fiducial mark. Simple template matching relies on overlapping features to get good result. The method is very vulnerable to differences in scale and rotation. Noise, different contrast and other effects that will introduce artifacts during thresholding and filtering will also reduce the reliability of this method.

To make feature description more robust Ming-Kuei Hu 1962 introduced a set of seven moments to create descriptors that are invariant to scale, rotation, and reflection. Hu moments produce better results than simple thresholding, but the results vary with image properties. Figure 8.4a shows a sliding window correlation and its score. There is a visible correlation tendency, but the results are not uniform between images or between the marks of one image. One possible explanation to the low reliability is that the fiducial marks are part of the film border. The part of the fiducial mark that lies towards the image border is not very distinct and it makes out a large part of the total circumference. Results that are not uniform makes it nearly impossible to pick out the true positives.

Histogram of oriented gradients

The next effort in fiducial mark detection is to apply machine learning methods. Dalal and Triggs 2005 describes a sliding window histogram of oriented gradients (HOG) object detector. The object detector, implemented in dlib, was initially trained using 16 images from Brøggerhalvøya. To create a more uniform training dataset the image patches that are output from the position based detection described i section 8.2.1 were equalized. This process partially removes the effect of different lighting conditions. The support vector machine (SVM) is trained by feeding it images where the fiducial mark is annotated with

\[1543,1544,1545,1546,1547,1548,1550,1551,1552,1553,1554,1555,1556,1557,1558\]
a rectangle like shown in figure 8.3a. In an effort to make the SVM as precise as possible, equally sized rectangles were generated around the fiducial mark reference point. As seen in figure 8.3a the box is not centered around the fiducial mark. The mark is centered on the long edge, but on the short edge the mark is placed closer to the edge facing the image border at a distance of $\frac{1}{3}$ of the short edge length. The centerpoints were manually picked out for the 16 images and boxes were generated around them. To avoid rotation related problems individual object detectors were trained for top, left, bottom and right fiducial marks. The HOG detector for top fiducial marks is shown in figure 8.3b.

Creating an annotated training dataset was done manually the first time, but when the correction algorithm described in section 8.4 started producing accurate results the process was automated. This makes it easier to generate larger and more robust training datasets. To get a more robust detector 10 images\(^2\) from Kongsvegen were added to the training dataset.

![Annotated fiducial mark]![Trained HOG detector]

Figure 8.3: Training a sliding window histogram of oriented gradients object detector with images where fiducial marks are annotated.

The resulting HOG detectors can now be used to detect fiducial marks in other images. Input images to the detector must be equalized to get good results because of the equalization done in the training process. Running the HOG detector on test images produce very good results. Figure 8.4b shows how the detector performs on a test image and compares it to template matching using moments. The HOG SVM detector clearly outperforms the moment template matching method.

In addition to returning boolean result regarding the existence of a fiducial mark the HOG object detector will also return a rectangle indicating the position of the fiducial mark. This rectangle may be used for to further narrow the search area for the reference mark.

\(^2\)2914,2915,2916,2917,2918,2919,2920,2921,2922,2923
(a) Feature matching by using scale, rotation, and reflection invariant Hu moments. (Ming-Kuei Hu 1962)
(b) Feature matching by using sliding window HOG object detector. (Dalal and Triggs 2005)

Figure 8.4: Comparison of methods for detecting fiducial marks. The HOG object detector outperforms the moment-based template matching.

point of the fiducial mark, but this has not been implemented. Some of the prediction methods described in section 8.3 will use the results from the HOG detector to check for errors and produce good results on odd images.

8.3 Prediction

The detection described in section 8.2 is able to confirm the existence of a fiducial mark and does provide a rough position. In order to solve for inner orientation the exact position of the reference point is needed. This section will focus on finding the reference point in a robust way.

Prediction will mainly be based on the shape of the fiducial marks and their distinct features, but results from feature based detection described in section 8.2.2 will be used as well.

8.3.1 Thresholding

To use the shape of the fiducial mark for prediction it is essential to separate it from the rest of the image. This is done by thresholding the input image into a binary image with two values; fiducial mark or not fiducial mark. Before thresholding it is smart to analyze the histograms of the images that are to be thresholded. The histogram is a discrete function \( h(i) = n_i \) where \( n_i \) is the number of pixels with the intensity value
A normalized histogram can be created by dividing the histogram values by the total number of pixels in the image.

Figure 8.5 shows the four fiducial marks in one image and their normalized histograms. It is clear that the histograms are very different even within a single image, but one thing remains the same in all the images. The fiducial mark is represented as the left-most peak in the histogram. Isolating it from the rest of the image is slightly more difficult than it seems. Thresholding problems for the fiducial marks are mainly caused by the varying background around the marks. It is different between fiducial marks within one image and different for the same mark in different images. This is a problem when we are trying to find a good value $T$ that will separate the image into fiducial mark or not fiducial mark.

A visual inspection of the histograms indicates that a threshold value $T = 15$ will give a reasonably good threshold. Unfortunately a global value of $T = 15$ will not give very good results. Dark background (8.5c, 8.5d) will increase the size of fiducial marks while light background (8.5a, 8.5b) will reduce the size. The background varies too much to successfully introduce global threshold values for all fiducial marks in all images.

Automatic or semi-automatic calculation of the threshold value is one way of getting around the problem of unknown histograms. One approach to solve this problem is Otsu’s optimum threshold method. The method calculates between-class variance for all threshold values and chooses the smallest one. (Otsu 1979) Otsu’s method can be used...
to calculate any number of thresholds, but thresholding with more than two thresholds should be solved using other methods. Otsu’s method does not work on the fiducial mark images. The top and bottom images in figure 8.5 will give good results using Otsu’s method while applying it to the left and right images will produce threshold values that are too high. One could create multiple threshold values and pick out the smallest one, but this is not considered robust enough.

The solution is to develop a method for finding the peaks and valleys of the histogram and pick out the first valley after the first peak. The first step is to remove noise and extreme histogram values by applying or correlating the image with a Gaussian kernel filter. A large 25 × 25 pixel (\(\sigma = 4.1\)) filter is chosen because we are interested in the feature shape and not in small details. The difference between the histogram of a filtered and a unfiltered image is shown in figure 8.6a and 8.6b. To identify peaks and valleys of the histogram, gradient or the derivative \(h'(i)\) can be used as shown in figure 8.6c. One way of calculating histogram gradient is to define a symmetric linear filter \(f\) with negative values to the left and positive values to the right. The filter can have any length, but a longer filter like the one chosen here will have a smoothing effect which can be smart when working with discrete functions like a histogram.

\[
f = \begin{bmatrix} -3 & -2 & -1 & 0 & 1 & 2 & 3 \end{bmatrix}
\]  \hspace{1cm} (8.2)

The linear filter is then applied to or correlated with the histogram to calculate gradients. To make \(h'(i)\) and \(h(i)\) the same length \(h(i)\) is padded with zeros before correlation.

\[
h'(i) = f \ast h(i)
\]  \hspace{1cm} (8.3)

Valleys and peaks in the histogram are located where the gradient changes sign. Peaks are located at positive to negative sign changes while valleys are located at negative to positive sign changes. Following this logic valleys and peaks in the histogram can easily be picked out with an algorithm that iterates through the values of \(h'(i)\) and compares signs of two consecutive values.

Now we know the position of all the valleys and peaks of the histogram, but we do not know which peak we are trying to find. Simply picking the first peak is not good enough. There may be a small peak before the main peak from the fiducial mark. Searching for the global maximum will not work either. As shown in figure 8.5d the first peak of the histogram is not always the global maximum of the histogram. The chosen solution is to say that the peak created by the fiducial mark will be the local maximum between 0
and a initial threshold $T_i$.

$$i_{\text{fiducial peak}} = h(i)_{\text{max}} \quad 0 \geq i \geq T_i$$  \hspace{1cm} (8.4)

Otsu’s method (Otsu 1979) is used to find the initial threshold value $T_i$. The intensity value of the peak created by the fiducial mark can now be calculated using equation 8.4. As a safety valve a maximum expected peak value $p_{max}$ is inserted. The intensity value of the peak is recalculated if it is larger than $p_{max}$. By default $p_{max} = 20$.

$$\text{if } i_{\text{fiducial peak}} > p_{max} \Rightarrow i_{\text{fiducial peak}} = h(i)_{\text{max}} \quad 0 \geq i \geq p_{max}$$  \hspace{1cm} (8.5)

Using time and resources to calculate $T_i$ may seem wasted, but it has proven very useful because of the large differences between the images. In many cases multiple peaks are present below $p_{max}$.

When the histogram peak of the fiducial mark is known it is a simple task to find the first valley after the peak. Set initial intensity to intensity value of fiducial peak $i = i_{\text{fiducial peak}}$ and iterate through all intensity values until $h'(i-1) < 0$ and $h'(i) > 0$. Intensity values where $h'(i) = 0$ are skipped. To ensure that the chosen $i$ is a real valley its value must be smaller than half the local maximum between 0 and a initial threshold $T_i$.

$$h(i) < \frac{h(i_{\text{fiducial peak}})}{2}$$  \hspace{1cm} (8.6)

Final threshold value is now $T = i$. If the final value is larger than the initial value obtained with Otsu’s method the initial value is used instead.

Using equation 8.7 the image is divided into background pixels with the value 0, and pixels with value 1 for the fiducial mark.
\[ g(i, j) = \begin{cases} 
0 & \text{if } f(i, j) \geq T \\
1 & \text{if } f(i, j) < T 
\end{cases} \]  

(8.7)

Before the thresholded image is passed to the corner detector some last noise removing steps are applied. First all features are filtered by size. This is done by measuring the area of groups of foreground pixels (features) and removing any feature with an area less than 100 pixels. Then the image is inverted and the process is repeated. The process will remove noise particles both on the fiducial mark and on the background. As a final step morphological closing with a 5 * 5 pixel filter is applied to the image. This will remove small imperfections along the edge of the fiducial mark. Figure 8.7 shows the final result of the thresholding process.

![Thresholded fiducial mark](image)

Figure 8.7: Thresholded fiducial mark

### 8.3.2 Equalizing before threshold

Equalization is not included in the general normal thresholding procedure because it will alter the histogram and may make some of the assumptions about the histogram false. For most of the images equalization will complicate the process and give worse results. This section will focus on the images that require equalization as a preprocessing step before thresholding.

#### Very dark images

Some images are very dark and this makes it nearly impossible to find a good threshold value. Stretching the histogram with equalization and then blurring the image with a 25 * 25 pixel (\( \sigma = 4.1 \)) Gaussian filter will give a histogram that is better suited for thresholding. Figure 8.8 shows how the process transforms the histogram. One key feature is that the number of pixels with intensity value zero \( i = 0 \) goes from \( h(0) \approx 0.55 \)
to $h(0) \approx 0.30$. The original image had a false over representation of pixels with intensity value zero caused by noise. The process effectively removes a lot of this noise. A simple blurring without equalization will also lower the amount of zero pixels, but it will not be as effective. Equalization is only done on very dark images that have a mean value smaller than 10.

Figure 8.8: Equalize and then blur very dark image to spread the distribution of intensity values.

After equalization the normal thresholding routine described in section 5.1 cannot be used. A new algorithm finding the end of the maximum peak is used:

1. Find maximum $h_{\text{max}}$ and minimum $h_{\text{min}}$ values of histogram
2. Calculate histogram range $h_{\text{range}} = h_{\text{max}} - h_{\text{min}}$
3. Find intensity value of histogram maximum $i_{h_{\text{max}}}$
4. Use $i_{h_{\text{max}}}$ as a starting point $i = i_{h_{\text{max}}}$ and calculate the difference between the current and next histogram value.
   \[ \Delta h = |h[i] - h[i + 1]| \]  
   (8.8)
5. Increase intensity value by one
6. Repeat step 4 and 5 until $\Delta h < h_{\text{range}} \times 0.05$. The histogram is now considered flat and threshold value $T = i$ is used. The equation 8.7 is used for thresholding.

**Histograms with one peak**

Images with only one histogram peak, like the one in figure 8.9a, are problematic when using the thresholding described in section 5.1. It is not possible to find valleys and
Otsu’s method will generally threshold with too high intensity values. The solution to the problem is to equalize the image and then blur the image. Figure 8.9 shows the histogram of an image that is first equalized and then blurred with a $25 \times 25$ pixel ($\sigma = 4.1$) Gaussian filter. The resulting histogram has clear valleys and peaks and is ready for thresholding using the method from section 5.1, but without the maximum expected peak value $p_{\text{max}}$. This is because it is very hard to estimate a value for $p_{\text{max}}$ after equalization. Other than that the thresholding method is identical.

Figure 8.9: Equalize and then blur images with only one histogram peak to remove noise and introduce more peaks and valleys.

### 8.3.3 Corner detection

The next step in predicting the position of the reference point of the fiducial mark is to find some stable and robust features in the image. Using the actual reference point (the hole) will not work as it often will be treated as noise and therefore removed during the thresholding process. Looking at the thresholded result in figure 8.7 shows that the most distinct features are the jaws of the fiducial mark.

In order to find corners a algorithm known as the Harris corner detector (Harris and Stephens 1988) can be used. Applying the Harris corner detector directly on the thresholded image will not produce good results because the method will produce positive results on every little imperfection when it is applied to a binary image. To better the result the somewhat rough output from the thresholding is be smoothed by applying $5 \times 5$ pixel ($\sigma = 1.1$) Gaussian filter. When the Harris corner detector is applied to the smoothed image it will only produce positive results on the large features of the image.

Figure 8.10 shows the output of the Harris corner detector. The numbering and distribution of points between the images will be random. The only thing that is constant is that a point is situated on the distinct features on the jaws of the fiducial marks. In
Figure 8.10: Fiducial mark corners found with Harris corner detector

Figure 8.10 the distinct points are annotated as 13, 14, 15, 16 and 17. The reference point is annotated as number 8 in the figure, but it is not detected often enough to use it as a distinct point.

8.3.4 Matching distinct points

When all the corners in the image have been located they can be used to estimate the reference point of the fiducial mark. To do this the correct points must be separated from all the other points. The algorithm is trained on a constellation of points and uses this to search for other constellations of points. Inspiration comes from the random sampling consensus (RANSAC) (Fischler and Bolles 1981) algorithm, but it uses a more brute force approach and searches through all combinations of points instead of applying a stop criteria. This section will explain the algorithm for matching distinct points and calculating a estimate for the fiducial mark reference point.

To train the shape matcher, an image with annotated distinct points must be created. In figure 8.11 the distinct points from figure 8.10 have been picked out and given a name. The points are fed to the training algorithm which calculates point positions with a series of angles and distance scales with $L_0R_0$ and $R_0L_0$ as baselines. Table 8.1 shows all the values that are stored during training.

The trained values can now be used to search for the two points, $L_0$ and $R_0$, that gives the best fit to the trained model. As mentioned earlier every combination is tested. This brute force approach is more resource intensive, but it is difficult to find a good criteria to break the search. The number of points that have to be tested is of great importance for the resources needed to complete the algorithm as the number of combinations $n$ is
related to the number of points \( n_p \) in the following way:

\[
n = n_p^2 \quad (8.9)
\]

The number of points generated by the corner detector (8.3.3) is usually around 20. For noisy and difficult images the number of points can be as high as 100. As long as the number of points are this small the brute force approach will be fast enough.

The algorithm for finding the best \( L_0 \) and \( R_0 \) in the set of points \( P = \{p_0, p_1, p_2, ..., p_n\} \) iterates through all combinations of \( \{p_i, p_j\} \in P \) and performs the following tests:
1. Calculate the baseline distance \( d_b \) between \( p_i \) and \( p_j \):

\[
d_b = | \overrightarrow{p_i p_j} | \tag{8.10}
\]

2. Test if the baseline distance \( d_b \) lies between two given extreme values \( d_{\text{min}} \leq d_b \leq d_{\text{max}} \). If it is not the combination is discarded and the algorithm is restarted from step 1 with the next combination of \( p_i \) and \( p_j \).

3. Calculate the points \( L_1, L_2, R_1 \) and \( R_2 \) using the trained values for angle \( \theta \) and scale \( s \) shown in table 8.1. The rotation matrix \( R \):

\[
R = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta)
\end{bmatrix} \tag{8.11}
\]

The scaling matrix \( S \):

\[
S = \begin{bmatrix}
s & 0 \\
0 & s
\end{bmatrix} \tag{8.12}
\]

Calculate position of \( R_1, R_2, L_1 \) and \( L_2 \):

\[
P_{\text{calc}}[R_1, R_2, L_1, L_2] = \begin{bmatrix} p_{ix} \\ p_{iy} \end{bmatrix} + \begin{bmatrix} p_{jx} - p_{ix} \\ p_{jy} - p_{iy} \end{bmatrix} RS \tag{8.13}
\]

4. Calculate the distance from the calculated points \( P_{\text{calc}}[R_1, R_2, L_1, L_2] \) to the closest point \( p \in P \). This distance is the error. How far the point is offset from its expected position. Store the sum of errors for this combination of \( p_i \) and \( p_j \).

5. Repeat step 1 to 4 for all combinations of \( \{p_i, p_j\} \in P \)

6. Find the combination of \( \{p_i, p_j\} \in P \) that has the lowest sum of errors and use these points as \( L_0 \) and \( R_0 \).

7. Use \( L_0, R_0 \), the stored training values from table 8.1 and the equations 8.11-8.13 to calculate \( R_P L \) and \( R_P R \).

8. Estimate reference point \( R_P \) based on the mean value of \( R_P L \) and \( R_P R \).

\[
R_P = \frac{R_P L_x + R_P R_x}{2}, \frac{R_P L_y + R_P R_y}{2} \tag{8.14}
\]
8.3.5 Error check

To confirm the validity of the points output from section 8.3.4 error checks must be performed. Based on results from previous processes the quality of the result can be judged to some extent. The following tests are performed on every $RP$:

1. **Is $RP$ a point?** If something goes wrong in the point matching process the output may be nothing. This is often an indication of bad thresholding. If output is *none* the image is equalized and blurred before it is thresholded again.

2. **Is $RP$ inside the bounding box returned from the HOG SVM detector?** If the HOG SVM detector returned a result the position of $RP$ can tested against the rectangle. If $RP$ is outside the rectangle the image is equalized and blurred before it is thresholded again.

3. **What is the mean intensity value of the area around $RP$?** The fiducial mark is always dark compared to the rest of the image. The mean value of a $30 \times 30$ pixel patch around $RP$ should always be lower than the mean value of the position based detection area returned by the process described in section 8.2.1. If it is higher the image is equalized and blurred before it is thresholded again.

4. **If equalization has not produced a valid result the last attempt is to calculate $RP$ from the output polygon of the HOG SVM detector.** In order to do this the HOG SVM detector must have returned a result. A coarse approximation of the $RP$ can be obtained from the polygon by applying the parameters, used for creating training annotations, in reverse (section 8.2.2). While the HOG SVM detector is very efficient in detecting fiducial marks it is not accurate enough to precisely pick out the $RP$. The approximation will often be offset with 20-40 pixels.

All tests will be performed but operations like equalization will only be done once even if multiple tests are failed. Failed tests or additional operations that have been performed will be output as information messages. If $RP$ fails all the tests a error message will be output.

Error check is the last step in the prediction workflow. The accuracy of the prediction will highly likely lie within 10 pixels if $RP$ was found using the normal method, with or without equalization. If the HOG SVM detector was used the offset will often lie between 20-40 pixels. Larger offsets will increase the probability of error when correcting the position. This will be discussed in section 8.4.
8.3.6 Failed attempts

Numerous methods for prediction have been tested, but did not give good enough results. They are mentioned in this section.

Cone filter

When searching for closed circular shapes cone filters can be applied. The cone has both positive and negative pixel values. Maximum pixel value is located at the filter center and decreases towards the edges. The most important feature of the filter is that pixels with value zero are placed in a ring with the same diameter as the fiducial mark reference point \( (RP) \). When the filter reaches the \( RP \), the dark fiducial mark will be hit by the negative values of the filter while the light pixels in the \( RP \) will hit the positive values in the filter center. In the filtered image the \( RP \) will be represented with the maximum value. Figure 8.12 illustrates the process of filtering an image with a cone filter.

![Cone filter process](image)

Figure 8.12: Locating fiducial mark reference point \( (RP) \) with cone filter. The maximum response is located in the center of the fiducial mark reference point.

Unfortunately the cone filter is sensitive to noise and other abnormalities in the image. It will give accurate results in many images, but the error rate is too high and is therefore discarded.

Hough transform

Hough transform (VC 1962) was tested. Using intersections of Hough lines generated from the teeth of the fiducial mark worked in some images, but not robust enough. Using Hough circles to detect the circular reference point was successful only in a few cases.
Prediction using regression trees and landmarks

Promising results using machine learning methods for detection of the fiducial mark inspired an attempt to use it for prediction of the fiducial mark reference point. The first attempt was to simply use the centerpoint of the bounding box returned by the HOG object detector. While the HOG object detector will detect fiducial marks in the image its accuracy is not good enough to accurately predict the reference point. As shown in figure 8.13a, the centerpoint of the HOG detector bounding box will often miss the reference point with 20-40 pixels. Offsets this high will increase the chance of errors in the correction algorithm described in section 8.4, but in some cases this is the only option.

As an experiment a regression tree algorithm for detecting facial landmarks (Kazemi and Sullivan 2014) was trained with 16 annotated images\textsuperscript{3} from Brøggerhalvøya. Some easily detectable features like the reference point and jaws were chosen as shown in figure 8.13b. Testing the predictor did not produce any useful results. Possible explanations of the failure is the limited number of training images and the authors limited ability to adapt the model to the purpose.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure8.13.png}
\caption{Prediction using (a) center of HOG detector bounding box and (b) Annotations for training predictor described in Kazemi and Sullivan 2014 bounding box}
\end{figure}

Figure 8.13: Attempts to use machine learning like regression trees and landmarks for precise prediction were not successful

8.4 Correction

The result from the prediction procedure described in section 8.3 does not produce a result that is accurate enough to solve for inner orientation. However a result with an

\textsuperscript{3}1543,1544,1545,1546,1547,1548,1549,1550,1551,1552,1553,1554,1555,1556,1557,1558
estimated maximum error of about 10 pixels is a very good starting point when trying to determine the exact location of the reference point. The estimate returned by the HOG SVM detector with expected accuracy of 20–40 pixels will also work in most cases, but some slight adjustments have to be done.

Knowing the approximate location makes it possible to reduce search extent and make more accurate assumptions about the image.

- The image will highly likely only contain the dark fiducial mark, noise and the reference point. No terrain will be visible except what is visible through the reference point in the fiducial mark.
- The histogram will mostly contain low pixel values due to the relatively uniform dark background while noise and the reference point will be represented by a few high pixel values.
- Noise will often have higher pixel values than the reference point.
- The pixel area ratio between noise and the reference point is unknown.
- The reference point is a large circle while noise often occur as lines or more elliptical shapes.
- The maximum radius of the reference point is known. With the resolution that most of the images have it will always be smaller than 10 pixels.
- Individual noise features are usually smaller than the reference point. Features smaller than 3 pixels in radius is always considered noise.

The validity of these assumptions is essential for creating a good algorithm. They will stay valid if the algorithm is run completely within the fiducial mark.

The algorithm to correct the reference position of the fiducial mark must take advantage of the accurate assumptions. The following steps are performed to accurately locate the fiducial mark reference point.

1. In order to maintain validity of the assumptions the correction algorithm is run within a relatively small area. The calculation area is calculated from the input image size. If the input reference point is produced with the normal thresholding and corner detection procedure the area will be approximately $40 \times 40$ pixels. If the input comes from the HOG SVM detector a much larger area of about $100 \times 100$ pixels must be used. The area is extracted from the original image with the input
reference point as center (figures 8.14a, 8.14b). The assumption made earlier will be true within the area extracted from the images. All the calculations to come are done within this area.

2. Noise is removed by blurring the image with a 5×5 pixel (\(\sigma = 1.1\)) Gaussian kernel.

3. The image is thresholded to separate the relatively uniform fiducial mark from its reference point. Looking at the image it looks bimodal, but applying bimodal thresholding (Otsu 1979) will remove the fiducial mark reference point in many images because of the high pixel values of noise. A better approach is to identify the extent of dark fiducial mark pixels. The pixel value range of the fiducial mark is unknown because of errors introduced by the scanner, but because it is heavily overrepresented in the image it is approximately normal distributed around the image mean. Thresholding the image with a threshold set to the image mean (equation 8.15) will separate noise and the hole from the background as shown in figure 8.14c.

4. After thresholding noise and the fiducial mark reference point are in the same class. In order to separate them their shape properties must be used. Calculating euclidean distance to the closest background pixel for every foreground pixel will give the highest scores in large circular objects. Large noise features like the one in figure 8.14d will cause trouble. To avoid wrong corrections caused by large noise particles all features with maximum distance larger than 10 pixels are removed as long as there are remaining features with distances larger than 3 pixels. The resulting distance matrix after filtering is shown in figure 8.14e.

5. The maximum value in the distance matrix is chosen as the new fiducial mark reference point (figure 8.14f).

6. If the maximum distance is smaller than 3 pixels the algorithm will not perform the correction. It will return the input reference point and a warning about dark image.

Figure 8.14 illustrates all the major steps of the process.
Sections 8.3-8.4 have explained how fiducial marks have been detected and their reference point calculated. Figure 8.15 is a graphical representation of the workflow for detecting and accurately defining the position of fiducial marks. This section will focus on the output and how well the workflow performs.

### 8.5.1 Process output

As mentioned in the very beginning of chapter 8 the exact positions of the fiducial marks must be known both in the analog film and the digital image. This enables solving the inner orientation of the digital image. Knowing the inner orientation enables us to apply digital photogrammetry methods to the images.
MicMac is used for the image correction. In order to do the job the software requires the parameters to be supplied as two xml-files with a given set of tags. These files are easily output with Python based on the results from the fiducial mark workflow.

The first file contains the positions of the fiducial marks on the analog film. These values have been measured on the original film in millimeters. This file is only generated once because these measurements apply to all the images.

The second file is generated for every image. It contains the position of the fiducial marks in the digital images measured in pixels. These are the values that are found with the fiducial mark workflow.

As an optional choice a graphic output can be generated for every image and placed in a html file. Statistics and a index with links to all the images is also created. This enables easy viewing in web browsers.

Figure 8.16 shows the graphic output generated when image S36_0830 was processed. Some general information about the image is presented as text in the first few lines. Below the text there are four small images. They are small patches of the input image covering
the area around the detected fiducial mark reference points. The yellow crosshair in the images show the calculated position of the reference point. Two enlarged images, one normal and one equalized, are shown on the left to better show the exact position. Other pieces of information like coordinates, the image and fiducial mark names are also printed on the images. Processing information is printed in blue. The \textit{S36_0830 [BOTTOM]} image in figure 8.16 has been equalized before threshold.

Figure 8.16: Graphic output from fiducial mark finder. The informative images makes it possible to check results very quickly.

The graphic output layout seen in figure 8.16 has been carefully chosen to enable quick manual inspection of the results. When all the images are lined up below each other in a web browser one can very quickly scroll through the results and pick out errors. The enlarged and equalized images enable inspection of results that are barely visible in the original image at normal resolution. Using this method the processing result of 857 images were manually inspected in approximately 30 minutes.

\subsection{8.5.2 Performance}

Performance can be measured in many ways. The measures used here are speed and accuracy.

When measuring processing time for the fiducial mark finder only the image processing is measured. This includes loading the image from disk and locating the four fiducial marks in the image. Processing one image takes between 3 and 4 four seconds on a old laptop. Images that require additional processing with HOG SVM or equalization will usually increase the processing time with about one second. Generating MicMac xml files is very fast and will not influence the total processing time, but the optional process of generating graphic output is time consuming and will take an additional 4 to 5 seconds.
Figure 8.17 shows a typical distribution of processing time when the location fiducial marks, generating MicMac xml files and producing graphic output.

![Distribution of processing time](image.png)

Figure 8.17: Distribution of processing time

The accuracy of the fiducial mark finder has been tested both with automatic and manual tests (table 8.2).

<table>
<thead>
<tr>
<th>General information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of images</td>
<td>857</td>
</tr>
<tr>
<td>Number of fiducial marks</td>
<td>3428</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Automatic processing feedback</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Image equalized before threshold</td>
<td>66 fiducial marks</td>
</tr>
<tr>
<td>Detected with SVM</td>
<td>7 fiducial marks</td>
</tr>
<tr>
<td>Unable to correct result, image is too dark</td>
<td>8 fiducial marks</td>
</tr>
<tr>
<td>Failed to calculate fiducial mark</td>
<td>1 fiducial mark</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Findings from manual inspection</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Correction slightly offset</td>
<td>1 fiducial mark</td>
</tr>
<tr>
<td>Wrong result</td>
<td>5 fiducial marks</td>
</tr>
</tbody>
</table>

Based on manual tests it is clear that the fiducial mark finder performs very well. Only 6 of 3428 fiducial marks have been detected with significant errors. In addition to this 8 fiducial marks were too dark to perform correction. However visual inspection of the result showed that they were in the right area based on the shape of the fiducial mark and it is likely not possible to get a better result using manual methods.
The major concern with the method is that it is not able to detect false positives. 5 of the 6 significant errors were not detected by the algorithm. One possible explanation for this is that many errors are detected and corrected in the algorithm. All the cases where equalization and HOG SVM is applied are errors that are detected and corrected. If these numbers are included the fiducial mark finder will have better error detection statistics. With 6 wrong results of a total of 3428 fiducial marks the error rate is 0.18%. In other words it accurately locates the fiducial mark reference point with a success rate of 99.82%.
9 Sparse Point Cloud Creation

This chapter will focus on creating a sparse point cloud from the images using structure from motion. The processing steps required are keypoint extraction, keypoint matching, georeferencing and bundle adjustment.

The first step of point cloud creation is to determine which datasets to include. Of the 857 images in the dataset 99 images were chosen to create a point cloud of Barentsøya (figure 9.1). There are more images in the dataset viewing Barentsøya, but they were removed because they were unsuited for point cloud creation. A collection of images in the northern part of the east line were discarded due to dense cloud cover and two lines going from north to south and south to north across Barentsøya were discarded because of lack of good keypoints on the glacier.

The scanned images do not contain EXIF data that enables us to calculate the relation between the image plane and focal length known as inner orientation. Before the images can be used in digital photogrammetry the inner orientation of the images must be calculated. This is done by applying properties from the analog film to the digital image. Focal length in millimeters (table 4.1), fiducial mark positions in millimeters (table 4.2) and pixel size given by image dimensions divided by number of pixels (0.0135mm) and fiducial mark positions from section 8.5 are used as input. The MicMax function ReSampFid does the correction and write the required metadata to the images. The geometry of the image plane and focal length is now known and can be projected into the scene.

With inner orientation known the 99 images are ready for processing. The next step is to extract keypoints in all the images and match them against each other to identify the same place in multiple images. Unfortunately the geometry between the images is not good enough to allow them to be processed together. In order to ensure reliable recognition between images in 3D scenes the rotation between images should not exceed 30 degrees (Lowe 2004). To get around this limitation the images are split into four lines, east, west, north and south, and processed independently. Figure 9.1 shows all the camera positions and how they are separated into four lines to avoid camera rotation exceeding 30 degrees.

To reduce processing time and the false-positive rate only keypoints of images that lie close to each other are matched. The neighboring images are estimated using the rough
camera positions provided by the Norwegian Polar Institute shown in figure 9.1.

Processing is done with MicMac. *Tapioca* extracts keypoints and *Tapas* matches them between images with the *FraserBasic* camera calibration algorithm (Fraser 1997). To avoid matching the four fiducial marks and the information boxes in the top-left and top-right corners a global mask is applied to exclude them. *AperiCloud* gives us a point cloud with relative coordinates. For it to be useful for anything else than visualization it must be georeferenced.

![Barentsøya Camera Positions](image)

Figure 9.1: Barentsøya camera positions divided into lines. Background map: Norwegian Polar Institute n.d.(a)

### 9.1 Georeferencing and Bundle Adjustment

To transform the relative point cloud into real world coordinates it must be pinned down with ground control points (GCPs) since the camera coordinates are inaccurate. GCPs for Barentsøya were picked using data with quite low resolution. The main source in x,y-direction is a WMTS service from The Norwegian Polar Institute containing Landsat-8
images with a ground resolution of 15 meters and maximum error of 65 meters with 90% confidence. (Norwegian Polar Institute n.d.(b), Landsat 8 (L8) Data Users Handbook - version 2.0 2016). The positions were checked against Svalbard S100 (Norwegian Polar Institute 2016a). Vertical coordinates (z) are picked from the Svalbard S0 terrain model with ground resolution 20 meters and a expected standard deviation of 25 meters in the vertical direction (Norwegian Polar Institute 2016b). GCP collection is done in the coordinate system ETRS89 utm zone 33.

With such low resolution and inaccurate data as input the GCPs will be inaccurate. Figure 9.2 shows the positions of 31 GCPs created on Barentsøya. They are created on distinct terrain features like peaks, small islands, sharp edges in topography like crevices, valley intersections and river intersections. Finding good GCPs on low resolution imagery is challenging and many of the chosen points are not optimal. Sharp edges in topography like peaks, crevices and valleys are places where the gradient of the terrain is likely to change rapidly which makes it likely that the largest errors in the digital terrain model are located in these areas. Erosion adds additional uncertainty to the GCPs. Rivers may have changed their course over the last 80 years.

One other alternative is to place GCPs along the shoreline where elevation is quite certain. The amount of change in sea level is relatively small compared to other errors. Some points were placed relatively close to the shore, but not on the shoreline because water water is very hard to match both with SIFT and in the dense correlation. Poor matching will lead to additional errors. In the end it boils down to choosing between poor GCPs or no GCPs. From the information provided with the data the GCPs must be expected to have errors with a standard deviation of at least 25 meters in all directions.

Using the MicMac tools SaisieAppuisInit, GCPBascule and SaisieAppuisPredic the GCPs are given pixel coordinates in the image plane. The manual process of connecting GCPs to the image is shown in figure 9.3.

As a last step in the georeferencing a bundle adjustment is performed with the tool Campari in MicMac. Bundle adjustment is a process which minimizes the sum of squared reprojection errors by adjusting the camera orientation and position. As input to the process the expected uncertainty of the user input GCPs is given both in ground coordinates and in image coordinates. The expected uncertainty of the GCPs in ground coordinates is set to 25 meters (figure 9.2) while the uncertainty in image coordinates is set to 5 pixels (figure 9.3). The georeferenced sparse point cloud created with AperiCloud after bundle adjustment is shown in figure 9.4.

One other output from bundle adjustment are the residuals. They are the remaining errors in GCPs after errors have been minimized. For every GCP the distance to
the corresponding point in the point cloud is calculated. Low values indicate that the model created with SfM fits the supplied GCPs well. High values mean that there are differences. Bundle adjustment will try to fit the model to the GCPs, but if the GCPs are poor there will be errors. The strong relationships created with photogrammetry will not be broken to fit GCPs, but it will be scaled and rotated to a best fit.

Table 9.1 summarizes the residuals for all the four flight lines. From the table it is clear the errors are large. They are much larger than the 25 meter expected error. This is partly caused by errors in the source dataset. Errors from the source data are further increased by the high-oblique angle of the images. These errors, introduced by poor GCPs, will stay in the data for the rest of the process. Precision and accuracy will be discussed more in chapter 14.
Figure 9.3: Barentsøya GCP’s manually placed in images with MicMac. Freemanbreen seen from south.

Figure 9.4: Sparse point cloud of Barentsøya seen from south towards north.
Table 9.1: Residuals after bundle adjustment for the individual flight lines. They are in meters in $x$, $y$, $z$ directions and total distance $d$.

<table>
<thead>
<tr>
<th></th>
<th>max</th>
<th>min</th>
<th>mean</th>
<th>std</th>
<th></th>
<th>max</th>
<th>min</th>
<th>mean</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>north</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>north</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>51.77</td>
<td>-138.71</td>
<td>-1.63</td>
<td>53.05</td>
<td>x</td>
<td>179.56</td>
<td>-79.02</td>
<td>-2.67</td>
<td>68.22</td>
</tr>
<tr>
<td>y</td>
<td>83.43</td>
<td>-97.56</td>
<td>-6.00</td>
<td>57.41</td>
<td>y</td>
<td>68.00</td>
<td>-27.92</td>
<td>-1.62</td>
<td>25.96</td>
</tr>
<tr>
<td>z</td>
<td>30.98</td>
<td>-30.59</td>
<td>0.79</td>
<td>15.82</td>
<td>z</td>
<td>52.78</td>
<td>-20.87</td>
<td>0.43</td>
<td>22.58</td>
</tr>
<tr>
<td>d</td>
<td>163.13</td>
<td>14.89</td>
<td>70.91</td>
<td>37.03</td>
<td>d</td>
<td>199.13</td>
<td>5.23</td>
<td>52.06</td>
<td>56.01</td>
</tr>
<tr>
<td>west</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>south</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>26.45</td>
<td>-28.13</td>
<td>-0.12</td>
<td>17.34</td>
<td>x</td>
<td>63.55</td>
<td>-36.94</td>
<td>0.18</td>
<td>29.07</td>
</tr>
<tr>
<td>y</td>
<td>46.46</td>
<td>-25.72</td>
<td>-0.30</td>
<td>18.87</td>
<td>y</td>
<td>116.24</td>
<td>-158.48</td>
<td>-2.64</td>
<td>65.12</td>
</tr>
<tr>
<td>z</td>
<td>27.35</td>
<td>-23.70</td>
<td>-0.00</td>
<td>16.97</td>
<td>z</td>
<td>44.44</td>
<td>-37.54</td>
<td>0.35</td>
<td>19.61</td>
</tr>
<tr>
<td>d</td>
<td>47.79</td>
<td>16.54</td>
<td>28.96</td>
<td>10.30</td>
<td>d</td>
<td>170.86</td>
<td>11.53</td>
<td>59.73</td>
<td>43.70</td>
</tr>
</tbody>
</table>
9.2 Adjusted Camera Positions

After bundle adjustment camera position and orientation is corrected. MicMac saves all image parameters as xml-files that can be parsed with Python to extract information. Given that the georeferencing is correct the exact position and orientation are returned as a point $P$ and a rotation matrix $R$.

\[ P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{9.1} \]

Using this information it is possible to display image attributes on maps. The camera position is obvious, but it is also possible to make coverage boxes for all the images. Another and possibly more intuitive way of referring to orientation is to use yaw, pitch and roll. Accurate camera positions are valuable in the point cloud processing workflow in chapter 12, because it enables filtering based on distance from camera.

To check the quality of the approximate positions provided by the Norwegian Polar Institute they are compared to the camera positions calculated from SfM. When comparing the two datasets they must be in the same coordinate system. The positions provided by The Norwegian Polar Institute are in WGS84 utm zone 33 while the corrected camera positions returned by MicMac are in ETRS89 utm zone 33. Using GDAL the approximate positions are transformed into ETRS89 utm zone 33.

The approximate dataset contains information about $x$, $y$, $z$ and $yaw$ while MicMac returned $x$, $y$, $z$ and $R$. Using simple calculations on $R$ enables us to extract $yaw$, pitch and roll for the corrected positions. For comparison only $x$, $y$ and $z$ are used. Offset in all directions ($\Delta x, \Delta y, \Delta z$) and the total offset $d$ are calculated for all points.

\[ \Delta x = |x_0 - x_1|, \quad \Delta y = |y_0 - y_1|, \quad \Delta z = |z_0 - z_1|, \quad d = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{9.2} \]

The results are summarized in figure 9.5. Looking at the boxplot (figure 9.5b) it is clear that most of the errors are in the $x$ and $y$ directions while $z$ errors are quite small in comparison. The map (figure 9.5a) shows that most of the approximate points are set too close. The images are generally taken from further away than the dataset indicated. Errors are larger than expected.
(a) Arrows point from the estimated position to the actual position. Background map: Norwegian Polar Institute n.d.(a)

Figure 9.5: Difference between approximate camera positions from Norsk Polarinstitutt and the more accurate camera positions extracted after bundle adjustment. The vertical difference is very small compared to horizontal difference.
10 Masking Terrain Features

In the 1936 images from Svalbard there is a lot of information that is not relevant. Areas containing clouds, horizon and water is not wanted in the end result and they have properties that make matching and point cloud generation difficult and full of errors. Removing them will save a lot of processing time in the dense correlation process and ensure a better end result.

In previous workflows this has been a manual task (Couderette 2016) which is very time consuming and boring. For a small set of images the manual approach may be an alternative, but for large datasets an automated workflow is far better.

This chapter will present an automated workflow for masking clouds, water and horizon based on the texture and object-background segmentation methods presented in chapter 5. At the end the quality of the result will be discussed.

10.1 Pre-processing

Before masking terrain features some preprocessing steps steps must be applied. Internal orientation is already calculated and the images are corrected based on the fiducial marks found in chapter 8, as explained in chapter 9.

The input images have a size of about $13000 \times 13000$ pixels and contain very much information. For terrain masking this level of detail is not needed and it will require a lot of processing time. Testing has shown that images downsampled to $\frac{1}{4}$ of the original size in $x$ and $y$ directions still contains the required detail level to perform terrain masking. To avoid aliasing and poor quality caused by spatial undersampling the image is blurred with a $5 \times 5$ pixel Gaussian kernel with $\sigma = 1.1$ before downsampling (Gonzalez and Woods 2010, Szeliski 2011). Downsampling the blurred image to $\frac{1}{4}$ of the original size is done with bilinear interpolation in the 16 pixel neighborhood in the blurred image.

10.1.1 Optional pre-processing

In addition to the preprocessing step described above two optional steps to remove image artifacts were implemented. They have work very well, but require a lot of processing time and the effect on the end result is minimal when using entropy as a measure of terrain features. To maintain an effective processing workflow they are excluded, but
kept as optional. These methods are mentioned here because they are interesting and may be useful for many other purposes.

**Lighting Correction**

The input images are heavily vignetted because of different light conditions in the image. Vignetting in the image is removed by applying a vignette mask. The algorithm for creating the vignette mask \( g \) is simple. The image \( f \) is convolved with a very large average filter \( h(k, k) \) where \( k \) is the kernel size. Initial kernel size is one quarter of the width \( w \) of the original image.

\[
g(i, j) = f(i, j) \star h(k, k) , \quad k = \frac{w}{4} \tag{10.1}
\]

The resulting mask gives a good representation of the background illumination, but border features like fiducial marks and text boxes disturb the result. To remove these effects the mask is recursively convolved with a filter that reduces in size with a factor of two.

\[
g(i, j) = g(i, j) \star h(k, k) , \quad k = \frac{w}{4 \times 2^n} , \quad n = 1, 2, ..., k < \frac{w}{500} \tag{10.2}
\]

The algorithm stops when the kernel size \( k \) is smaller than the image width \( w \) divided by 500. Adding the vignette mask to the original image will create a corrected image \( f_{corr} \).

\[
f_{corr}(i, j) = f(i, j) + g(i, j) \tag{10.3}
\]

Figure 10.1 shows the vignette mask and how it corrects vignetting in an image. Effects from border features can still be observed.

![Figure 10.1](image)

Figure 10.1: Different lighting conditions in the image is compensated for by adding a vignette mask to the original image. Mistakodden, 1936.
Removing banding effects from digital scanner

Vertical banding effects introduced by a digital scanner are present in all the original images. One example of this is shown in figure 5.6 and it is even clearer in the entropy response images as seen in figure 5.7b.

Removing the vertical banding effect can be done in the frequency domain where it appears as a horizontal line (figure 5.6b). The image is transformed into the frequency domain with the Fourier transform (section 5.3.4) where the banding is reduced by applying a vertical median filter, to both the real and imaginary part, along the horizontal frequency response of the banding (figure 5.6c). When the image is brought back into the spatial domain with the inverse Fourier transform the banding effect is heavily reduced (figure 5.6d). This is a very simple approach filtering the complex response directly without extracting magnitude and phase.

Sloppy filtering in the frequency domain can introduce artifacts in the image. The simple median filter does a good job, but because this operation has been removed from the main processing workflow very little time has been used to optimize it.

10.2 Entropy Mask

When segmenting object from background a entropy filter is a good choice (Pun 1980, Kapur, Sahoo, and Wong 1985, N. Pal and S. Pal 1989). The entropy $E$ filter with kernel size $(u,v)$, as described in section 5.3.5, is applied to a image $f$ to produce a entropy response image $H$ as shown in figure 5.7b. The kernel $(u,v)$ used on the images is a circular kernel with size 15 by 15 pixels.

$$H(i,j) = E(u,v) \star f(i,j) \quad (10.4)$$

The resulting entropy response image, created with scikit-image, contains information about the level of disorder or uncertainty in the neighborhood around every pixel. In figure 5.7b one can easily observe that the tendency is that water, clouds and horizon get low values while glaciers and mountains get high scores. To successfully segment features in the image a good method for separating interesting from uninteresting features must be found. Numerous methods for segmentation have been proposed in the literature (Pun 1980, Kapur, Sahoo, and Wong 1985, N. Pal and S. Pal 1989). They usually involve statistical analysis of the histogram to determine the optimum threshold value. In this thesis a simple statistical analysis based on image samples is used.

Segmentation is calculated based on samples from the terrain features water, clouds,
glacier, foreground topography and background topography (figure 10.2a-e). Entropy is calculated for all feature samples and the values are entered into a boxplot (figure 10.2f). Based on the boxplot the internal spread of values within the feature samples and how the samples relate to each other can be analyzed. The boxplot reveals that there is a quite good separation between the background classes water and clouds and the foreground classes glacier, foreground topography and background topography. The biggest problem lies in separating background topography and clouds in the horizon. Based on visual inspection and trial and error a threshold value of $T = 4$ seems to be a reasonable choice. It is shown as the red horizontal line in the boxplot (figure 10.2f). A entropy response mask $g(i,j)$ can then be created with thresholding.

$$g(i,j) = \begin{cases} 
1 & \text{if } H(i,j) > T \\
0 & \text{if } H(i,j) \leq T 
\end{cases}$$ 

(10.5)

The global thresholding method will likely produce a fairly good result, but it is based on intuition and visual inspection. When dealing with statistical problems intuition and visual inspection like the method above is not a good idea, even if it produces fairly good results. A more robust method must be applied.

To separate the background classes water and clouds and the foreground classes glacier, foreground topography and background topography into a single mask k-Nearest Neighbor (kNN) is applied to the dataset. The samples shown in figure 10.2 are used as training data. The samples are to large to be used as an effective kNN training dataset because the vast number of pixels will slow down the process. Bilinear resampling is used.
to reduce each of the samples to 100 representative points, creating a total of 500 points in the kNN training dataset. Each of the 500 points have values in both entropy and intensity, and these are used to define the feature space as shown in figure 10.3a.

While having 5 classes is interesting for many purposes it is not relevant when we only want to create a binary mask separating interesting features from uninteresting ones. As illustrated in figure 10.3b the classes are merged based on whether they should be masked out or not. This leaves us with the two classes mask and no mask.

When kNN assigns classes to new features it is done by calculating distance to features in the training dataset. The distance is found by calculating Euclidean distance in feature space. As seen in figures 10.3a and 10.3b the range of the axis in feature space are very different. While entropy goes from 2 to 7 intensity goes from 0 to 255. This means that intensity will have a very large influence on the distance calculation compared to entropy. When the task is to classify terrain feature it should be the other way around since entropy separates the features well (figure 10.2) while intensity does not. Intensity should be considered a supplementary measurement in the process of classifying terrain features. To model this the intensity is scaled from 0-255 to 0-1. This approximately establishes a 1:10 relationship between intensity and entropy in the distance calculation in kNN (figure 10.3c).

![Figure 10.3: Terrain features in feature space where vertical axis is entropy and horizontal axis is intensity.](image)

The kNN algorithm is applied with $k = 3$ neighbors. Every pixel in the input image after preprocessing (section 10.1) is tested against the training dataset to create a binary mask as shown in figure 10.4.

Unfortunately, the data samples in figure 10.2 does not perfectly represent the reality captured through a airborne camera over Svalbard in 1936. The samples represent an ideal case and not reality. Borders between terrain features and the internal variation is more fuzzy than the homogeneous samples that are selected. The small boxes, long
whiskers and numerous outliers in the boxplot (figure 10.2f) shows this to some extent. The kNN method applied to create the masks is not able to model this fuzzyness good enough to make a perfect masking result. Typical examples that are hard or impossible to classify with entropy are scattered or complex cloud formations that get high entropy responses or flat, homogeneous plains that get low entropy responses. Figure 10.4 shows a mask created from thresholding the image with kNN based on entropy and to some extent intensity. The result is quite noisy and the fuzzy nature of terrain features can be observed.

![Figure 10.4: Image masked with mask created from thresholded entropy response.](image)

### 10.3 Post-processing

The mask created from thresholding entropy response values is very noisy and not very well suited as a mask for dense correlation. To remove the noise a combination of morphology and area-based filtering is applied.

Recursive morphological closing of the mask $g(i,j)$ with a circular kernel $b(k,k)$, where $k$ is the kernel size, will remove small noise particles and cluster noise particles that lie close to each other. Closing will preserve wanted features while unwanted are removed. The closing starts with a kernel size of $k = 3$ and increases with 3 until the last closing where $k = 30$. Recursive closing will give a cleaner result than just closing with the largest filter directly.

$$g(i,j) = g(i,j) \ast b(k,k) , \quad k = 3 \times n , \quad n = 1,2,...,10 \quad (10.6)$$

Figure 10.5b illustrates how the recursive morphological closing removes and clusters
As a last cleaning procedure a area filter is applied to the mask to remove all areas that are smaller than 2% of the total image area. This is applied to both background (black) and foreground (white). First the mask \( g \) is dilated with a circular kernel \( b(k, k) \) with \( k = 15 \) to separate background areas that are loosely connected.

\[
g(i, j) = g(i, j) \oplus b(k, k) , \ k = 15
\]  

(10.7)

The area filter is applied and all background areas that are smaller than 2% are converted into foreground. The mask is then eroded to separate foreground features. Erosion is done twice to compensate for the earlier dilation.

\[
g(i, j) = g(i, j) \ominus b(k, k) , \ k = 15
\]  

(10.8)

Foreground features smaller than 2% of image size are converted to background. To restore feature sizes the mask is dilated (equation 10.7) as the last step. The final image mask and the masked image can be seen in figures 10.5c and d.

To save disk space and make the mask compatible with MicMac the mask is made bitonal by changing pixel bit depth from 8 to 1 bit with ImageMagick. In addition xml-files with image descriptions are written for MicMac.
Figure 10.5: Entropy threshold cleaned with morphological and area-based filters.
10.4 Mask Quality

The resulting masks are generally quite good. The algorithm performs well in separating land and sea. Clean white snow is masked away while the rougher outer parts of glaciers are kept in the image. White snow may also be a problem in the dense matching process so this is not a big problem. The largest challenge with this kind of foreground/background separation is clouds and the horizon. They often cause problems and their irregular nature often cause them to be classified as foreground. These problems are very difficult or maybe impossible to overcome with a simple entropy algorithm. Figure 10.7 contains various masks and illustrates the strengths and weaknesses of the algorithm.

Efforts were made to strengthen the filter by applying more methods. Gabor filters are a class of filters that were first introduced as 1D filters in signal processing by Gabor 1946. The spatial 2D Gabor filter was introduced by Daugman 1985 and is the product of a 2D Gaussian envelope and a 2D sinusoidal plane wave. It is parameterized by the standard deviation of the Gaussian envelope ($\sigma$), the wavelength of the sinusoidal ($\lambda$) and the orientation ($\theta$). Figure 10.6 shows how these parameters influence the Gabor filter.

![Gabor kernels](image)

(a) $\sigma = 3$, $\theta = \frac{\pi}{4}$, $\lambda = 5$  
(b) $\sigma = 5$, $\theta = \frac{\pi}{4}$, $\lambda = 5$  
(c) $\sigma = 5$, $\theta = \frac{\pi}{2}$, $\lambda = 5$  
(d) $\sigma = 5$, $\theta = \frac{\pi}{4}$, $\lambda = 10$

Figure 10.6: Gabor kernels with different combinations of $\sigma$, $\theta$ and $\lambda$.

The 2D Gabor filter has proven helpful in texture segmentation by storing several of them in a filter bank and comparing the response in the image to be classified with the response from a set of labeled sample images (D. Dunn and W. Higgins 1993, Weldon, W. E. Higgins, and D. F. Dunn 1996, Li and Staunton 2008). The main purpose of implementing this method was to classify clouds and horizon as this is the main problem with the entropy algorithm. The method was not very helpful for the Svalbard 1936 images. It is likely because the clouds and the horizon have very little texture and their properties vary from image to image. The Gabor filter ended up classifying everything lacking texture, which is very similar to what the entropy filter does, but the results were not as good.

To further improve the the masking result it may be required to try something new. One very interesting research field is the application of deep learning using convolutional
neural nets for image analysis. Much of the theoretical foundation is quite old, but the implementation require processing power that has not been easily available until recently. The research field evolves very quickly and there are promising results when the method is used for segmenting normal images (Long, Shelhamer, and Darrell 2015) and remote sensing images (Romero, Gatta, and Camps-Valls 2016). Deep learning using convolutional neural nets is outside the scope for this thesis.
Figure 10.7: Examples of entropy masks. Some images get very good masks while other images get poor masks. The overall performance of the masking algorithm is good.
11 Dense Point Cloud Creation

The next processing step is to apply multiple-view stereo methods to create a dense point cloud from the images. After bundle adjustment (section 9.1) a strong relationship between the images is established. The relationship between any two images can be described with the essential and the fundamental matrix. The essential matrix describes the translation and rotation between the images in world coordinates, while the fundamental matrix in addition to this contains the camera intrinsics so that we can describe the relationship between the two images in pixel coordinates. Using this information the relationship between the images can be represented in *epipolar geometry*. (Faugeras, Luong, and Papadopoulo 2001)

Figure 11.1 illustrates epipolar geometry between two cameras $O$ and $O'$. The line between the two camera centers $O$ and $O'$ is known as the *baseline* and it will remain constant in the epipolar geometry between two cameras. When projecting a point $X$ in the scene back into the two cameras a *epipolar plane* is formed between the point $X$ and the two cameras $O$ and $O'$. In the image planes the epipolar plane is represented as lines $l$ known as *epipolar lines*. The point where the epipolar line intersects the baseline is known as the *epipole* $e$. In the image plane of one camera the epipole is the position of the other camera, but it may lie outside the image and therefore not be visible. If the point $X$ is moved around in the scene new epipolar planes and lines will be formed, but the baseline and epipoles remain constant.

![Figure 11.1: Epipolar geometry](OpenCV: Epipolar Geometry 2017)

The most important property of the epipolar geometry is that it allows us to establish point-line relationships between the two images. A point in one image is a line in the other. Figure 11.1 illustrates this relationship very well. If a point $x$ in the image plane
of camera $O$ is projected into the scene as $X$ it will appear somewhere along the epipolar line $l'$ in the image plane of the other camera where $x'$ is different possible positions. It is not possible to find the exact position $X'$ in epipolar geometry with two images, but the possibilities are reduced to a line in image plane or a vector in world coordinates.

To find the exact positions of points in world and image coordinates a method called plane sweep is applied. From the epipolar geometry we know that a point $X$ in world is located along a vector going through the pixel $x$ and the camera center of one image. We also know that the same point $x'$ is located somewhere along the epipolar line in the other image. The information we want to find is the distance between the camera center and the point $X$. The simplest approach is to define one camera as the reference camera and define a plane parallel to the image plane at a distance $d_m$ from the image center along the normal of the plane as shown in figure 11.2a. By projecting both images into the plane a correlation between the two images can be calculated for every corresponding pixel. By changing the distance $d_m$ at given intervals and calculating the correlation every pixel is assigned a correlation value for all $d_m$. Selecting the highest correlation value for every pixel and assigning it the corresponding $d_m$ will result in a three dimensional coordinate in the reference system of the reference camera for every pixel. These can then be projected into world coordinates. Plane sweep using planes parallel to the image plane is often referred to as fronto-parallel (Haavardsholm 2016).

The plane sweep using a fronto-parallel plane is simple and effective for many cases, but a major drawback is that it is only able to find points that are visible from the reference camera. Points that are hidden from the reference camera will not be a part of the output point cloud. (Collins 1996)

An alternative approach is to define the correlation plane that is not parallel to any of the image planes. The most common is to define it parallel to the ground. The distance $d_m$ will then have the same direction as the ground normal. Images are projected into the ground plane before they are correlated. This enables us to extract ground elevation directly. It also allows points to be correlated from different angles and the result is not confined to what is visible from a reference camera. Figure 11.2b illustrates a plane that is not oriented parallel to any of the image planes. Plane sweep using planes parallel to the ground is often referred to as ground normal (Haavardsholm 2016).

Using ground normal planes give good results, but they require images where the angle between the optical axis and the ground normal is not too large. If the angle between the ground normal and the optical axis of the camera is too large it will distort pixels and make matching difficult.

When doing multiple-view stereo it is a huge advantage to have many images covering
the same area. Having only two images we are able to perform the matching process, but it is not possible to check the validity of the result. To check the result three overlapping images are required. As mentioned earlier epipolar geometry allows us to project a point in one image to a line in another image. If plane sweep gives us the pixel position of point $X$ in three different images the result can be checked using epipolar geometry. By projecting the point $X$ from any two of the images into the third the two epipolar lines will intersect in the point $X$ in the third image. If they do not something is wrong. (Opsahl 2016a).

The Svalbard 1936 images are converted into a point cloud using MicMac. The images are high-oblique and the optical axis of the camera is pointed at an angle of about 70° off the vertical axis. Using a ground normal plane sweep for dense correlation was not possible and the fronto-parallel method with a reference image was used instead. In addition to this the images have very little overlap and the minimum number of images used in the correlation was set to two. This means that the correlated result can not be controlled with epipolar geometry in some areas. The images were processed in groups of three where the middle one was used as reference plane or master. In MicMac the Malt command was used in the mode GeomImage. The masks created in chapter 10 were used to avoid dense correlation in certain areas.

After processing with Malt the correlated results were turned into point clouds with Nuage2Ply. Point clouds in Ply-format are stored as 32-bit single-precision floating-point often referred to as Float32. Numbers are stored with 32 bits where 1 bit is sign, 8 bits are exponent and 23 bits are fraction. This means that larger numbers are stored with lower
precision. (Single-precision floating-point format 2017) Storing the full UTM coordinate for every point will have a large effect on the precision. The larger Y-coordinate will be stored without decimal spaces. This means that all Y-coordinates are rounded to the closest meter. The resulting point cloud will appear strange with all the points lined up in rows. The lining effect will likely have a negative effect on the final result as it will influence interpolation. To prevent this the points are stored in a local coordinate system by applying a global shift. Table 11.1 shows the shift parameters.

Table 11.1: Global shift values for point clouds stored in ply-format.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>600000 m</td>
</tr>
<tr>
<td>Y</td>
<td>8600000 m</td>
</tr>
<tr>
<td>Z</td>
<td>0 m</td>
</tr>
</tbody>
</table>

Processing all the images resulted in 89 point clouds with a total size of 110.3 GB as shown in table 11.2. Figure 11.3 shows one of the dense point clouds output from the process.

Table 11.2: Number of point clouds and total data size

<table>
<thead>
<tr>
<th>Line</th>
<th>Count</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>20</td>
<td>25.6 GB</td>
</tr>
<tr>
<td>South</td>
<td>33</td>
<td>41.8 GB</td>
</tr>
<tr>
<td>East</td>
<td>12</td>
<td>16.7 GB</td>
</tr>
<tr>
<td>West</td>
<td>24</td>
<td>26.2 GB</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>110.3 GB</td>
</tr>
</tbody>
</table>

87
Figure 11.3: Dense point cloud showing Jeppeberget on the east coast of Barentsøya. Freemanbreen can be seen in the background.
12 Point Cloud Processing

The output point clouds from the dense correlation process are not very useful without further processing. To make them useful for other purposes than viewing they must be processed. This chapter will focus on automatic processing steps that will result in orthophotos and digital elevation models.

When processing the Svalbard 1936 point clouds a choice must be made whether to process them individually or as one large cloud. Processing as one large point cloud has some advantages. More data will give a better statistical baseline. It will enable radiometric processing of the point clouds (Luc Girod and Pierrot-Deseilligny 2014) and the process of removing noise along the edges of individual point clouds is easier. Despite the advantages of processing as one cloud it was discarded because of the large amount of data (table 11.2) and individual point cloud processing was implemented.

12.1 Point Cloud Cleaning

The raw point clouds from dense correlation are noisy and contain errors that must be minimized or removed before extracting digital elevation models and orthophotos. The errors in the point clouds are mainly caused by masks that are not perfect while noise can be caused by poor matching.

PDAL is used to implement an automatic cleaning workflow. It is a multiple step analysis implemented through the PDAL pipeline functionality. Custom pipelines are generated with Python for every image to enable the use of the camera positions extracted in section 9.2. Figure 12.1 shows the process involved to clean the noisy point clouds. Every step of the process will be explained in detail in the following sections.

12.1.1 Crop Z-axis

The first step in the cleaning process is to remove extreme elevation values. From the datasets provided by Norsk Polarinstitutt it is possible to identify the maximum and minimum elevation values on Barentsøya. The lowest point is 0 meters along the coastline while the highest point is Solveigdomen at 666 meters. To prevent removing useful information a buffer value is added to the maximum and minimum values. Based on the residuals after bundle adjustment (table 9.1) a buffer value of 50 meters is chosen. This
Figure 12.1: Processing steps involved to clean point clouds.

gives us the extreme values $z_{\text{min}} = -50m$ and $z_{\text{max}} = 716m$. All points with z-values outside this interval are removed with PDAL using the `filters.range` method.

### 12.1.2 Crop Range

The next filter is a range filter based on the range or distance $d$ between every pixel $P$ and the camera center $C$. Accurate camera positions for every camera extracted in section 9.2 makes it possible to calculate the distance $d$. Note that the camera positions from section 9.2 have true coordinates in ETRS89 utm zone 33 while the point clouds are in a local coordinate system shifted according to table 11.1. Before calculating range the camera positions are translated into the local coordinate system. The distance $d$ can now be calculated for every point.

$$
d = |P - C| = \sqrt{(P_x - C_x)^2 + (P_y - C_y)^2 + (P_z - C_z)^2} \quad (12.1)
$$

Finding the range $d$ for every point is done with a custom Python script in PDAL through `filters.programmable`.

Now that the range is known pixels can be filtered using the `filters.range` method in PDAL, but this requires a threshold distance. Instead of applying a global threshold range, individual threshold distances are calculated for every camera position based on its height above sea level.

It is assumed and observed that point cloud quality will decrease as distance from the camera increases. This is partly because ground resolution decreases and partly because the angle of intersection between the vector from the camera to the point ($\vec{CP}$) and the ground decreases. Instead of using distance directly it is calculated from the intersection angle between a vector from the ground plane ($z = 0$) to the camera and its relation to camera height $h$. 


Setting the angle $\alpha$ to 10° gives an approximate $d_{\text{max}}$ of 17 kilometers. While this does not give an intersection angle in every point it produces a $d_{\text{max}}$ based on camera height. Calculating the true intersection angle for the surface in every point would require calculating the normal vector in every point and comparing it to the camera position. A simple intersection angle with the ground plane ($z = 0$) requires less calculations and gives a good enough result.

All points with range $d$ larger than $d_{\text{max}}$ are discarded.

### 12.1.3 Remove Outliers

To remove outliers from the point cloud a statistical approach created by Rusu et al. 2008 is applied through the statistical method in PDAL filters.outlier.

For every point $p_i$ the mean distance $\mu_i$ to its $k$ nearest neighbors where $p_j$ is the $k$-th nearest neighbor of $p_i$.

$$\mu_i = \frac{1}{k} \sum_{j=1}^{k} |p_i - p_j|$$

(12.3)

From $\mu_i$ the global mean $\bar{\mu}$ can be calculated along with the global standard deviation $\sigma$.

$$\bar{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mu_i$$

(12.4)

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (\mu_i - \bar{\mu})^2}$$

(12.5)

Based on $\bar{\mu}$ and $\sigma$ a global threshold value $t$ is calculated using a multiplier $m$.

$$t = \bar{\mu} + m\sigma$$

(12.6)
A second iteration over the dataset is performed to locate outliers.

\[
\text{outlier}_i = \begin{cases} 
\text{true} & \text{if } \mu_i \geq t \\
\text{false} & \text{if } \mu_i < t 
\end{cases}
\]  

(12.7)

For the Svalbard 1936 dataset \( k = 8 \) and \( m = 2 \). All points marked as outliers are removed.

### 12.1.4 Ground Return Filter

The result from the outlier removal has removed extreme values, but it does not attempt to model the actual surface. To model the surface and remove non-ground points the Simple Morphological Filter (SMRF), created by Pingel, Clarke, and McBride 2013, is applied to the point cloud through PDAL filters.smrf.

The SMRF algorithm uses four steps to filter ground returns from a point cloud (Pingel, Clarke, and McBride 2013):

1. A minimum surface \((ZI_{\text{min}})\) with a given cell size is generated from the point cloud. SMRF is intended for LIDAR data where the points are labeled with return values, but these do not exist in our point cloud. All points are therefore treated as last returns.

2. A progressive morphological filter is applied to the surface. This is a series of multiple morphological openings with a circular kernel that increases in size from 1 to \( \frac{\text{window radius}}{\text{cell size}} \). For every iteration the difference between the opened surface and \( ZI_{\text{min}} \) is calculated and cells in the opened surface are labeled bare earth (BE) and object (OBJ) based on the slope tolerance parameter.

3. A provisional DEM \((ZI_{\text{pro}})\) is created from the points marked as BE.

4. The final step is to calculate vertical distance between the input point cloud and \( ZI_{\text{pro}} \) and threshold points based on the parameter elevation threshold. Points in steep slopes are likely to have a larger vertical displacement than points in flat terrain. Making the threshold slope-dependent. The parameter scaling factor controls how slope-dependent the threshold will be.

In their paper Pingel, Clarke, and McBride 2013 presents optimized algorithm parameters for various surfaces. The one that fits Barentsøya best is one called steep, terraced slopes (5-3). It has no vegetation and the steep irregular surface is well suited to preserve the ridges and crevices of Barentsøya. In the paper a cell size of 1 meter is used. On
Barentsøya it is increased to 2 meters and therefore the window size is increased by a factor of two. Table 12.1 shows the parameters used in the ground filtering. Other than cell size and window size the parameters are identical to the optimized parameters for surface 5-3.

Table 12.1: SMRF parameters for ground filtering. The parameters are identical to those optimized for terrain type 5-3 (steep, terraced slopes) except for cell size and window radius that are doubled.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell size (m)</td>
<td>2</td>
</tr>
<tr>
<td>Slope tolerance (dz/dx)</td>
<td>0.45</td>
</tr>
<tr>
<td>Window radius (m)</td>
<td>6</td>
</tr>
<tr>
<td>Elevation threshold (m)</td>
<td>0.10</td>
</tr>
<tr>
<td>Scaling factor</td>
<td>3.80</td>
</tr>
</tbody>
</table>

12.2 Raster Creation

The cleaned point cloud is used as input for raster data creation. Raster data is created by bringing the data from 3D to 2D. The x and y coordinates are preserved in the 2D grid filled with attribute values from the point cloud. For the Svalbard 1936 point clouds 3 2D grids are created from the point cloud. Orthophoto (intensity), digital elevation model (z) and distance from camera (range) are extracted into 2 meter resolution rasters. Examples of these are shown in figure 12.3.

The transformation from 3D to 2D is done with GDAL through the PDAL writer called *writers.gdal*. A grid with 2 meter cells is created for the x-y extent of the point cloud. For the output dimension (intensity, elevation, range) every cell value is given the mean value of all the points within a given radius from the cell center. For the orthophoto and the range raster the radius is $\lceil \frac{2}{\sqrt{2}} \rceil$ meters. The elevation model is slightly smoothed by setting the radius to $\sqrt{8}$ meters.

To remove small holes and imperfections in the rasters one additional processing step is applied. Cells that do not have values are assigned values from their neighborhood if there exist any neighbors. A $5 \times 5$ window is created around unassigned cells. Inverse distance weighting (Shepard 1968) is applied to calculate the new cell value.

All point cloud calculations have been done in the coordinate system *ETRS89 utm zone 33* with shifted coordinates according to table 11.1. To bring the rasters to their correct position the geotransform of the rasters are translated according to the shift parameters.
Figure 12.3 shows the final result. By observing the output rasters one can see that there are large areas without cell values. The larger ones appear in areas that are not visible from the camera while the smaller ones are areas where the dense correlation did not succeed. The number of cells without data increase as the distance from the camera increases. This is an expected result.

Figure 12.3: Output rasters with 2 meter resolution generated with PDAL and GDAL. The rasters are created from a point cloud with image S36 3297 as master.

12.3 Raster Mosaicing

The individual rasters created from point clouds are generally good, but they have some obvious weaknesses that are revealed on closer inspection. Areas that are not visible from the camera position will cause holes without data. Data along raster or image edges also have poor quality. In general the central part of rasters have the best quality and the quality decreases as the distance from the camera increases. These properties will be used to generate a mosaic containing the best possible data.

The range rasters created in section 12.2 are used to find the pixel closest to its camera for every pixel value. GDAL is used to stack the range rasters $R$ on top of each other in $n$ bands and NumPy is used to extract an index $I$ describing which image that is best suited for every pixel.

$$I_{(x,y)} = \text{argmin}(R_{(x,y)1}, R_{(x,y)2}, \ldots, R_{(x,y)n})$$  \hspace{1cm} (12.8)

This approach extracts the best part of every image, but it also allows holes in images to be filled with information from neighboring images. The index is used to mosaic the images together into an orthophoto and a DEM. Figure 12.4 illustrates how the index
assigns pixel values from different source images to create a mosaic with the best possible coverage. One can observe how the areas with no coverage in one image is assigned pixel values from neighboring images. This does create spotted patterns in some areas in the orthophoto, but it is better than no coverage.

There are methods for correcting color differences between images in image mosaics. This is a well studied field and robust solutions to the problem exists (Luc Girod and Pierrot-Deseilligny 2014). In photogrammetry the color differences are referred to as radiometric heterogeneities and they can be corrected or equalized within images or between images. One example is Radiometric Aerial Triangulation (Chandelier, Martinoty, et al. 2009) which uses a statistical model based on tie-points in areas with overlapping images. Radiometric correction is also available in MicMac (Deseilligny, Rupnik, et al. 2017).

Radiometric correction is not applied in the very simple mosaicing algorithm presented here. It would likely improve the result and may be added in the future.

Figure 12.4: Raster mosaic with 2 meter resolution of the east coast of Barentsøya. The index is used to assign pixel values to the DEM and the orthophoto from the input images.
Part IV

Results
13 Complete Models of Barentsøya

The examples in section 12.3 all show models of the east coast of Barentsøya. This was done because the examples are better when a small area is shown in high resolution. The method was applied to all the images creating larger datasets. Datasets containing all data are displayed in this chapter. Detailed explanation is located in the figure texts.

Figure 13.1: Dense point cloud of Barentsøya seen from south. To reduce file size the point clouds created in section 12.1 are downsampled and merged together. To remove noise from poor matches in water it has been clipped with a polygon shaped as Barentsøya (Norwegian Polar Institute 2016a). This dataset is intended for visualization only and is not a part of the processing workflow.
Figure 13.2: Image mosaic of Barentsøya created from the available images. Ground resolution is 2 meters. The coverage is quite good, but there are some holes in areas that were not visible from any of the cameras or covered by clouds. Radiometric heterogeneities or differences in color are quite clear. It is easy to spot the seamlines between images. The quality of the result will be discussed more in chapter 14.
Figure 13.3: Digital Elevation Model of Barentsøya created from the available images and visualized with hillshading. Ground resolution is 2 meters. The coverage is quite good, but there are some holes in areas that were not visible from any of the cameras or covered by clouds. When all the elevation models are mosaiced together it becomes clear that the georeferencing of the z-axis is different on the flight lines (north, south, east, west). Steps on the z-axis appear between north and west flight lines and between west and south flight lines. The quality of the result will be discussed more in chapter 14.
Figure 13.4: Digital Elevation Model of Barentsøya created from the available images and visualized with colored elevation intervals and a slight hillshade. Ground resolution is 2 meters. The coverage is quite good, but there are some holes in areas that were not visible from any of the cameras or covered by clouds. The elevation intervals reveal that parts of the model lies below sea level. This error is largest in the northeastern and southwestern parts of the model. The quality of the result will be discussed more in chapter 14.
14 Quality of the Results

This chapter will discuss the quality of the orthophoto and DEM produced in part III and presented in figures 13.2, 13.3 and 13.4. Errors and artifacts that are visible in the results will be discussed. These can be identified without comparing with other products. The second part will focus on accuracy and precision compared to other products. Separating accuracy and precision is important because it makes it possible to discuss properties of a dataset even if absolute accuracy is poor. Horizontal and vertical quality is discussed separately.

14.1 Errors and Artifacts

When using image matching techniques like SfM errors will always occur. These errors are often the result of noise in the images or poor image matching in difficult areas. A lot of effort has gone into minimizing these error sources in the processing workflow presented in part III but the workflow does not succeed in eliminating them completely. This section will present typical errors that must be expected when applying the workflow from part III.

14.1.1 Errors related to noise

As mentioned in chapter 11 using high-oblique imagery to create digital elevation models and orthophotos will give a very large angle between the image plane and the ground plane. The consequence is that the pixels of the original image will be rectangular in the ground plane. As the distance from the camera increases the pixels will become increasingly elongated with the longest side parallel to the optical axis. The large horizontal extent of every pixel will make the result less precise and introduce noise.

Along with rectangular pixels there is another design choice that will increase the chance of noise appearing in the result. Because of low overlap between the images dense point cloud creation was allowed with matches from only two images. As explained in chapter 11 this removes the ability to check the validity of the match in epipolar geometry. This increases the chance of noise.

The effect of noise is easily visible in the DEM and it will appear as a rough surface even in flat terrain. Resampling the DEM to a lower resolution will remove a lot of the
noise and show a smoother surface. Figure 14.1 shows the rough surface of the 2 meter DEM and a profile of the presumably flat surface on top of Jeppeberget. When the DEM is resampled much of the noise is removed and the surface will appear smoother. Creating a 2 meter DEM of the 1936 high-oblique images may be a bit optimistic. A DEM with 10 or 20 meter resolution may be better.

Figure 14.1: A hillshade visualization of the 2 meter DEM reveals that it is rough and noisy (a). Resampling the DEM to lower resolutions will remove a lot of noise and give a better representation of the presumably flat surface on top of Jeppeberget (b).

14.1.2 Errors related to matching

One of the largest matching challenges with the Svalbard 1936 images is water. It is impossible to match and it will therefore introduce large errors if not masked before dense correlation. The masking algorithm presented in chapter 10 is quite good at detecting water, but it does not manage to remove everything. This creates large errors in areas where water is matched. The errors from water matching will in some areas affect the shoreline making it wrong. Poor matching in water can be seen all along the shoreline. In figures 13.2, 13.3 and 13.4 matching errors in water are clearly visible off the east coast of Barentsøya.

14.1.3 Artifacts from image mosaicing

The simple image mosaicing algorithm from section 12.3 does not apply radiometric correction. The orthophoto is not homogeneous and will have sharp edges between
images and areas filled with information from other images. This problem only applies to the orthophoto which gets its pixel values directly from the Svalbard 1936 images. The effect is not visible in the DEM because the elevation pixels are derived from the geometry (z-values) of the point clouds, which are not affected by intensity values in the images.

Figure 14.2: The simple image mosaicing algorithm from section 12.3 does not apply radiometric correction. The result is not very visually appealing, but it is not wrong.

14.2 Accuracy and Precision

Accuracy and precision are very important characteristics of the result. In this discussion accuracy is how close a result is to the absolute position or truth, while precision is a relative measure of variance, random errors or details.

The residuals of the GCPs after bundle adjustment (section 9.1) provide a good starting point for the accuracy and precision discussion. However the tabular form of the residuals (table 9.1) makes it hard to get a good understanding of the errors. A boxplot visualization is provided in figure 14.3 to make interpretation easier.

The boxplots only provide information on residuals or error in the GCP positions, but they do give a good indication of the quality of the DEM and the orthophoto. They give individual measures on the directions $x$, $y$ and $z$ as well as total residual. The residuals of the flight lines are slightly different, but the overall impression is that a error of 50 meters must be expected. However the residuals are only valid for the GCP positions. This means that points between the GCPs may have other values. One important quality of the bundle adjustment is that it will try to minimize the total error in the system. The mean error will therefore be relatively close to zero. This can be observed in the residual boxplots (figure 14.3) and the residual tables (table 9.1). Given that the GCPs are well distributed the accuracy may be better than what is indicated by the residuals.

While the residuals provide a good starting point, accuracy and precision is tested
using other methods as well. The remainder of this section will discuss this using comparison to other datasets.

![Boxplots of GCP residuals from bundle adjustment (section 9.1).](image)

Figure 14.3: Boxplots of GCP residuals from bundle adjustment (section 9.1). They are in meters in $x$, $y$, $z$ directions and total distance.

### 14.2.1 Horizontal Accuracy

The horizontal accuracy of the DEM and the orthophoto is hard to judge because of the coarse resolution of reference data. This is the same problem encountered when collecting GCPs in section 9.1. Based on comparisons with data from Norsk Polarinstittut and TanDEM-X the products seem to have a maximum error of about 50 meters. This observation corresponds to the observation based on residuals. The relatively high errors
are likely caused by poor GCPs. It is likely that better data for GCP selection is required to produce data with higher accuracy from the Svalbard 1936 images.

### 14.2.2 Vertical Accuracy

Figure 14.4 displays the vertical accuracy of the Svalbard 1936 DEM compared to a DEM from Norsk Polarinstittut with 20 meter resolution. The errors range from $-53m$ to $+115m$, but the majority lies between $-25m$ and $+25m$. This observation fits well with the residuals in the $z$ direction that can be observed in figure 14.3 and table 9.1.

When examining the distribution of errors it is not random. The output Svalbard 1936 DEM is tilted. The tilt is not the same for the entire model, but the DEMs from the four flight lines *north*, *south*, *east* and *west* (figure 9.1) have different tilt. The northern DEM is tilted towards northeast, the southern DEM is tilted towards west, the western DEM is tilted towards north and the eastern DEM is tilted towards northeast. The tilt is caused by poor GCPs. It is possible to rotate, shift or scale the elevation models to correct them, but it has not been done in this thesis.

Unfortunately a simple tilt does not explain all errors in accuracy. Some are noise and poor matching while others seem to be actual differences in the terrain. If the terrain models in figure 14.4 are rotated so that the tilt errors are removed there will still be differences in certain areas. Some of these areas are large and continuous and they seem to fit the extent of the glaciers on Barentsøya. In figure 14.4 large blue or red areas depicting extreme errors, except the western part of the southern model, are glaciers. Areas surrounding the glaciers do not have the same extreme values. A likely explanation is that the elevation of the glaciers have changed.

To highlight this tendency the elevation difference dataset (figure 14.4) was clipped with a mask dataset of the large glaciers on Barentsøya. The glaciers were labeled and presented in the map shown in figure 14.5. Duckwitzbreen, Besselbreen, the higher part of Freemanbreen and the lower part of Hübnerbreen have higher elevation in the 1936 dataset than in the recent dataset from Norks Polarinstittut. The lower parts of Reymondbreen and Freemanbreen have lower elevation in the 1936 dataset.

<table>
<thead>
<tr>
<th>Glacier</th>
<th>Year Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freemanbreen</td>
<td>1955 - 1956</td>
</tr>
<tr>
<td>Duckwitzbreen</td>
<td>1918</td>
</tr>
<tr>
<td>Reymondbreen</td>
<td>1956</td>
</tr>
<tr>
<td>Hübnerbreen</td>
<td>between 1930 and 1936</td>
</tr>
</tbody>
</table>
The glacier observations from the Svalbard 1936 elevation model are supported by previous studies of Barentsøya. Besselbreen has had a continuous retreat, without surges, since its assumed Little Ice Age extent (Lefauconnier and Hagen 1991) and was higher in 1936 than it is now. In table 14.1 the surge year is listed for surging glaciers on Barentsøya. Hübnerbreen and Duckwitzbreen surged before the images were taken in 1936 and have retreated ever since. Their elevation were higher in 1936 than now. Freemanbreen and Reymondbreen have surged after the images were taken and their elevation is higher than it was in 1936. Having higher accuracy by using better GCPs or correcting the DEM might have enabled accurate glacier mass calculations from 1936.

14.2.3 Precision

As discussed in sections 14.2.1 and 14.2.2 the accuracy is quite poor and it is likely caused by inaccurate GCPs. While the accuracy is poor this does not necessarily affect the precision. In this section precision will be treated as a local phenomenon and a measure of how detailed the DEM is able to describe the terrain.

To describe precision the Svalbard 1936 DEM is compared to a DEM with 12 meter resolution from TanDEM-X (Wessel 2013) and a DEM with 20 meter resolution from Norsk Polarinstitutt. The precision is judged based on visual inspection and with terrain profiles. Figure 14.6 shows the analysis.

When precision analysis is performed the differences between the three models is clear. The DEM created from Svalbard 1936 images has a lot more detail than the DEM from Norsk Polarinstitutt. Even if the 1936 model is resampled to 20 meters it will have a lot more detail. Compared to the DEM from TanDEM-X the 1936 model performs quite well, but it is not as good. Creating hillshade maps of the models make it easy to visually compare the level of detail in the models, but to inspect specific features as ridges and valleys profiles were created. Profiles reveal the high level of detail in the Svalbard 1936 DEM.

When inspecting the difference between the DEM from Norsk Polarinstitutt in figure 14.6a and the Svalbard 1936 in figure 14.6b the largest differences are located in areas where the terrain changes rapidly. Examples are ridges, peaks and riverbeds. The DEM from Norsk Polarinstitutt, used for GCP collection, fills riverbeds and rounds ridges and peaks. As explained in section 9.1 these distinct terrain features were used as GCPs. This is likely the cause of the large errors in accuracy.
Figure 14.4: Map showing the difference in elevation between 20 meter DEM from Norsk Polarinstittutt and DEM created from Svalbard 1936 images. It is clear that the S36 DEM is tilted and the individual lines are tilted independently of each other.
Figure 14.5: Map showing the difference in elevation between 20 meter DEM from Norsk Polarinstitutt and DEM created from Svalbard 1936 images. Glaciers, defined by a mask from the S100 dataset from Norsk Polarinstitutt, are highlighted and labeled. The large differences on glaciers are likely caused by changes in the amount of ice.
Figure 14.6: Precision comparison of three elevation models. Hillshade visualization is shown in (a), (b) and (c). Figures (d), (e) and (f) are profile comparisons.
14.3 Overall Quality Assessment

In this chapter various aspects of quality have been discussed. The overall assessment is that the result is quite good when the input is taken into consideration. With poor GCPs and old, noisy images a orthophoto and a DEM with reasonable quality has been produced. The products are noisy and have some accuracy issues, but the precision is good. With some tuning and better input data for georeferencing the method has the potential to produce good results based on old data.

An alternative to processing the data with better GCPs is to correct the models with post-processing methods. The data has high precision, and as indicated in section 14.2.2 the accuracy errors can likely be corrected with simple shift, scale and rotate operations. Nuth and Kääb 2011 present a method for statistical error modeling and correction of digital elevation models. It uses co-registration with another DEM to model statistical errors. A correction is calculated and applied to the DEM with poor accuracy. The method produces good results and it would likely improve the accuracy of the 1936 DEM if the flight lines are processed individually.

Regardless of whether high accuracy is ensured through accurate GCPs or with post-processing methods like co-registration the accuracy of the result is totally dependent on the accuracy of the data used for GCP collection or co-registration. In either method points that are considered true must be linked to the corresponding point in the model that is to be corrected or georeferenced.
Part V

Conclusion
15 Workflow Review

The complete processing workflow and its results have been presented and form the basis for a more thorough discussion. Chapter 7 introduced the workflow without presenting any details about the individual steps. This chapter will go into more detail and review how the individual steps and the entire workflow performed. Alternative solutions and possible improvements will also be mentioned.

![Workflow Diagram](image)

Figure 15.1: Workflow for processing Svalbard 1936 images into orthophotos and digital elevation models.

After deciding which images to use the first processing step is finding the positions of fiducial marks in the images. This is a crucial step as it allows us to solve inner orientation in the images and use them in digital photogrammetry. It is also the processing step that has been the most challenging and time consuming to implement. The fiducial mark finder uses many different image analysis and machine learning techniques to build a fast, robust and accurate workflow. The fiducial mark finder accurately locates the fiducial
mark reference point with a success rate of 99.82%. Even with such a high success rate it is very important to find the errors. The workflow will flag the error it is able to detect and an effective manual checking routine has been created to find any remaining errors. The fiducial mark finder is considered a success.

Based on the fiducial marks, inner orientation is calculated and a sparse point cloud is created by matching multiple images. The sparse point cloud has a relative orientation and GCPs must be collected to georeference it. GCP collection is likely the largest error source in the entire workflow. On Barentsøya it was difficult to find accurate data for georeferencing. In addition to inaccurate data, there are very few features on Barentsøya that make good GCPs. Most of the GCPs are located on distinct terrain features like ridges, riverbeds and peaks. From section 14.2.3 we know that these areas have the largest errors in the DEM from Norsk Polarinstitutt. The errors introduced by the GCPs stay with the data for the rest of the process and are clearly visible in the end result. It would be interesting to run the process again using the much better TanDEM-X data as elevation source for the GCPs. This would likely have resulted in a more accurate result while the precision would be unchanged. Unfortunately the TanDEM-X dataset became available late in the process and it will have to be done in the future.

The sparse point cloud is georeferenced and corrected with bundle adjustment. Before creating a dense point cloud individual masks removing unwanted objects from the images must be generated. This is done by calculating entropy for every pixel with a sliding window. Entropy is a fairly good approach when separating unwanted terrain features like water and clouds from wanted terrain features. The entropy image is classified using a kNN predictor trained with samples of different terrain types. The predictor performs fairly well, but the chosen samples are unable to represent the terrain perfectly. The kNN approach is very simple and there is a lot of room for improvement. It would be interesting to implement deep learning using convolutional neural nets for image analysis with a large database of samples, but it is outside the scope of this thesis.

With the georeferenced sparse point cloud and image masks as input, a dense point cloud is generated for every input image. The dense point cloud generation is done with sets of three neighboring images with the center image as master. Due to low overlap, points are generated even if they are matched only in two of the three images. The high-oblique angle of the optical axis and the low image density makes it difficult to improve this step for the Svalbard 1936 dataset.

Before creating digital elevation models and orthophotos the point clouds must be cleaned to remove noise and extreme values. The cleaning procedure was performed on every point cloud individually instead of merging them together and performing the
operations on one large point cloud. Performing the operation on one large point cloud might have been beneficial because it would have provided a better foundation for statistical analysis around the edges of the point clouds, but it was abandoned because of the large amount of data. Instead the point clouds were cropped by elevation and distance to camera before they were filtered with a ground return filter adapted to steep terrain with no vegetation. The point cloud cleaning procedure performs well, but struggles to remove some artifacts along the edges of the point clouds.

The last step in the processing workflow is raster creation. Again there are two processing options. The point clouds can be processed individually or they can be merged and processed as one large point cloud. Processing as one large point cloud has some advantages. It would make it possible to apply radiometric correction between the point clouds (Luc Girod and Pierrot-Deseilligny 2014), but processing as one large point cloud was once again abandoned because of the large amount of data. Every point cloud was converted into individual rasters and then mosaiced choosing the pixels that are closest to the camera for every location. This technique effectively removes artifacts along the edges of point clouds because these areas are covered by the neighboring image. This produces a mosaiced orthophoto and elevation model. The mosaicing works very well for the elevation values of the elevation model, but for the orthophoto the boundaries between images becomes very distinct because of differences in intensity values. The effect of different intensity values might have been reduced by applying a lighting correction algorithm like the one mentioned in section 10.1.1, but the most effective method would likely be to implement radiometric correction (Chandelier, Martinoty, et al. 2009) in the mosaicing algorithm. Radiometric correction would be a relatively simple way to make the orthophotos more visually appealing.

The overall performance and design of the workflow is good, but there is room for improvement. Good GCPs is however the main success criteria when recovering old data with new methods.
16 Software Review

An important element in this thesis is to do all the work with open source software. It is not meant as an attack on proprietary code and commercial software. The author is quite pragmatic when it comes to choosing software, but for this thesis it is an exciting experiment to only use open source software.

A main principle of open source software is that source code, design blueprints and documentation is freely available to the public. Development is often a collaborative effort where programmers improve the software and share their changes with the community. The open development process is a strength in open source software. It enables anyone to check the validity of the code and report or fix bugs. Many open source software packages are free to download and can be modified and published in new versions without restriction. However license terms of open source software vary and must therefore be studied carefully. (Open-source model 2017)

While open source software is usually free to download and use, it still requires a lot of effort to develop and maintain. One important contribution is that users report bugs, propose improvements or contribute with new code or functionality. Donations from satisfied users is also a common way of funding development. Another funding alternative is to sell premium versions of open source software with added functionality or support plans. In the end you may end up paying money for open source software, but you don’t have to.

Open source software exists for all operating systems, but many will argue that it is a lot easier to work with open source software on Linux based operating system. All the work in this thesis is done with the Linux based operating systems Ubuntu\(^1\) and Red Hat\(^2\).

In the work with this thesis a huge amount of open source software packages have been used. Everything from typesetting in \LaTeX{} to advanced image analysis in MicMac is open source and the software packages themselves are usually built on numerous other software packages and libraries. It is not realistic to list them all and therefore this software review will focus on the main contributing softwares in the data processing. Figure 16.1 shows the main software packages and how they relate to the processing steps. The main feature of this graph is to show how the software packages work together.

\(^1\)https://www.ubuntu.com/
\(^2\)https://www.redhat.com/
Figure 16.1 shows the role of every software. In general one can say that the software with the most arrows leading to it is the most important. By studying the graph one can see that MicMac and GDAL very often have direct connections to the processing steps. This illustrates their importance and ability to perform different tasks. The number of subprocesses in different tasks is not shown in this figure, but by examining the workflows of fiducial mark finding (figure 8.15), mask creation (chapter 10) and point cloud processing (figure 12.1) it becomes clear that OpenCV, PDAL and scikit-image play important roles as well. In the bottom of the graph we find the spider in the software web. Python is used to connect all the components together and create automated workflows. Every task is connected to Python in some way. Either through Python bindings or by
generating bash scripts that are used to execute commands in other software. Python has proven itself as a flexible and multi-purpose programming language.

Working with open source software has been a very rewarding experience. The huge number of libraries and software packages gives a very large degree of freedom and flexibility. However the freedom and flexibility comes at a price. While proprietary commercial software often offers a complete package with all the tools needed for the task, the open source approach will often require that the user combines the different tools to complete the task. There is also less focus on graphic user interface. Tools are often command line tools or libraries for some programming language, like Python. Using open source software will often require that the user has a basic understanding of programming or scripting to use it effectively. There are of course exceptions to this. QGIS is one example of a open source product that offers a very good graphic user interface. If the user is prepared to sacrifice the convenience of the all-in-one package with a polished graphic user interface for freedom and flexibility, open source software is a good choice.

Documentation is very important when working with programming languages and command line tools. It is impossible to guess syntax and the meaning of short argument abbreviations. The experience from working with this thesis is that the software is generally very well documented. Some are not, but they were discarded and not used. The documentation comes in many forms. Sometimes as web solutions like wikis or pdf-documents and sometimes as built-in commands available by typing `--help` in the terminal. Often the documentation is duplicated and available both as web pages and built-in commands. When the software implements algorithms described in the literature they are usually cited or even explained in detail in the documentation. If the documentation is not sufficient there are numerous forums with helpful community members that will help you along. This also applies if you discover a bug or an error in the software. Bug reporting is appreciated. On two occasions during the thesis work bugs were discovered and reported. On both occasion they were fixed and new versions of the software was released within 24 hours.

The overall experience with open source software is very good, but it will likely require a basic understanding of programming languages. Using Linux based operating systems will make it a lot easier as well.
17 Conclusion

Creating orthophotos and digital elevation models from high-oblique imagery is not a trivial task. The large angle between the vertical axis and the optical axis of the cameras is a challenge. Distorted pixels makes matching difficult and it reduces accuracy and precision. The old data standard and noisy quality of the Svalbard 1936 images adds additional complexity to the task.

The main task in this thesis was to derive an automatic workflow for creating orthophotos and digital elevation models from the 1936 and 1938 high-oblique aerial images of Svalbard. This must be considered completed. All steps except GCP collection and checking results are automated and produce good results. There are some accuracy issues but they are mainly caused by poor GCPs. On the other hand the precision of the output is very good given the quality of the input high-oblique images. Some noise and artifacts must be expected when using old data that is acquired from a suboptimal angle for creating orthophotos and digital elevation models. There are several things that can be improved to produce better results. With the limited amount of time available to complete this project all of these possible improvements have not been explored in depth.

Using well known methods to develop a workflow that can be used with other data was stated as an important part of the solution to the problem. It has never been the intention to develop a specific tool that will only work on one dataset, but rather present a set of interesting methods and try to apply them to a specific dataset. The workflow is specific to the Svalbard 1936 dataset, but with slight modification it can likely be applied to other datasets. The fiducial mark finder will need new training parameters and the masking algorithm will need new samples, while the point cloud cleaning and raster mosaicing procedures are likely to work without other modification than modifying parameters. All the software tools used are available on the internet and the basic methods applied are well documented in scientific literature. The image analysis methods used are well known in the computer vision community and have been developed and used for many years.

Completing the task with only open source software was also an important part of the solution to the problem. A large collection of software packages were used to build a robust workflow with the programming language Python as the backbone. It is important to state that choosing to use only open source software is an experiment and not a
statement. The author has a firm belief in a pragmatic approach when it comes to choosing software. However, it has been a very positive experience to only use open source software.

The methods presented in this thesis are already in use for processing data in geosciences. Recent development in technology and processing power make these methods more easily available and the use of computer vision methods like SfM in geosciences is increasing (Westoby et al. 2012, Fonstad et al. 2013). Robust and efficient methods for processing data will be important in the future. Both for processing newly acquired data as well as extracting additional information from data collected in the past.


Couderette, Frédérique (2016). 1936 oblique aerial images : Spreading the time series of Svalbard glaciers. Internship Report. ECOLE NATIONALE DES SCIENCES GEOGRAPHIQUES.

Crandall, David J. et al. (2013). “SfM with MRFs: Discrete-Continuous Optimization for Large-Scale Structure from Motion”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 35.12, pp. 2841–2853. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2012.218.


Luc Girod (2016). *Position of fiducial marks in Svalbard 1936 image negatives*. IV


“Norwegian Expedition to Svalbard, 1938” (1939). In: Polar Record 3.17, p. 5. ISSN: 0032-2474, 1475-3057. DOI: 10.1017/S003224740003802X.

— (n.d.[b]). NP_Satellitt_Svalbard_WMTS_25833. url: https://data.npolar.no/dataset/71528149-b8e0-4646-9e5d-f7202acae6e.


