Registration of 3-D laser radar data and hyperspectral imagery for target detection

KEVIN CHAN

FOI is an assignment-based authority under the Ministry of Defence. The core activities are research, method and technology development, as well as studies for the use of defence and security. The organization employs around 1350 people of whom around 950 are researchers. This makes FOI the largest research institute in Sweden. FOI provides its customers with leading expertise in a large number of fields such as security-policy studies and analyses in defence and security, assessment of different types of threats, systems for control and management of crises, protection against and management of hazardous substances, IT-security and the potential of new sensors.

FOI
Defence Research Agency
Sensor Technology
P.O. Box 1165
SE-581 11 LINKÖPING

Phone: +46 13 37 80 00
Fax: +46 13 37 81 00

www.foi.se

FOI-R--2101--SE
ISSN 1650-1942

Technical report
November 2006

Sensor Technology

Kevin Chan

Registration of 3-D laser radar data and hyperspectral imagery for target detection

FOI-R--2101--SE Technical report **Sensor Technology**
ISSN-1650-1942 November 2006

Issuing organisation FOI – Swedish Defence Research Agency Sensor Technology P.O. Box 1165 SE-581 11 LINKÖPING	**Report number, ISRN** FOI-R--2101--SE	**Report type** Technical report
	Research area code C^4ISTAR	
	Month year November 2006	**Project no.** E3080, E3082
	Sub area code Above water Surveillance, Target Acquisition and Reconnaissance	
	Sub area code 2	
Author/s (editor/s) Kevin Chan	**Project manager** Tomas Chevalier, Jörgen Ahlberg	
	Approved by Mattias Severin	
	Sponsoring agency Swedish Armed Forces	
	Scientifically and technically responsible Jörgen Ahlberg	

Report title
Registration of 3-D laser radar data and hyperspectral imagery for target detection

Abstract
The purpose of this master's thesis is to combine sensors for target detection and recognition, the sensors in this case being a hyperspectral camera and a scanning 3-D laser.

Registration of the sensor data is essential in order to combine the sensors. The registration is performed by geometrical calibration of the sensors with repsect to each other. A problem with calibrating the sensors is that the scanning 3-D laser delivers 3-D point clouds and the hyperspectral camera returns 2-D images. Also, there is a wavelength problem. The hyperspectral camera has 240 different bands at its disposal, whereas the laser only uses one.

This thesis is a part of a two-man project, where the other part is done as a master's thesis by Christina Freyhult. Her part is to implement the anomaly detector and the visualization of the results.

We have designed and implemented a method for automatic registration of 3D-laser data and hyperspectral imagery. We have thus shown that this kind of fusion is possible to realize, and we judge the potential to be tremendously high.

Keywords
laser radar, hyperspectral, target detection, sensor fusion, registration

Further bibliographic information	**Language** English
ISSN ISSN-1650-1942	**Pages** 54 p.
	Price acc. to pricelist

Utgivare FOI – Totalförsvarets forskningsinstitut Sensorteknik Box 1165 581 11 LINKÖPING	**Rapportnummer, ISRN** FOI-R--2101--SE	**Klassificering** Teknisk rapport
	Forskningsområde Ledning, informationsteknik och sensorer	
	Månad år November 2006	**Projektnummer** E3080, E3082
	Delområde Spaningssensorer	
	Delområde 2	
Författare/redaktör Kevin Chan	**Projektledare** Tomas Chevalier, Jörgen Ahlberg	
	Godkänd av Mattias Severin	
	Uppdragsgivare/kundbeteckning Försvarsmakten	
	Tekniskt och/eller vetenskapligt ansvarig Jörgen Ahlberg	

Rapportens titel
Registrering av 3-D laserradardata och hyperspektrala bilder för måldetektion

Sammanfattning

Syftet med detta examensarbete är att kombinera sensorer för måldetektion och igenkänning, där sensorerna vi har till förfogande är en hyperspektral kamera samt en 3D-laser.

Att kunna registrera sensordata är en grundläggande förutsättning för att kunna kombinera sensorerna. Registrering utförs genom att geometriskt kalibrera sensorerna gentemot varandra. Ett problem med denna kalibrering är att 3D-lasern levererar 3D-punktmoln medan den hyperspektrala kameran levererar 2D-bilder. Dessutom finns ett våglängdsproblem. Den hyperspektrala sensorn har 240 olika våglängdsband till sitt förfogande medan laser bara använder ett.

Detta arbete är en del i ett tvåmansprojekt där den andra delen är gjord som en examensarbete av Christina Freyhult. Hennes del är att implementera anomalidetektorn och att visualisera resultaten.

Vi har designat och implementerat en metod för automatisk registrering av 3D-laserdata och hyperspektrala bilder. Vi har därmed visat att det är möjligt att realisera denna type av datafusion, och vi bedömer potentialen som mycket stor.

Nyckelord
laserradar, hyperspektral, måldetektion, sensorfusion, registrering

Övriga bibliografiska uppgifter	**Språk** Engelska
ISSN ISSN-1650-1942	**Antal sidor:** 54 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Contents

Preface

This thesis will finalise a Master of Science degree in Electrical Engineering at the Institute of Technology, Lund University. The thesis was done at the Swedish Defence Research Agency (FOI), division of Sensor Technology. Supervisors at FOI have been Tomas Chevalier and Jörgen Ahlberg.

I would first like to thank Tomas Chevalier for his outstanding support and supervision. Without his help I do not believe I would be able to finish this thesis.

I also would like to address my appreciation to Jörgen Ahlberg. Discussions with him have really inspired me with ideas and solutions to problems I faced. His knowledge in this area is outstanding and I have learned a lot by talking and discussing problems with him. Also, big thanks to Erik Thorin whom I shared the office with. He has during this thesis helped my straighten out some minor problems I faced and discussing possible solutions via different approaches. Also, the table tennis matches we played throughout this period have helped me clear my head and find alternative approaches to my problems. I also want to thank Ove Svensson, Linda Nordin, and Christina Freyhult for great discussions throughout this period.

1 Introduction

1.1 Background

The Swedish Armed Forces use a variety of sensors for purposes of surveillance and reconnaissance. Most of the time, a certain sensor will only give information regarding a particular parameter (e.g., IR-sensors detect heat radiation) and fail in finding other parameters (e.g., ordinary broadband IR-sensors cannot return any spectral information).

Using a hyperspectral camera in cooperation with a scanning 3-D laser to detect anomalies is a new concept. The idea behind this is to make use of the many spectral bands the hyperspectral camera posess. Since it has so many bands (about 240), it has the capability to detect objects that, in its spectral signature, deviate from the enviroment. After finding the region where a possible anomaly is, we use the scanning 3-D laser to shoot at the object to find out what kind of object it is, by constructing a point cloud.

The detection algorithm is designed by our supervisor Jörgen Ahlberg and is implemented by my partner, Christina Freyhult. My part of this project is to implement a system that will allow usage of data from both sensors. In order to do so, we ahve to register the sensor data, i.e., both sensors mustbe geometrically calibrated to each other.

1.2 Purpose

The purpose of this thesis is to develop and implement methods to estimate the geometric relation between a hyperspectral camera and a scanning 3-D laser radar looking at the same scene. The hyperspectral camera scans an area and detects objects that in its spectral signature deviate from the neighbouring enviroment, also called anomaly detection. The laser, on the other hand, is used for detailed analysis of the detected objects. The anomaly detection and the 3-D visualisation is described in a parallell thesis [ref till Christinas rapport].

This is the first step in interaction of different kind of sensors for detection purposes. In this thesis, we are going to show the strength and benefits of using interacting sensors and also where the potential problems can occur.

1.3 Problems

While different sensors give different data about a given scene, the result of their sum will be hard to interpret or use as long as the two data sets are not registered. It is possible to do that manually, but it takes a lot of time and effort. This is what the thesis will try to resolve by creating a program that will automatically calibrate the information and then use the data to extract further intelligence.

Also, the two sensors treated in this report measures in completely different wavelengths. As mentioned earlier, the hyperspectral sensor has 240 spectral bands, each band uses one dedicated wavelength, whereas the laser uses one

wavelength only. The problem that could occur is that an object seen by a sensor using a particular wavelength might not have the same apperance viewed by the other sensor, since it is using a totally different wavelength/wavelengths. Suppose that an object have the color transistion 'black/white/black'. One of the sensors might register the transistion as it is, but the other one might register it as 'white/black/white'.

A third problem is that the hyperspectral camera delivers a 2-D image whereas the laser returns a 3-D point cloud. Performing matching of these completely different data structures will, of course, be hard. What consequences will these aspects bring to our results and how can these problems be solved?

2 Sensors and Data

2.1 Data acquisition

During a two-day period, a number of FOI staff-member conducted a field study, collecting necessary data for both the hyperspectral sensor and the scanning 3-D laser required to perform the analysis of the sensors. The collected data would be a basis on which to test programs and different proposed methods.

2.1.1 Location

FOI has access to an area near Ströplahult called P4/Kvarn, which is a military facility for training purposes. The data aquisition was conducted in cooperation with the Army Combat School (MSS). The sensors were placed next to each other, facing a small hill composed of some forest area and open terrain, depicted in figure 2.1. The varying terrain will give our acquired data more versatility and consistency.

Figure 2.1: Location where the data acquisition was conducted

2.1.2 Targets

For the purpose of the study several vehicles and mines were lent to FOI by the MSS to be placed in different locations. The vehicles were Volvo C303, Pbv 401 (armored personnel carrier) and T72 (main battle tank) and the mines were of type AT-47b, TMM-1, TMRP-6, and TMA-1. The data collection was conducted such that different scenarios were recorded using both sensors with variations in the placement of the targets, see [22]. These displacements of different targets were registered for further reference. Apart from the targets that were to be detected in our work, data was collected of some checkerboards in the scene for sensor calibration purposes. Figure 2.2 depicts some of the targets that were placed in different positions in the scene.

Figure 2.2: Targets that were to be found during the survey

2.2 Sensors

The two sensors were placed as close to each other as possible, at approximately the same height and facing the same direction so that the data would have a maximum overlap. The sensors and the data they produce are described in more detail in the following section. In figure 2.3, the left-hand side sensor is the hyperspectral sensor and the right-hand side the scanning 3-D laser ILRIS-3D.

Figure 2.3: Sensors used in this thesis

2.2.1 Hyperspectral sensor: ImSpec

ImSpec, from the Finnish company Specim, is a sensor that is used to register hyperspectral images of the incoming light in the visual and near infrared region. The name hyperspectral spawns from the fact it registers in hundreds of narrow wavelength bands, spanning from 391 to 961 nm. This can be compared to a consumer camera that registers only three bands of wavelengths that correspond to red, green, and blue color.

The components of the ImSpec sensor can be decomposed into three parts. At the back of the sensor a CCD-array is placed for the registration of the data. In front of the CCD-array is the 'Imspector', a crystal that divides the incoming light into different wavelengths. At the front is the scanning mirror which is necessary for the registration of the entire image and not just a single line. The CCD-array registers the incoming light over all the spectral components, over one line, and the mirror scans the entire area to make an entire image. The sensor has accompanying software allowing changes in its performance and settings.

2.2.2 Scanning 3-D laser: ILRIS-3D

To collect the 3-D space data the ILRIS-3D sensor from Optech was used. This sensor is a scanning 3-D laser radar and is composed of a laser, a detector, and controllable mirrors.

To compute the distance from the sensor to the target hit by the laser, the time of flight is used together with the known speed of the laser pulse. The scanner scans from the lower left corner and scans row-wise left to right followed by moving up one one row and starting the scanning all over again. The field of view of the sensor is about $40 \times 40°$ and the divergence between the shots is 170 μrad. With a minimum angle increment of $0.00015°$ the maximum number of points is approximately 0.7 billion. The scanning speed is 2000 points per second making the total scan time at most approximetely 100 hours. However, most of the time it is sufficient to collect around a million points, making the effictive scan time $8 - 10$ minutes.

The raw accuracy is ± 10 mm for every 3-D space direction (X, Y, Z) based on a target in 100m range. These values are within the range of one standard deviation, $\pm\sigma$, in a Gaussian distributed system corresponding to approximately 68% of the measured points. The maximum 3-D space resolution is approximately 1.3 mm at a distance of 50 m and around 2.6 mm at a distance of 100 m.

With precision control over the horizontal and vertical positions of the laser shot, a 3-D point cloud is created. The coordinates are given as the 3-D space coordinates (X, Y, Z) with an additional variable I that represents the intensity of the reflected pulse.

Two general methods are used by the system, *first* and *last echo*. This means that the position recorded in the sensor is that of first or the last thing the laser hits, exceeding a certain amplitude limit. To be able to penetrate the forest the equipment needed to use *last echo*. More information concerning the laser can be found in [17].

2.3 Data

The data acquired from the ImSpec and ILRIS-3D sensors are briefly described in this section.

2.3.1 Hyperspectral

The data collected from the ImSpec sensor contains information about the spectral signature of the image.

The data collected is arranged in a 3-D matrix. Each 'layer' is comprised of 512×197 pixels and the dataset is 240 bands deep. Each single band contains the information for the wavelengths recorded for the corresponding pixel. Because the frequency information ranges from 396 to 961 nm, each band represents approximately 2.4 nm. The field of vision of the sensor is about $20 \times 20°$, so a pixel registered with default settings represents $0.056°$.

Additional information concerning the hyperspectral camera and its data can be found in [25].

2.3.2 3-D laser

The data collected from the ILRIS sensor is called a point cloud and contains the positions and intensities of the points hit by the laser. The positions are arranged in a 3-D coordinate system, given by either the 3-D space coordinates $[\,X\,Y\,Z\,]$ or the polar coordinates [Azimuth, Elevation, Range]. The intensity

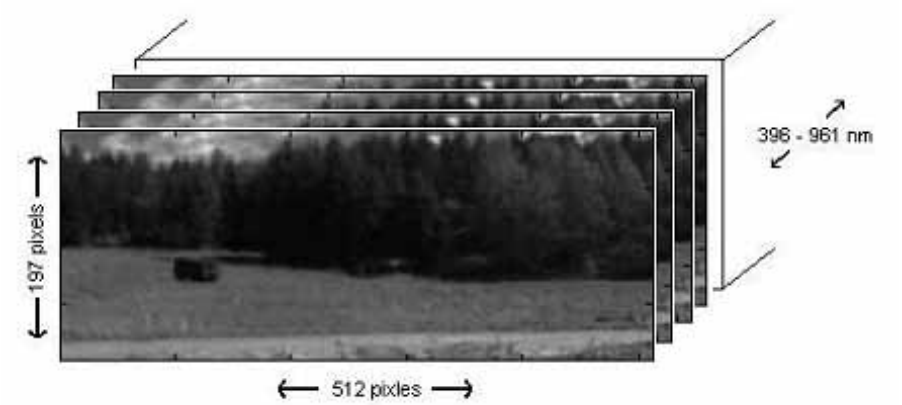

Figure 2.4: Image of scene 'car' taken with the hyperspectral sensor with its different bands

value is given as a scalar from $0 - 255$, independent of the coordinate system used. These four parameters are stored in four columns for later analysis.

Figure 2.5: Corresponding point cloud to scene 'car'

Many areas will have 'information shadows' where solid objects block the view of objects laying behind them and so there will not be any information in this area.

2.3.3 Software for acquired data

After collecting the data from both sensors, some post-processing must be done in order to perform the analysis of it. For raw data, the software used was a Matlab toolbox developed by Jörgen Ahlberg [1], and for 3-D point cloud, also a Matlab toolbox developed by Tomas Chevalier [18] and Polyworks [24] is used for viewing and inspection of the point cloud.

For every picture taken with the hyperspectral camera, we get a raw data file, a header, and the corresponding log-file. The Matlab toolbox used for this simply takes the raw data file as its input to output a picture with 190 spectral bands as default. This value can simply be changed in the camera picture software used. This picture is used for further analysis.

With the laser, a `xyz`-file is returned. This file is input to the point cloud toolbox, which has several features such as getting space coordinates of point cloud, angles etc. The output of this toolbox is a 3-D point cloud for analysis purpose. If only inspections of the point cloud is necessary, Polyworks can generate the point cloud from the `xyz`-file for this purpose. However, no heavy analysis can be done with Polyworks.

3 Theory

3.1 Camera projection and calibration

A corner stone of this thesis is to map 3-D points to pixels, and then go back from pixels to 3-D points. Imagine that a picture is taken with the hyperspectral camera and an anomaly is detected. The anomaly will only appear as a region with different color in the picture. In order to see what it is, the 3-D laser needs to scan the region of interest to form the 3-D point cloud. Hence, every pixel will have its corresponding 3-D space coordinates stored, i.e., every pixel will generate a set of 3-D space coordinates. How big this set is for every pixel depends on for example how far the object is to the camera, object density etc. To store the 3-D space coordinates for every pixel may seem very unnecessary and excessive waste of memory storage space. However, this is not as redundant as it seems. Suppose that an object is detected somewhere in the open terrain. The depth information about the object will be lost when the picture is taken of the object. Hence, finding the object from the picture in the 3-D space will result in the *scaling ambiguity* phenomena. Scaling ambiguity can be illustrated by holding a pen in front of a door. Placing the pen closer to the eye will eventually make the pen bigger than the door. With this in mind, it is essential to find a method to retrieve the depth information lost at the moment when the picture was taken.

There is a number of different methods to reconstruct 3-D objects from pictures. Most of these involve stereo vision. However, in our case we only have one camera (hyperspectral) at our disposal, thus, this method is not of any use. Instead, by *calibrating* the camera to find out how every 3-D point will be mapped into the picture, we can do the reverse and find out which 3-D points correspond to an arbitrary pixel on the picture, i.e., for every pixel a *tunnel with 3-D space coordinates* in the real 3-D space will be returned.

A lot of work have been done in camera calibration. For example, [3] [4] [14] [5] suggest a number of ways to perform a camera calibration.

3.1.1 Camera geometry

A camera can at its most simple form be modelled with a circular aperture with a diameter D. A lens refracts all rays from a single point source to a point in the image plane. If the lens is considerably thin, then the assumption that the lens only refracts rays is valid. Gauss' lens law states that:

$$\frac{1}{a} + \frac{1}{b} = \frac{1}{f}$$

where a is the object's distance to the lens, b is the distance between the image plane and the lens, and f the focal length. For a point source very far away, we do the assumption $b \approx f$. This is the principle of the *pinhole camera*. A pinhole camera is basically a box with a small lens for light refraction and a film to collect the refracted light.

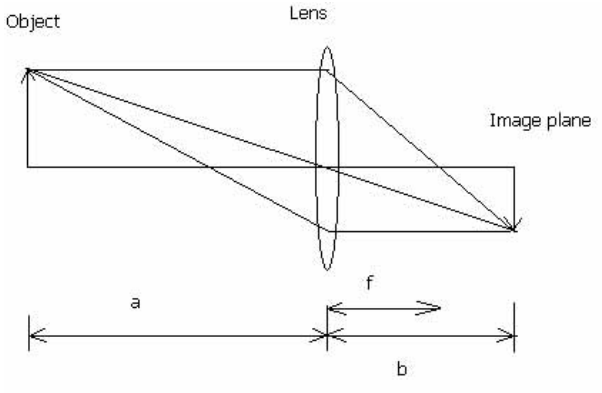

Figure 3.1: Illustration of Gauss' lens law

3.1.2 Perspective projection

Before we examine camera calibration, we must understand how the mapping between the real world and image is done. Consider figure 3.2. This type of projection is better known as *perspective projection*, which conserves angles and directions unlike *affine projection*, which let straight lines remain straight no matter what the distance to the line is. Hence, perspective projection conserves distance information by projecting distant object smaller. The key to this is to let all projection lines intersect in the origin $(0, 0, 0)$ of the 3-D coordinate system and assume that the image plane intersects the Z-axis at the point $(0, 0, 1)$. Using triangles, it can be seen that the relationship between 3-D and image coordinates is given by:

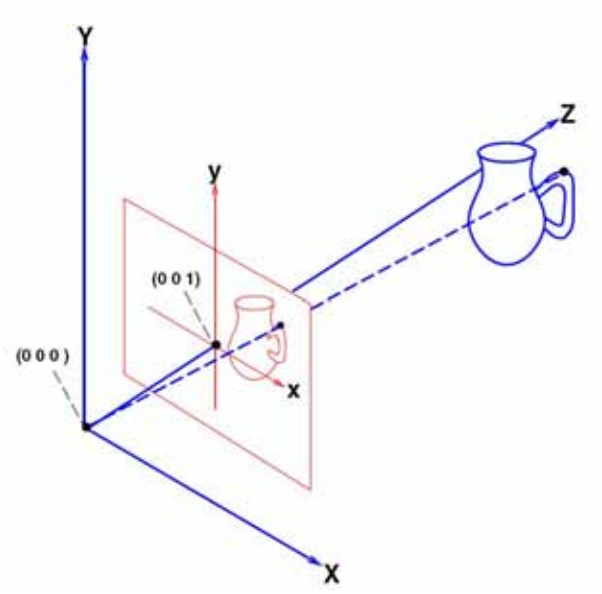

Figure 3.2: Perspective projection [3]

$$x_i = \frac{X_i}{Z_i} \tag{3.1}$$

$$y_i = \frac{Y_i}{Z_i}$$

These two equation can be combined into a vector equation in order to simplify further analysis:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{3.2}$$

where $\lambda = 1/Z_i$ is a scale factor for this particular case. This vector equation is called *homogeneous coordinates* and is a linear combination and it is only valid when the image plane intersects the Z-axis of the 3-D coordinate system at $(0, 0, 1)$. If the intersection is at an arbitrary point along the Z-axis, (3.1) becomes:

$$\hat{x}_i = \frac{X_i}{Z_i} f \tag{3.3}$$

$$\hat{y}_i = \frac{Y_i}{Z_i} f$$

where $x = \hat{x}/f$, $y = \hat{y}/f$, and f is the *focal length*, which defines the distance between the image plane to origin of the 3-D space system. Thus, (3.2) becomes:

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \lambda \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{3.4}$$

where $\lambda = f/Z_i$.

As (3.4) indicates, this describes a very general projection. The 3-D space coordinate will be projected on the image plane, ***not*** on the image itself when $f \neq 1$. From now on, the image plane coordinates will be denoted with a hat, i.e. $(\hat{x}, \hat{y})^T$ and the pixel coordinates with $(x, y)^T$.

3.1.3 Camera rotation and translation

Suppose a picture is taken from a particular direction. Moving the looking direction means that a rotation has been introduced to the system, i.e. the camera orientation with respect to a given world frame is subject to a change. In order to perform further analysis, compensation for this rotation must be accounted for. Let $[\hat{x}'\, \hat{y}']$ and $[X'\, Y'\, Z']$ denote the image frame coordinates and the 3-D space coordinates of the rotated system, respectively. Using the same concept of perspective projection, the rotated image frame coordinates are projected as:

$$\hat{x}'_i = X'_i/Z'_i \tag{3.5}$$

$$\hat{y}'_i = Y'_i/Z'_i$$

which has an equivalent vector representation:

$$\begin{bmatrix} \hat{x}'_i \\ \hat{y}'_i \\ f \end{bmatrix} = \lambda' \begin{bmatrix} X'_i \\ Y'_i \\ Z'_i \end{bmatrix} \tag{3.6}$$

where $\lambda' = f/Z'_i$ also is a scaling factor. Now, consider figure 3.3, where the system is rotated about the Y-axis with an angle θ. Using simple triangle

expressions, it is possible to obtain a matrix describing the rotation about the Y-axis, given by:

$$\begin{bmatrix} X_i' \\ Y_i' \\ Z_i' \end{bmatrix} = \mathbf{R}_Y \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{3.7}$$

$$\Leftrightarrow$$

$$\begin{bmatrix} X_i' \\ Y_i' \\ Z_i' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}$$

Plugging this equation into (3.6) yields

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \lambda' \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{3.8}$$

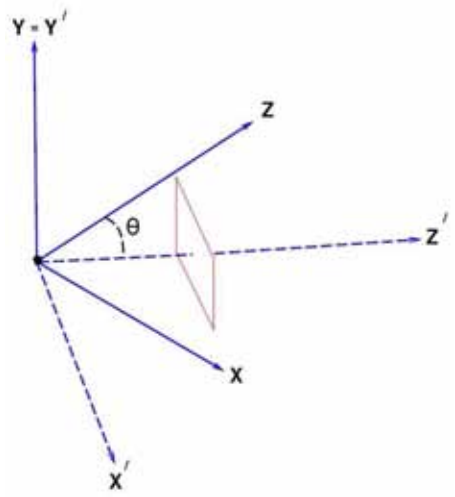

Figure 3.3: Camera rotation about the Y-axis with angle θ [3]

Doing the same analysis for the X and Z-axis, expressions for the rotation matrices describing rotations about the X and Z-axis will be given by:

$$\mathbf{R}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \tag{3.9}$$

$$\mathbf{R}_Z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

where (φ, ψ) denotes rotation about X and Z-axis, respectively. By combining all three rotation matrices in a matrix multiplication, expression for any rotation is given by

$$\mathbf{R} = \mathbf{R}_X \mathbf{R}_Y \mathbf{R}_Z \tag{3.11}$$

which also is a 3×3 matrix given by

$$\begin{bmatrix} \cos\theta\cos\psi + \sin\theta\sin\varphi\sin\psi & \cos\theta\sin\psi - \sin\theta\sin\varphi\cos\psi & \sin\theta\cos\varphi \\ -\cos\varphi\sin\psi & \cos\varphi\cos\psi & \sin\varphi \\ -\sin\theta\cos\psi + \cos\theta\sin\varphi\sin\psi & -\sin\theta\sin\psi - \cos\theta\sin\varphi\cos\psi & \cos\theta\cos\varphi \end{bmatrix}$$

If the camera's position is changed during the photo shoot, a *translation* of the camera's position has been performed. The translation of the position can be seen as a position offset of the camera's 3-D space coordinates, i.e. if the translation is (X_0, Y_0, Z_0), then (3.2) becomes

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \lambda \begin{bmatrix} X_i - X_0 \\ Y_i - Y_0 \\ Z_i - Z_0 \end{bmatrix} \tag{3.12}$$

If rotation and translation are to be imposed on the camera, then the image plane coordinate will be mapped as

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \lambda \mathbf{R} \begin{bmatrix} X_i - X_0 \\ Y_i - Y_0 \\ Z_i - Z_0 \end{bmatrix} \tag{3.13}$$

for any arbitrary rotation and translation.

3.1.4 Camera parameters

As probably most know, different cameras behave differently regarding the resulting photo. The reason is that the cameras have different camera parameters. These parameters are usually divided into two groups: *internal* and *external* parameters. The external parameters describe the displacement and the rotation of the camera with respect to a reference camera. This reference camera could be the camera with which the first picture was taken. The *internal parameters* describe the 'inner mechanics' of the camera. At its simpler form, it can be expressed by the following matrix:

$$\mathbf{\Phi} = \begin{bmatrix} \sigma_x f & \gamma & x_0 \\ 0 & \sigma_y f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

The internal camera parameters include the effective focal length, f, which is the distance between the image plane and the projection center, scale factors (σ_x, σ_y), and the image center (x_0, y_0) also known as the *principal point*. The general properties of these parameters are listed below.

- The scale factors (σ_x, σ_y) must be accounted for in order to relate distance of the 3-D coordinate system to image coordinate system
  - Objects farther away will yield smaller pixel size
- The origin of an image is usually in the upper left corner of the image array. Since the principal point does not necessarily coincide with the origin of the image coordinate system, (x_0, y_0) are usually denoted as the image center.
- The zooming of a camera is controlled by varying the focal length, f.
- If the image coordinate system is not orthogonal, i.e. the pixels are not square, then we must introduce a *skew factor*, γ, to account for this.

Using this matrix, a new mapping can be formed as

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \mathbf{\Phi}\, \lambda \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{3.15}$$

Worth noting is that this mapping is only valid if the camera is placed at the same place, looking in the same direction. A valid mapping with an arbitrary rotation and translation can be obtained by imposing rotation and translation constraints introduced in section 3.1.3, yielding the following mapping:

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \mathbf{\Phi}\,\mathbf{R}\,\lambda \begin{bmatrix} X_i - X_0 \\ Y_i - Y_0 \\ Z_i - Z_0 \end{bmatrix} \tag{3.16}$$

However, it is most inconvenient to have the translation in the same vector as the 3-D space coordinates. Seperating them from each other will simplify the analysis tremendously. It can be seen by inspection that the expression

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \mathbf{\Phi}\,\mathbf{R}\,\lambda \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3.17}$$

is equivalent with (3.16). Using the following simplified notation, (3.17) can be rewritten to

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \mathbf{\Phi}\,\mathbf{R}\,\lambda\,[\mathbf{I} \mid -\mathbf{X}_0] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

where $\mathbf{I}$ and $-\mathbf{X}_0$ are the identity matrix and translation vector, respectively. Further analysis yields

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \mathbf{\Phi}\,\lambda\,[\mathbf{R}\,\mathbf{I} \mid \underbrace{-\mathbf{R}\,\mathbf{X}_0}_{\mathbf{T}}] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \underbrace{\mathbf{\Phi}\,\lambda\,[\mathbf{R} \mid \mathbf{T}]}_{\mathbf{P}} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3.18}$$

$$\begin{bmatrix} \hat{\mathbf{x}}_i \\ f \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X}_i \\ 1 \end{bmatrix} \tag{3.19}$$

where $\mathbf{T}$ is the vector containing a rotated translation, $\hat{\mathbf{x}}_i = [\hat{x}_i\ \hat{y}_i]^T$ is the vector containing image plane coordinates, $\mathbf{X}_i = [X_i\ Y_i\ Z_i]^T$ is the vector with the 3-D space coordinates, and $\mathbf{P}$ is the *projection matrix*, which we will discuss more in next section. This result is a cornerstone in camera calibration: knowing $\mathbf{P}$ means that we are able to find an outstanding mapping between the 3-D space to the image.

3.1.5 Calibration: Linear Method

Camera calibration is actually an image post processing tool which allows the picture to be adjusted more correctly to the object it is looking at. The main idea behind camera calibration is to find a mapping between the 3-D space and the image without any distortions or other artefacts. A typical example of this is to remove lens distortion[1]. Lens distortion will add 'round edges' on

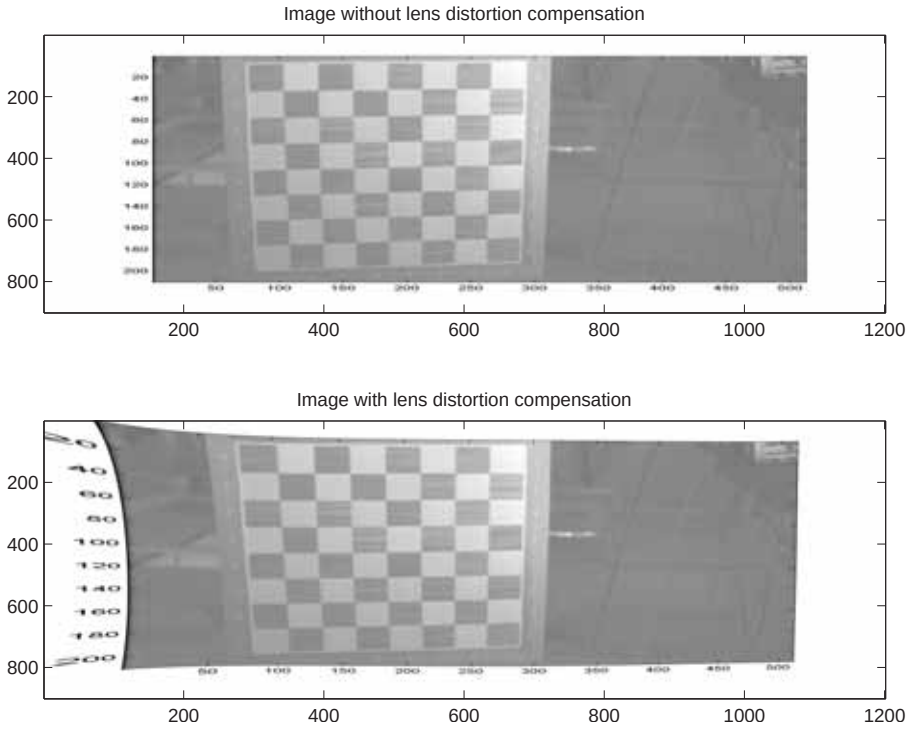

Figure 3.4: Example of lens distortion in the hyperspectral sensor

the picture, making straight lines curvy and thereby letting more information being visulized then what the camera is supposed to.

As seen on the figure 3.4, the image edge on the left side is seriously curved out to compensate for the distortion with respect to the checkerboard.

Calibration of a camera is equivalent to finding the 3-D to 2-D mapping, $[\hat{\mathbf{x}}_i \; f]^T = \mathbf{P}\,[\mathbf{X}_i \; 1]^T$, given by:

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ f \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3.20}$$

Hence, each image frame coordinate is given by:

$$\begin{aligned} \hat{x}_i &= X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14} \\ \hat{y}_i &= X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24} \\ f &= X_i p_{31} + Y_i p_{32} + Z_i p_{33} + p_{34} \end{aligned} \tag{3.21}$$

Recall from (3.3), the image pixel coordinates are given by:

$$\begin{aligned} x_i &= \frac{\hat{x}_i}{f} \\ y_i &= \frac{\hat{y}_i}{f} \end{aligned} \tag{3.22}$$

[1]Removing lens distortion cannot be done without knowing the mapping between the 3-D space system and the image coordinate system. To remove the distortion, non-linear optimization techniques need to be called for. The objective function to optimize is an odd power polynomial of infinite length.

From (3.22) and (3.21), it is easy to verify that the pixel coordinate can be written as

$$x_i = \frac{X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + p_{34}} \tag{3.23}$$

$$y_i = \frac{X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + p_{34}}$$

This implies that

$$0 = X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14} - x_i X_i p_{31} - x_i Y_i p_{32} - x_i Z_i p_{33} - x_i p_{34}$$

$$0 = X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24} - y_i X_i p_{31} - y_i Y_i p_{32} - y_i Z_i p_{33} - y_i p_{34}$$

Now, consider the vector

$$\mathbf{p} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{21} & p_{22} & p_{23} & p_{24} & p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}^T$$

By measuring n 3-D points with corresponding pixel coordinates, form the following matrix equation:

$$\mathbf{D} \cdot \mathbf{p} = \mathbf{0} \tag{3.24}$$

$$\Leftrightarrow$$

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ \vdots \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

where $\mathbf{0}$ is a $2n$-vector with zeros. Hence, the objective is to find the non-trivial solution, since the trivial solution $\mathbf{p} = \mathbf{0}$ is physically insignificant. To obtain the non-trivial solution, constrained optimization technique must be called for. The objective function to minimize is given by:

$$\min_{\mathbf{p}} ||\,\mathbf{D}\,\mathbf{p}\,||^2 \qquad \text{subject to} \quad ||\mathbf{p}||^2 = 1 \tag{3.25}$$

Let $\lambda > 0$ be the Lagrange multiplier. The Lagrangian function to be minimized is

$$L(\mathbf{p}, \lambda) = ||\,\mathbf{D}\,\mathbf{p}\,||^2 - \lambda\,(\,||\,\mathbf{p}\,||^2 - 1) \tag{3.26}$$

$$L(\mathbf{p}, \lambda) = (\mathbf{D}\,\mathbf{p})^T(\mathbf{D}\,\mathbf{p}) - \lambda\,(\mathbf{p}^T\mathbf{p} - 1)$$

Performing the differentiation on L with respect to $\mathbf{p}$ and setting to 0 gives

$$\frac{\partial L}{\partial\,\mathbf{p}} = \mathbf{D}^T\mathbf{D}\,\mathbf{p} - \lambda\mathbf{p} = 0$$

$$\mathbf{D}^T\mathbf{D}\,\mathbf{p} = \lambda\mathbf{p} \tag{3.27}$$

Differentiating L with respect to λ and setting to 0 yields

$$\frac{\partial L}{\partial \lambda} \;=\; \mathbf{p}^T\mathbf{p} - 1 \;=\; 0$$

$$\mathbf{p}^T\mathbf{p} \;=\; 1 \tag{3.28}$$

Pre-multipliying both sides of (3.27) by $\mathbf{p}^T$ gives

$$\mathbf{p}^T\mathbf{D}^T\mathbf{D}\mathbf{p} \;=\; \lambda\mathbf{p}^T\mathbf{p}$$

$$\Leftrightarrow$$

$$(\mathbf{D}\mathbf{p})^T(\mathbf{D}\mathbf{p}) \;=\; \lambda \cdot 1$$

$$||\,\mathbf{D}\mathbf{q}\,||^2 \;=\; \lambda \tag{3.29}$$

This equation states that minimizing $||\,\mathbf{D}\mathbf{q}\,||^2$ is equivalent with minimizing λ. Now equation (3.27) tells us that $\mathbf{p}$ should be an eigenvector of the matrix $\mathbf{D}^T\mathbf{D}$ with λ being the corresponding eigenvalue. Equation (3.29), on the other hand, says that we should minimize λ as much as possible. These two pieces of information together simply state that $\mathbf{p}$ should be the eigenvector that corresponds to the smallest eigenvalue of $\mathbf{D}^T\mathbf{D}$. Hence, to solve for $\mathbf{p}$ we can simply make use of the eigen-decomposition of $\mathbf{D}^T\mathbf{D}$.

The eigen-decomposition $\mathbf{D}^T\mathbf{D}$ will lead to a closed form solution. However, what is of more interest is the question of the rank of the matrix $\mathbf{D}$, since it reinforces the understanding of how the reference points should be chosen. Let's reconsider (3.24). Suppose that we have found the non-trivial null vector $\mathbf{p}$ of the matrix $\mathbf{D}$. From standard linear algbra, given a $n \times m$ matrix $\mathbf{D}$ then

$$\text{rank}\,(\,\mathbf{D}\,) + \text{null}\,(\,\mathbf{D}\,) \;=\; m$$

where null($\mathbf{D}$) represents the dimension of the null space of $\mathbf{D}$. In our case $n \geq m = 12$ and there are three cases to consider:

- rank($\mathbf{D}$) = 12. The null space has dimension 0, and there is only one solution to the system, namely $\mathbf{p} = 0$, which is not very meaningful.
- rank($\mathbf{D}$) = 11. The null space has dimension 1 and there is a unique solution (up to a scale factor).
- rank($\mathbf{D}$) < 11. The null space has dimension 2 or more. The null vector $\mathbf{p}$ we are seeking for can be any vector in this 2-dimensional space, which means that there is an infinite number of solutions to (3.27). One way in which this can happen is if all the 3-D space reference points are in a plane.

Note that the rank of $\mathbf{D}$ is often 12 rather than 11 in calibration of real data - noise inflates the rank of the matrix. When noise is present in our data points, the end result is that the smallest eigenvalue of $\mathbf{D}^T\mathbf{D}$ is not zero but a small positive number, since $\mathbf{D}^T\mathbf{D}$ is positive definite and symmetric. We can often pinpoint how much noise there is in our data (i.e., the world and image coordinates of the reference points) by inspecting the ratio between the smallest and the largest eigenvalues of the data matrix $\mathbf{D}^T\mathbf{D}$. If the rank of $\mathbf{D}$ is less than 11 means that there are two of more null vectors that are approximately equally good and the best solution in least square sense is being used.

The quality of $\mathbf{P}$ is determined by two parameters:

- how good the point correspondences are matched

  - the deviation between 3-D space coordinate and pixel coordinate should be as small as possible

- how well the point correspondences are distributed

The first point is straightforward. The other point, however, might require a bit of explaination. What actually is done here is a straight line fitting to a number of points, i.e. the 3-D space coordinates are optimized with respect to the distance in meter, which is depicted in figure 3.5 where the objective is to minimize all e_i, $i = 1...n$.

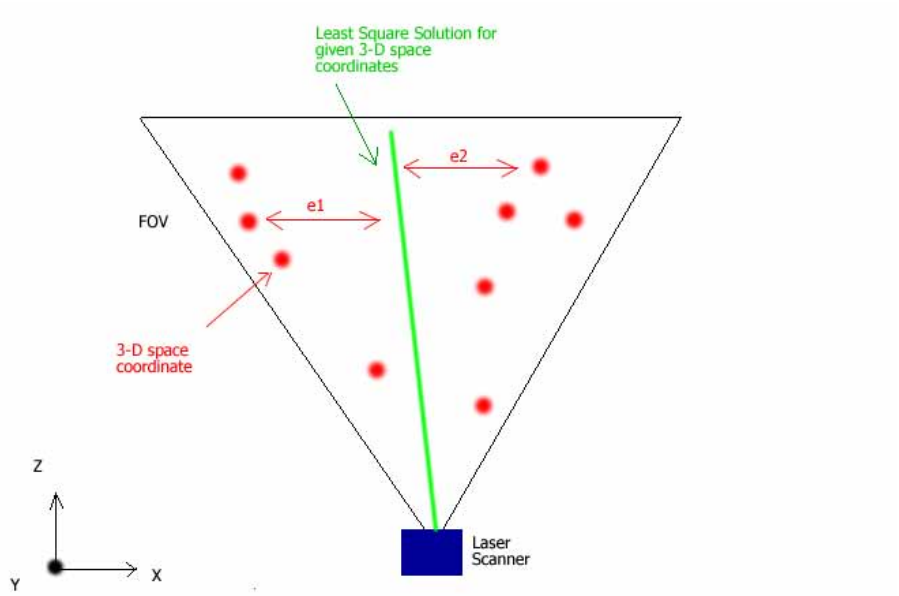

Figure 3.5: Optimum solution for the given 3-D space coordinates

If the 3-D space coordinates are distributed all along the field of vision then the resulting line ought to be good but most importantly, the 3-D space coordinates cannot have any rotation error. For example, suppose that a 3-D space coordinate is nominally pointing at a direction given by $(a\,, b\,, c)$. If the nominal direction is perturbed to $(a + \Delta a\,, b + \Delta b\,, c + \Delta c)$ for some small Δ, points lying far away from the laser will result in a rather large error in meter. Hence, we must make sure that the 3-D space coordinates are rotation invariant at all distances to minimize the potential error that could occur with this rotation in mind.

This approach of camera calibration is the foundation of many camera calibration methods available as of today. As mentioned in [4], all it takes to perform a good calibration is a 3-D space object whose geometry is known with very good precision, and from there finding point correspondences in 3-D space to image pixels. Usually, this is done by using a checkerboard as a calibration object. A number of pictures of the checkerboard from different angles are taken. The calibrator is then asked to mark all the corners on the checkerboard images, and also state the distance between the corners. By knowing this 3-D space distance and the corresponding distance on the image, a mapping is generated.

After finding the camera calibration matrix, or the projection matrix, $\mathbf{P}$, we can remove distortions and other unwanted effects caused by the uncalibrated camera. What $\mathbf{P}$ actually does is acting as a bridge between 3-D-coordinates

and pixel coordinates. The projection matrix contains every camera parameter, which describe the camera's world-pixel mapping, where it is standing and looking at with respect to a certain reference camera. It is clear that if we know $\mathbf{P}$, we must be able to obtain a new perfect mapping between the world and pixel coordinate system.

For more information regarding camera calibration, please refer to [3], [5], [14].

3.1.6 Re-projection error

The re-projection error is defined as the error between the projected 3-D space coordinates to pixel coordinates and the actual pixel coordinates, and it is given by:

$$\min_{\mathbf{p}} e = \sum_{i=1}^{n} \left|\left| \frac{\mathbf{p}_1^T \mathbf{X}_i + p_{14}}{\mathbf{p}_3^T \mathbf{X}_i + p_{34}} - x_i \right|\right|^2 + \left|\left| \frac{\mathbf{p}_2^T \mathbf{X}_i + p_{24}}{\mathbf{p}_3^T \mathbf{X}_i + p_{34}} - y_i \right|\right|^2 \tag{3.30}$$

where $\mathbf{p_j} = [\, p_{j1}\ p_{j2}\ p_{j3}\,]^T$. It is clear that this is the error that should be as small as possible since it defines how well the 3-D space to image space mapping is: the lower error e, the better result as (3.30) indicates. Depicted below is an illustration of the re-projection error.

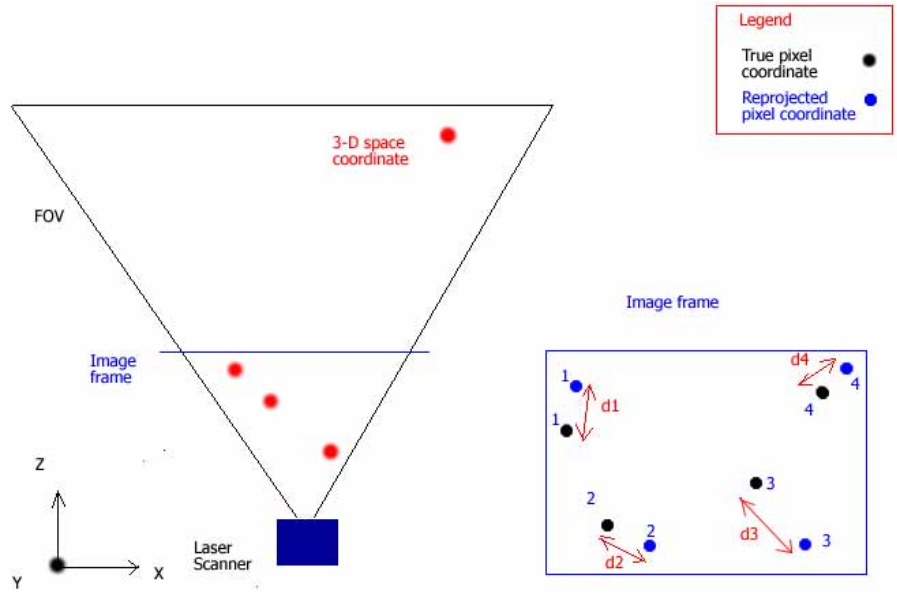

Figure 3.6: Illustration of re-projection error

where d_i, $i = 1...4$ are the pixel errors subject to minimizations. This objective function will be optimized using optimization techniques described in the next sections.

3.2 Imaging problems

When pictures are taken of an object, it is most desirable that the shape and the enviroment it is in remain the same as seen by the eye. Looking at the image taken and comparing it with what the eye can see, no major differences may be found even with careful observations. However, there are some differences, although they are hard to spot. Consider figure 3.7.

Figure 3.7: Picture taken with a normal digital camera [15]

This is a typical picture taken with a modest digital camera (although the guy might be a bit strange). Most of the cameras for private use deliver picture like this one. However, this picture is actually not delivering the information it should. The correct image should look more like figure 3.8.

Figure 3.8: Correct image [15]

A question that may arise is how figure 3.8 is more correct than figure 3.7. Careful examination of figure 3.7 will reveal some flaws in the image. Consider the left-hand side of the checkerboard edge on figure 3.7 As one can see, the checkerboard is a bit bulged in the centre, which actually is the truth. The camera visualizes a straight line as a curvature, and to compansate for that, the edges of the image must be bent towards the opposite direction of the curvation. This is only possible if the camera has been calibrated.

Although knowing that these kind of problems are present, we do not do anything about it in this thesis. The reason is that the lens distortion caused by the hyperspectral camera is not as severe as in figure 3.8 and taking care of them will require another three months worth of work, which we do not have. Consider this section as something worth knowing when camera calibration is to be performed.

3.2.1 Camera model deviation

In the previous section we have modelled our camera as a pinhole camera. This approach has several flaws. Real cameras deviate from the pinhole camera in several aspects. These are listed below:

1. Lens distortion
2. Imaged rays do not necessarily intersect at one point

We will investigate these points more in detail.

1. Ray intersection deviation

A insidious deviation from the pinhole camera model is that the imaged rays do not necessarily intersect at a single point. As a result, there need not be a mathematically precise principal point, or nodal point for a real lens. The consequence of this is that it is impossible to say with complete accuracy that a particular image was taken from a particular location in space; each pixel must be treated as its own separate ray. Although this effect is most noticeable in extreme wide-angle lenses, the locus of convergence is almost always small enough to be treated as a point, especially when the objects being imaged are large with respect to the locus of convergence.

2. Lens distortion

Optical lens systems used in imaging equipment suffer from distortion artefacts, which detract from the quality of the images produced, as we can see in figure 3.7. In applications such as computer vision, the determination of and compensation for distortion is required to enable accurate location, measurement and registration of features in images.

Lens distortion model

To fully model non-linear lens distortion, infinite series are needed. However, in practice it is normally sufficient to model only the dominant radial (a.k.a. barrel or pincushion distortion) using a single parameter, ζ. Assuming that the value of ζ is known, the radial lens distortion can be modelled as

$$r_u \;=\; r_d(1 + \zeta r_d^2) \tag{3.31}$$

where r_u is the correct, undistorted radial distance to a point from the optical centre of the image, and r_d is the distorted radius. As mentioned

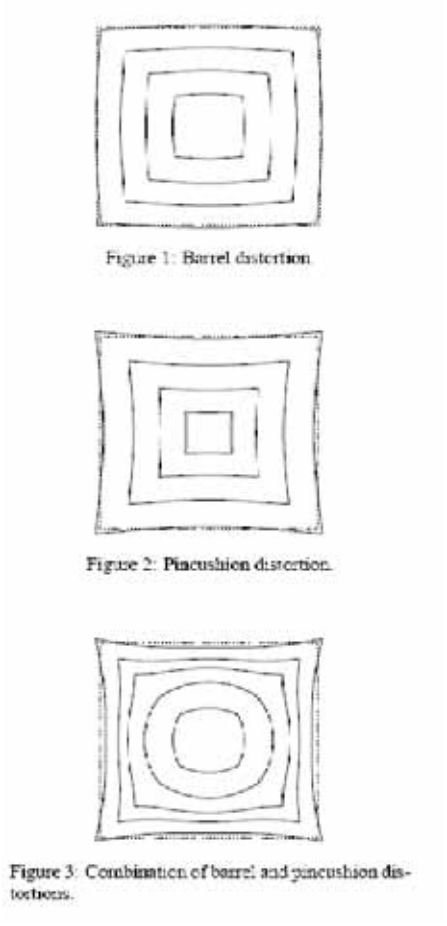

Figure 3.9: Lens distortion models [10]

earlier, we are here only considering the dominant parameters to the distortion. Actually, this radial distortion is an odd power polynomial of infinite length:

$$r_u = r_d + \zeta_1 r_d^3 + \zeta_2 r_d^5 + \zeta_3 r_d^7 \cdots \tag{3.32}$$

But due to the small values of the polynomial coefficients $\zeta_n,\ n \geq 2$, they can be discarded without adding any major errors in the calculation.

Given r_u, computation of r_d requires the solution of this cubic equation. Using the substitution $r_d = w - \frac{1}{3\zeta w}$ yields the quadratic form

$$(w^3)^2 - \frac{r_u}{\zeta} w^3 - \frac{1}{27\zeta^3} = 0 \tag{3.33}$$

with its solution

$$w = \sqrt[3]{\frac{r_u}{2\zeta} \pm \sqrt{\frac{r_u^2}{4\zeta^2} + \frac{1}{27\zeta^3}}} \tag{3.34}$$

The value of r_d is found by substituting either of these roots for w.

Now that the distortion is found, we wish to perform a distortion correction. [13] proposed two methods:

1. Individual pixel resampling
2. Local affine transformation with texture mapping

For more detailed information regarding these two schemes, please refer to [13] and [10], [26] for more general information concerning lens distortion.

3.3 Automatic calibration tools

As mentioned earlier, this thesis covers how to make use of the data aquired from the hyperspectral camera and the scanning 3-D laser to detect anomalies. In order to detect anything at all, both sensors need to be incorporated by one another in the sense that what is seen by one sensor must also be seen by the other one. Suppose we have a camera setup where the sensors are standing next to each other, pointing in the same direction. An illustration of both sensors' vision fields is depicted in figure 3.10.

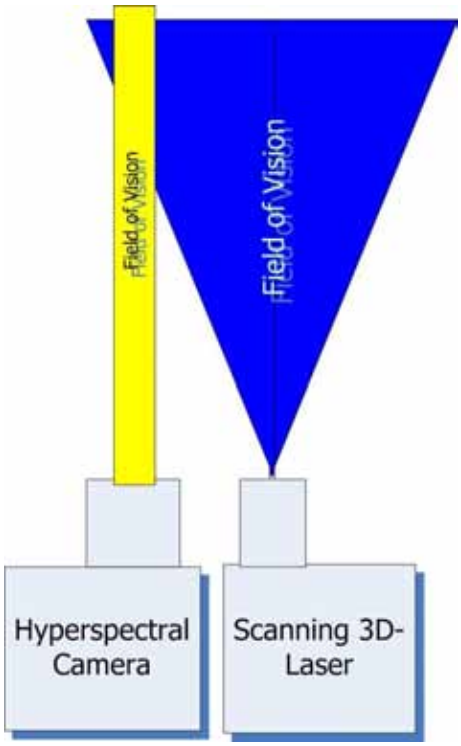

Figure 3.10: Vision field for both sensors

As shown in figure 3.10, the only thing we are interested in is the small section where the yellow part is overlapped by the blue part. The hyperspectral camera returns a hyperspectral image in the size of 192×512 pixels, whereas the laser returns images much bigger in dimensions. Hence, the image aquired by the hyperspectral camera must be contained in the image from the laser. Figure 3.11 illustrates an example of this.

The idea behind this automatic calibration is that the sensors will be placed in arbitrary positions, in which the vision field intersection discussed earlier will be obtained and by looking at an object, it is desirable to have the the sensors calibrated, i.e. $\mathbf{P}$ will be generated.

However, there is a number of problems with this scheme:

- As seen in the lower image of figure 3.11, the intersection area of the vision fields is rather small, at least for the eye. Is it possible to find sufficiently many good point correspondences within this area?

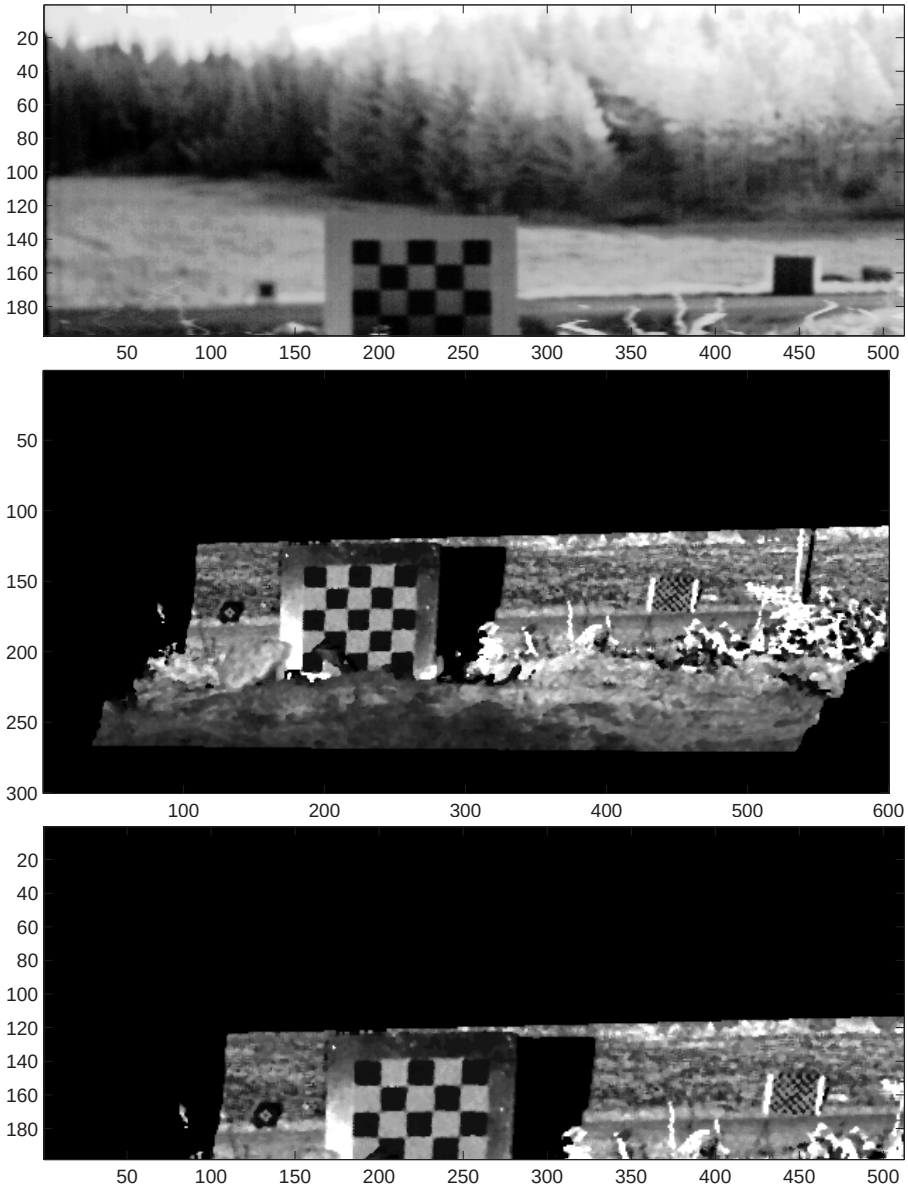

Figure 3.11: Example of vision field intersection

- The hyperspectral camera might not give the indications of feature points due to its low resolution
- Is it possible to find any feature points at all in complex enviroments such as highly dense woods?

The answers to these key points will determine the quality of the automatic calibration.

3.3.1 Gradient filtering

As mentioned earlier, one of our biggest concerns is the fact that the hyperspectral camera has a very low resolution. Using the camera looking at objects far away will not give good results. Consider figure 3.12 where only few of the hyperspectral camera's spectral bands are used.

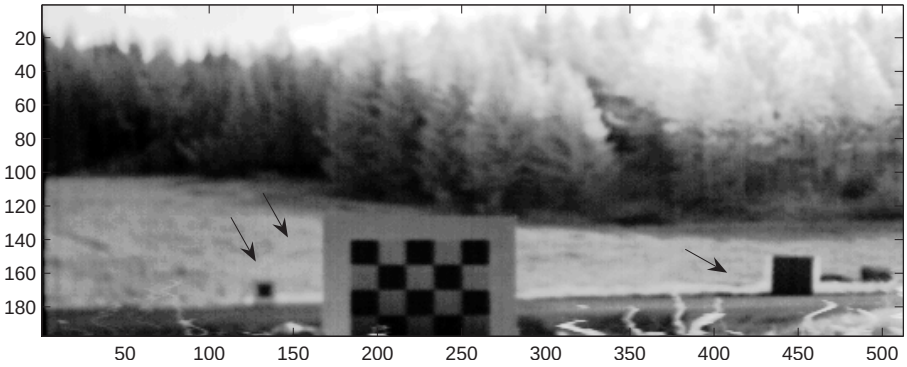

Figure 3.12: Hyperspectral image where a few spectral bands are used

On this image, there are three rather big checkerboards placed in different positions indicated by the arrows. Looking at the smallest on the left-hand side, it is impossible to determine the corners due to the low resolution in the hyperspectral camera. The same goes for the one on the right-hand side. By using *gradient filters*[2], we can detect strong lines although the resolution is fairly low, since the pixel transition is still there even though the resolution is low. Introduce the gradient filters:

$$\nabla_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \nabla_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.35}$$

Convolving the image, $f(x, y)$, with both filters and summing the result will yield the filtered image, $\hat{f}(x, y)$:

$$\hat{f}(x, y) = \nabla_x * f(x, y) + \nabla_y * f(x, y) \tag{3.36}$$

where $*$ denotes the convolution operator. An example of the result using this scheme is given in figure 3.13 below. Worth noting is that only a few bands of the hyperspectral image is being filtered.

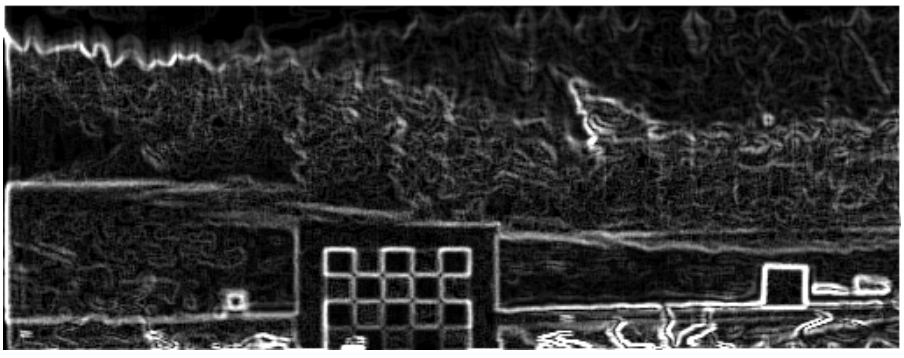

Figure 3.13: Gradient filtered hyperspectral image

This is a high-pass process in which the high-frequency components in the image, i.e. corners and edges are preserved. This image will be further processed to extract image features.

[2]This is also known as the Prewitt filter

3.3.2 Information theory

To provide with an adequate measure on the improvement using this automatic calibration scheme, concepts in information theory must be called for. The main concepts used here are

- *entropy*
- *mutual information*

and will be discussed more in detail.

1. Entropy

Entropy is defined as the average information per message of a source. Consider a memoryless source $\mathcal{S}$, emitting messages $m_1, m_2, ..., m_n$ with the probabilities $\sum_{i=1}^{n} P(m_i) = 1$. The information content, $\mathcal{I}_i$ of message m_i is given by:

$$\mathcal{I}_i \;=\; \log_2 \frac{1}{P(m_i)} \quad \text{bits}$$

The average information per message of a source $\mathcal{S}$ is called its entropy, given by

$$H(m) \;=\; \sum_{i=1}^{n} P(m_i)\,\mathcal{I}_i \quad \text{bits}$$

$$H(m) \;=\; \sum_{i=1}^{n} P(m_i)\,\log_2 \frac{1}{P(m_i)} \;=\; -\sum_{i=1}^{n} P(m_i)\,\log_2 P(m_i) \quad \text{bits} \qquad (3.37)$$

An intuitive understanding of information entropy relates to the amount of *uncertainty* about an event associated with a given probability distribution. Consider a box containing many colored balls. If the balls are all of different colors and no color predominates, then the uncertainty about the color of a randomly drawn ball is maximal. On the other hand, if the box contains more red bals than any other color, then there is slightly less uncertainty about the result that a red ball is drawn.

2. Joint entropy

Given a random variable X, the entropy for this variable is $H(X)$ as discussed in section 3.3.2.1. Consider another random variable Y, containing events y_j occurring with probabilities $\sum_{j=1}^{m} P(y_j) = 1$. The variable Y has an entropy given by $H(Y)$. If X and Y describe related events, the total entropy of the system may not be $H(X) + H(Y)$. For example, imagine choosing an integer between 1 to 8 with equal probability for each integer. Let X represent whether the integer is even, and Y represent whether the integer is prime. One-half of the integers between 1 and 8 are even, and one-half are prime, hence, $H(X) = H(Y) = 1$. However, if we know that the integer is even, there is only a 1 to 4 chance that it is also a prime; the distributions are related. Instead of looking at the entropy as a linear combination of two random processes, it must rather be seen as the entropy of a joint random process. Given two random processes, X and Y, find the joint probability, $P(X, Y)$, which is defined as the pair of event outcome that satisfy

$$P(X, Y) \;=\; \frac{\partial^2}{\partial X \partial Y} P(X \leq x, Y \leq y) \qquad (3.38)$$

When the joint probability is known, form the joint entropy defined by:

$$H(X,Y) \;=\; -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i,y_j)\log_2 P(x_i,y_j) \tag{3.39}$$

$$x_i \in X \;\forall\; i = 1,2,...,n$$

$$y_j \in Y \;\forall\; j = 1,2,...,m$$

3. Conditional entropy

Consider a discrete memoryless channel. Let a source emit symbols $x_1, x_2, ..., x_n$. The receiver receives symbols $y_1, y_2, ..., y_m$. The set of symbols $\{y_k\}$ may or may not be identical to the set $\{x_k\}$, depending on the nature of the receiver. If the receiver used is not of the type 'optimum receiver', the constraint that the set $\{y_k\}$ is identical to the set $\{x_k\}$ may not be valid. If the channel is noiseless, then the reception of some symbol y_j uniquely determines the message transmitted. However, due to the noise there is a certain amount of uncertainty regarding the transmitted symbol when y_j is received. If $P(x_i|y_j)$ represents the conditional probabilites that x_i was transmitted when y_j is received, then there is an uncertainty of $\log_2[1/P(x_i|y_j)]$ about x_i when y_j is received. When this uncertainty is averaged over all x_i and y_j, we obtain $H(X|Y)$, which is the average uncertainty about a transmitted symbol when a symbol is received (a.k.a conditional entropy). Thus

$$H(X|Y) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i,y_j)\,\log_2\frac{1}{P(x_i|y_j)} \quad \text{bits per symbol} \tag{3.40}$$

4. Mutual information

The mutual information, or transinformation, of two random variables is a quantity that measures the mutual dependence of the two variables. Given the random variables X and Y, the mutual information can formally be expressed as

$$I(X;Y) \;=\; \sum_{x\in X}\sum_{y\in Y} P(x,y)\log_2\frac{P(x,y)}{P(x)P(y)} \tag{3.41}$$

where $p(x,y)$ is the joint probability distribution function of X and Y. It can be seen by inspection that (3.41) can be written as:

$$I(X;Y) \;=\; H(X) - H(X|Y) \;=\; H(X) + H(Y) - H(X,Y) \tag{3.42}$$

To illustrate all the quantities discussed here, consider figure 3.14

where the blue circle represents the entropy of the random variable X and the red circle the entropy of the random variable Y. Depicted is also the conditional entropy for each random variable, which can be found as the half-moon of respective color. The white section in between the half-moons is the mutual information of the two random variables.

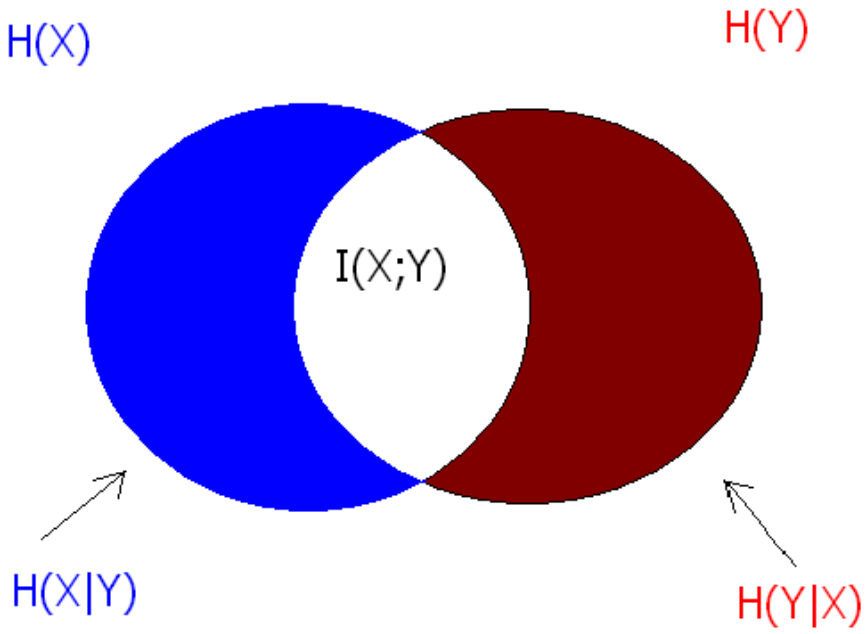

Figure 3.14: Information theoretical quantities

4 Work procedure

In this section, a detailed presention of the work procedure will be given. It will start off from the very beginning with data acquisition to the iterative solution of the projection matrix. A brief list of discussed topics is given below.

- Obtain initial estimate of $\mathbf{P}$
- Down project 3-D laser image
- Image enhancement
- Common feature extraction using Harris corner detector
- Calibration
- Optimization: iterative minimization of re-projection error, e.

4.1 Initial estimate of $\mathbf{P}$

As mentioned earlier, the projection matrix $\mathbf{P}$ can be seen as a bridge between the 3-D space coordinate system and the pixel coordinate system. To get a good initial guess of this matrix, point correspondences between these two systems need to be obtained. This is simply done by using reference objects, e.g. checkerboards, and pinpoint the same points in both systems. After obtaining a number of point correspondences with fairly good precision, i.e. the point obtained from the laser image should be a point on the object and not on the noise/particles surrounding the reference object, using (3.24) described in section 3.1.5, will yield a very good inital guess of $\mathbf{P}$. As discussed in section 3.1.5, points uniformly distributed on the entire image is preferred. Also, using correspondences in varying depth, i.e. Z on the 3-D space, will lock the least square solution for points lying far away. Consider figure 4.1. The ideal scenario will be when point correspondences are distributed uniformly on the image frame and image frames placed close to each other to cover the 3-D space of interest. If the frames are placed close enough and point correspondences on the entire image frames are found, then the 3-D reconstruction of the scene is done and the projection matrix will be excellent, since there is a f_i for every Z_i then and we have all the information there is to have for point matching [1]. But due to the hyperspectral camera's limitation this will not happen, since there is no zoom function on the hyperspectral camera. Hence, there will only be one image frame per image to cover for all the Z-values.

Finding point correspondences on the hyperspectral image has been very hard to do due to the low resolution in the hyperspectral sensor. Consider the two checkerboards on the left and right-hand side in figure 4.1. On the laser image depicted in figure 4.2 the checkerboards are displayed clearly and finding the crossings is an easy task. But on the hyperspectral image on the other hand,

[1] These point correspondences on the image frames should resemble of a shoebox with image frames coordinates

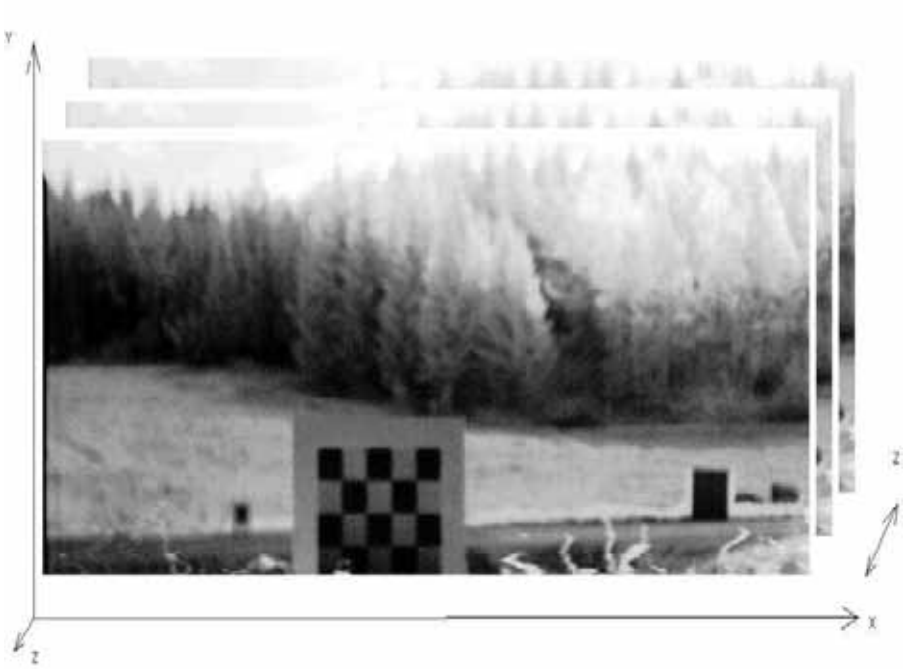

Figure 4.1: Finding point correspondences on each frame yielding $\mathbf{P}$

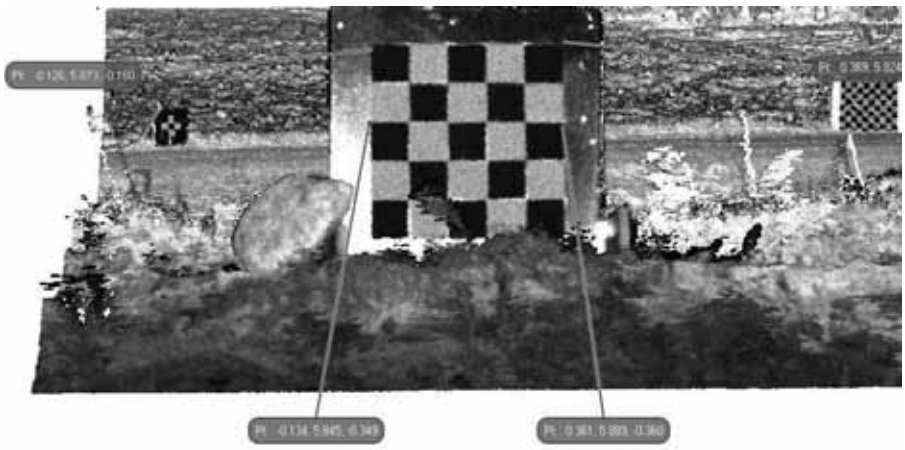

Figure 4.2: Corresponding laser image for the 3-D space coordinates

this is not possible. Hence, finding point correspondences far away will result in noisy point, i.e. the pinpointed coordinate is subject to perturbance, and the projection matrix will be perturbed as well.

4.2 Laser image down projection

After acquiring the 3-D laser data it must be down projected to a 2D-image in order to find point correspondences. The down projected laser image should look like a picture taken with the hyperspectral sensor at the laser sensor's position. Hence, the resulting image should look like the hyperspectral image with a certain translation. The method used for this down projection is based on the projection matrix found earlier. After applying the manually picked

point correspondences, a projection matrix is obtained. What is known from previous discussion is that the size of the hyperspectral image is 192×512 pixels. Hence, the size of the down projected laser image must be within these bounds. What has been used here is the fact that the 3-D-2-D mapping is given by:

$$\begin{bmatrix} \hat{\mathbf{x}}_i \\ f \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X}_i \\ 1 \end{bmatrix}, \qquad \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \hat{x}_i/f \\ \hat{y}_i/f \end{bmatrix}$$

Using this equation, a number of pixels will have the same coordinates due to the fact that there will be laser hits on the same 3-D space coordinate (X, Y) with varying depth Z. The down projection can be obtained by finding the corresponding 3-D space coordinates for each pixel. After finding all the 3-D space coordinates, putting the corresponding intensity value on the pixel coordinate will yield the down projected image. As mentioned earlier, there will be pixels with more than one 3-D space coordinate, resulting in depth ambiguity, i.e. it is impossible to determine which 3-D space coordinate to use for a particular pixel coordinate. The solution to this is to only use the 3-D space coordinates that are closest to the sensor, i.e. lowest Z-value. The reason is that the hyperspectral sensor should not be able to see anything behind a solid object. Hence, using the 3-D space coordinates that are closest to the hyperspectral camera for every pixel will result in a correct down projection of the laser data.

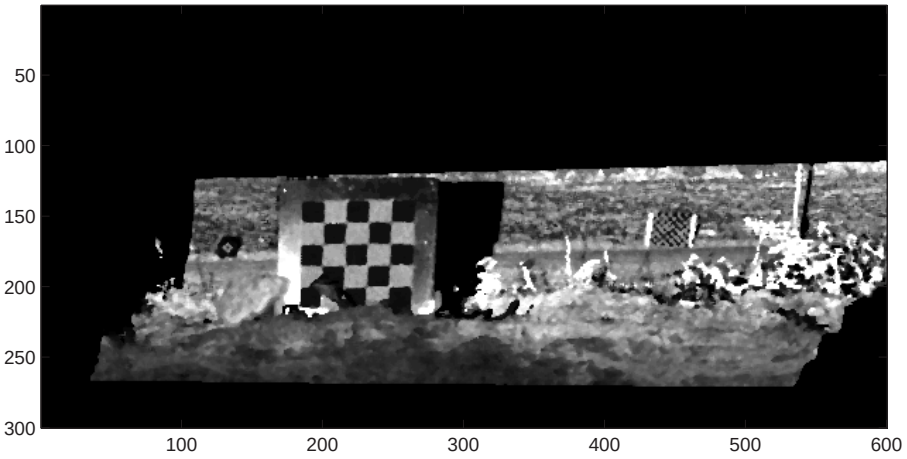

Figure 4.3: Down projected laser image

4.3 Enhancement of projection matrix $\mathbf{P}$

4.3.1 Image enhancement

After obtaining the down projected laser image, image enhancement must be performed in order to find any feature points on both images. The list below states, in descending order, how this enhancement is done:

1. Intensity value normalization
2. Histogram equalization
3. Edge detection

Intensity value normalization

Intensity value normalization is achieved simply by finding the maximum pixel value in both images and dividing each pixel value with this particular value. This is done in order to get pixel values that are somewhat correlated, i.e. both images should have corresponding pixel values that are close to each other.

Histogram equalization

Histogram equalization is a common technique for enhancing the appearance of images. Given a predominantly dark image, its histogram will be skewed towards the lower end of the grey scale and all the image detail is compressed into the dark end of the histogram. Histogram equalization is a technique that 'stretches' the grey levels at the dark end to produce a more uniformly distributed histogram, which will result in a much clearer image. Depicted in figure 4.4 below illustrates this technique.

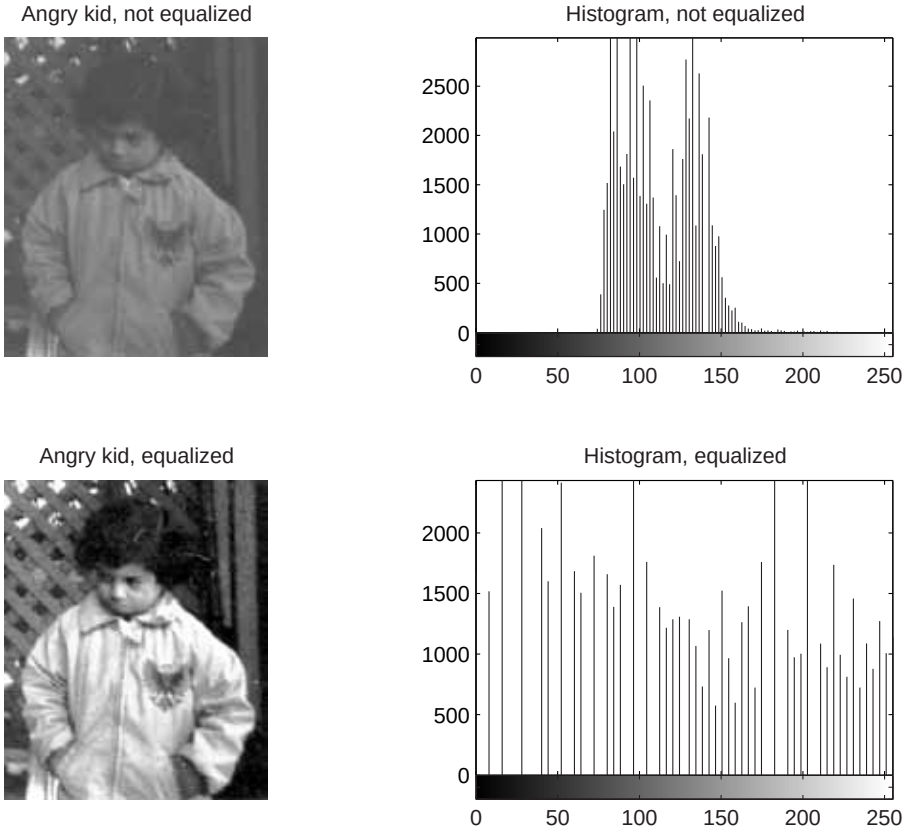

Figure 4.4: Histogram equalization

Edge detection

The earlier introduced gradient filter is used for detecting strong lines and edges on both images. However, there are several filter kernel proposed for this purpose such as Sobel detector, Canny detector etc. After the evaluation of these edge detectors, the gradient filter was the filter kernel giving the best result (at least for the eye). Although Canny's edge detector is the standard in edge detection, for the low resolved hyperspectral images, it won't give any satisfactory results.

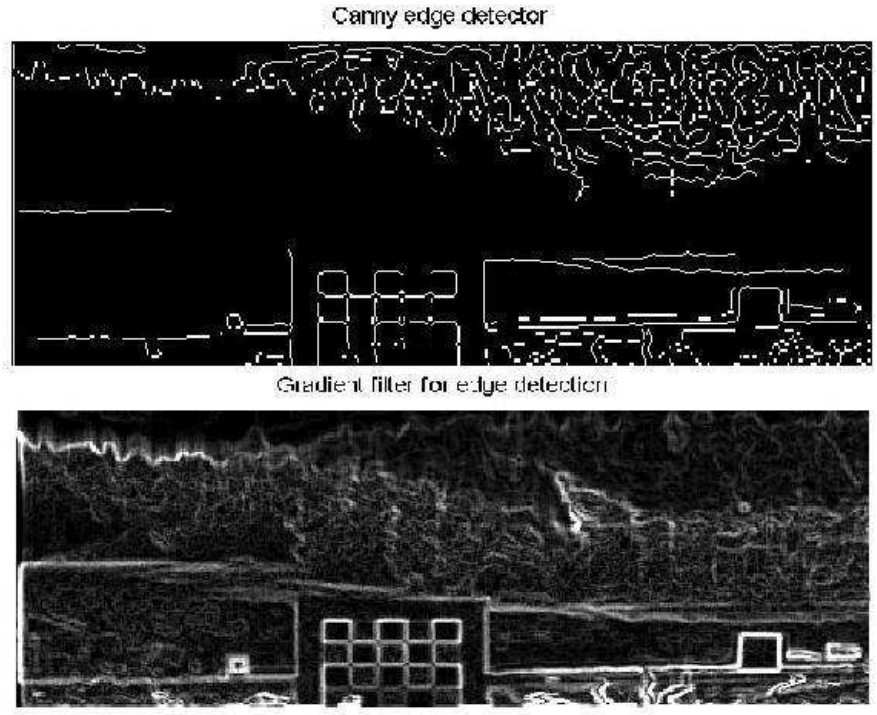

Figure 4.5: Canny edge detector (upper) vs. Gradient filter (lower)

4.3.2 Common feature extraction

After the image processing, features on both images shall be found. Used here is Harris corner detector given in appendix A. The detector is applied on the hyperspectral images, where the image has been partioned into smaller regions comprised of 24×32 pixels. In each of these regions, the strongest Harris corner response is stored for further analysis. Given an image of 192×512 pixels and the patch size of 24×32 pixels, there will be 128 corners detected per image. Now, consider the down projected laser image. This image must be contained in the hyperspectral image, resulting in a laser image which is much smaller than the original. After cutting out the laser image, the image should look like figure 4.7. Now, to get a good match between corners on both images, corners in the laser image will be obtained based on where the corners are found in the hyperspectral image. Taking the corners' coordinates in the hyperspectral image to form a patch, which is centered around the corners' coordinates and 2 pixels wide in every direction (i.e. horizontal and vertical) and using this patch to evaluate the strongest corner within this region will result in corners that are very well matched. A matching of this kind is depicted in figure 4.7 below, where the red crosses indicates corners in the particular area.

Since the corners in the laser image are obtained by evaluating the hyperspectral corner patch on the laser image, it can be assumed that detected corners are almost perfectly matched. Worth noting is that if the coordinate of the corner in the hyperspectral image happens to be empty space in the down projected laser image, then this matching is void, i.e. it does not return anything and that corner is omitted.

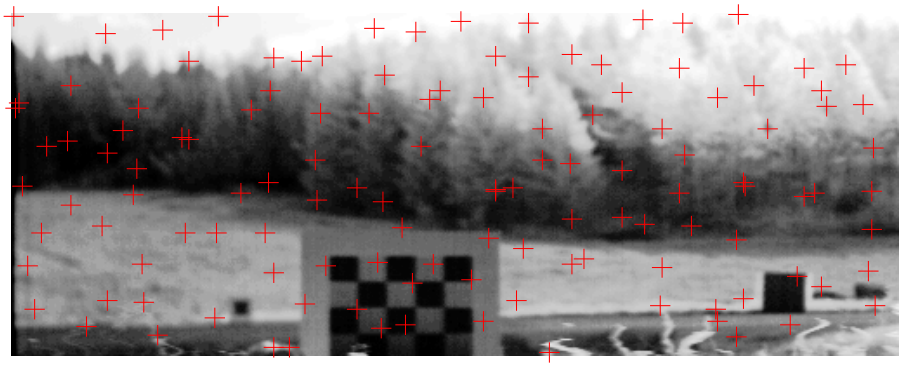

Figure 4.6: Corner extracted in hyperspectral image using Harris corner detector

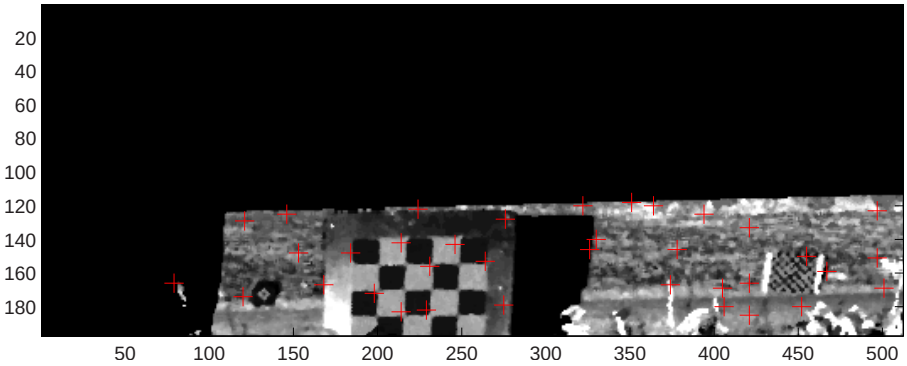

Figure 4.7: Corner extracted in laser image using Harris corner detector

4.3.3 Retrieval of 3-D space coordinates

After obtaining the matched corners in both images, the 3-D space coordinates on the down projected laser image must be retrieved. When the mapping between the 3-D space and the image space is performed it simply returns two tables: one with all the 3-D space coordinates from the point cloud and one with the corresponding pixel coordinates for each 3-D space coordinate. To retrieve the 3-D space coordinates from the pixel coordinates of the corners, we just need to find out which 3-D space coordinates these pixels were transformed from. Simply put, given the corner coordinates $[x_{\text{corner}}, y_{\text{corner}}]$, find these coordinates in the matrix $\hat{\mathbf{x}}_{\text{corner}}$ and return $\mathbf{X}_{\text{point cloud}}$

$$\begin{bmatrix} \hat{\mathbf{x}}_{\text{corner}} \\ f \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X}_{\text{point cloud}} \\ 1 \end{bmatrix}$$

$$\mathbf{x}_{\text{corner}} = \hat{\mathbf{x}}_{\text{corner}}/f$$

which is a straightforward procedure. However, as mentioned earlier, each pixel coordinate will generate a tunnel with 3-D points resulting in a depth ambiguity, i.e. it is impossible to determine which 3-D space coordinates correspond to the pixel of interest. For example, it is possible that a pixel is transformed from 3-D space coordinates $\mathbf{X}_{\text{spat}}$ such as

$$\begin{bmatrix} \hat{\mathbf{x}}_1 \\ f \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X}_{\text{spat}} \\ 1 \end{bmatrix}, \quad \mathbf{X}_{\text{spat}} = \begin{bmatrix} 1.1 & 1.1 & 1.2 & 1.3 \\ 0.1 & 0.1 & 0.2 & 0.3 \\ 1.0 & 1.1 & 50 & 99 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

where the third row in $\mathbf{X}_{\text{spat}}$ represents the Z-values in meters. So the question is which 3-D space coordinate should be used when this depth ambiguity is present? The answer to that is to not use any of them. Since it is impossible to determine which one is correct, it is better to only use the 3-D space coordinates which lie within a certain boundary. A constraint used is to only select the 3-D space coordinates that satisfy the following criterion:

$$\bar{Z}_{\max} - \bar{Z}_{\min} \ < \ \text{threshold} \tag{4.1}$$

where $\bar{Z}$ is a vector containing all the Z-values for a certain pixel and threshold is a scalar. This scalar determines how much the laser hits can deviate in depth and should be adjusted according to the current conditions during the data acquisition. If the data is noisy then the threshold should be set larger to compensate for the noise and so on. The value used in this thesis is 0.2, which means that the laser shots can at most be 20 cm apart from each other in Z-direction. After using this constraint, there will still be a number of 3-D space coordinates to choose from. To avoid any unnecessary problems, the chosen 3-D space coordinate used is simply the mean value of the coordinates that satisfy (4.1).

4.3.4 Calibration and Optimization

After obtaining the 3-D space coordinates from the matched corners the camera calibration can begin. Using (3.24) will yield the projection matrix. Recall from previous discussion that these equations are basically optimization processes that optimizes with respect to the metric distances. The next step is to optimize the solution further by minimizing the re-projection error discussed in section 3.1.6 using Sequential Quadratic Programming (SQP), which can be found in littertures and reports such as [12] and [23].SQP minimizes the re-projection error and does not care if the metric distances will increase doing so. Since the metric distances for the 3-D space coordinates already are optimized, it can be assumed that using SQP won't increase the metric error drastically. Hence, these two optimization steps can be summerized by:

1. The least square solution optimizes a straight line to the 3-D space coordinates by finding a line that minimizes all the points' distances to the line

2. SQP optimizes the reprojected pixel coordinates by minimizing the re-projection error and does not care if the metric distances in the least square solution increase

In Matlab, this algorithm has been implemented in a number of functions. The function we used is `fmincon` which can be found in the optimization toolbox. `fmincon` is a constrained optimization techinque in Matlab that finds a minimum of a constrained nonlinear multivariable function using Sequential Quadratic Programming

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to}$$

$$\begin{aligned} \mathbf{A} \cdot \mathbf{x} &\leq \mathbf{b} \\ \overrightarrow{lb} \leq \mathbf{x} &\leq \overrightarrow{ub} \end{aligned}$$

where $\mathbf{x}$ is the solution vector, $\mathbf{A}$ a matrix containing the vector coefficients, $\mathbf{b}$ is a upper bound for the function value, and $\overrightarrow{lb}$ and $\overrightarrow{ub}$ the lower and upper bound vectors for the solution vector, $\mathbf{x}$, respectively.

What is done here using the function `fmincon` is using the estimated projection matrix to compute the optimal value of the matrix entries, with the constraint that the sum of all entries should be greater than 0 and less than 10. The initial estimation of $\mathbf{P}$ is

$$\mathbf{P}_0 = \begin{bmatrix} 0.6492 & -0.0364 & 0.1850 & -0.2209 \\ -0.0347 & -0.6925 & 0.0661 & 0.0952 \\ -0.0001 & -0.0002 & 0.0007 & 0.0000 \end{bmatrix}$$

After performing the optimization of $\mathbf{P}_0$, the optimized projection matrix is given by

$$\mathbf{P}_1 = \begin{bmatrix} 0.0373 & -0.0149 & 0.0087 & 0.5251 \\ -0.0053 & 0.0510 & 0.0031 & 0.2054 \\ -0.0001 & -0.0001 & 0.0000 & 0.0057 \end{bmatrix}$$

Although the resulting matrices differ by a lot, that does not rule out the quality of one another. Recall from the discussion in section 3.1.4 where the projection matrix can be partioned as:

$$\mathbf{P} = \mathbf{\Phi}\,[\mathbf{R}\,|\,\mathbf{T}]$$

where $\mathbf{R}$ comprises of rotation about a certain axis and $\mathbf{T}$ is the column vector containing information about the translation multiplicated by the rotation. If the assumption that camera parameter matrix, $\mathbf{\Phi}$, is constant throughout the iterations then it is clear that the algorithm is only adjusting the external camera parameters, i.e. it is trying to resolve the position where the sensors should have been placed. Hence, major adjustment in the projection matrix after iterations can be seen as position adjustment of the sensors. However, it is not certain that $\mathbf{\Phi}$ is constant throughout the iterations. Perhaps the guessed focal length is incorrect or the skewness not correctly estimated etc. Hence, what the projection matrices state after each iteration do not really give any good information on how well this algorithm is behaving. The main theme to this discussion would be that it is necessary to use different measure to determine the improvement or degeneration of the resulting projection matrices after each iteration.

4.3.5 Iteration procedure

Now we have all the tools necessary to obtain a new, optimized projection matrix. However, if we want to use this new projection matrix to obtain another further optimized projection matrix, we must perform an iteration of the whole algorithm. The steps involved are described below:

1. Use the estimated projection matrix $\mathbf{P}_0$ to obtain the down projected laser image

2. Apply the image enhancement techniques described in section 4.3.1 on the hyperspectral image and the laser image and find features on both images

3. Retrieve 3-D space coordinates from the features found in the laser image

4. Use (3.24) to obtain a new projection matrix and optimize the new matrix to obtain $\mathbf{P}_{\text{new}}$ using LMA

5. Replace $\mathbf{P}_0$ with $\mathbf{P}_{\text{new}}$ and redo all steps until a satisfactory result is obtained

4.3.6 Image alignment and error estimation

Before getting deeper in the topic, we shall recall what we want to accomplish here. What shall be done here is to align images from different sensors, find feature points in both images and using these points to perform a calibration on the sensors. What is given is an initial guess of the projection matrix using a certain setup for the sensors. Using this initial guess, we want to perform camera calibration for any arbitrary setups. One way to determine if the iterated projection matrix is better or worse is to align both images and measure the alignment. The alignment is measured by inspecting how well the laser image is being mapped to the hyperspectral image, which is the reason why information theoretical quantities have been called for. How well both images map is the quantity better known as the mutual information. To compute the mutual information discussed in section 3.3.2, the following probability quantities must be retrieved:

1. Histogram of both images ($P(X)$ and $P(Y)$)
2. 2-D histogram ($P(X,Y)$)

1. Histogram, $P(X)$ and $P(Y)$

To obtain the probability that a certain pixel has a certain pixel value, a histogram can be used for this computation. This is done by computing the histogram and dividing it by the total number of pixels in the picture. In Matlab this can simply be done via the command

```
nx=hist(x(:),256);
ny=hist(y(:),256);
px=nx/numel(x);
py=ny/numel(y);
```

2. 2-D histogram, $P(X,Y)$

The 2-D histogram is basically a table with pixel value distribution on both images. Suppose two images A and B are given with pixel values ranging from $0-4$:

$$A = \begin{bmatrix} 0 & 4 & 1 \\ 2 & 4 & 3 \\ 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 3 & 4 \\ 0 & 1 & 1 \end{bmatrix}$$

then the resulting 2-D histogram will be where the pixel values in both images

Table 4.1: Resulting 2-D histogram

		B				
		0	1	2	3	4
	0	1	0	0	0	0
	1	1	1	0	0	1
A	2	0	1	1	0	0
	3	1	0	0	0	1
	4	0	0	0	1	0

set the axes on the table. Taking each table entry and dividing by how many pixels an image has (in this case 9) will yield the joint probability $P(X,Y)$. This is implemented in Matlab using following commands:

```
pxy=zeros(256); % Number of gray scale levels
for i=1:size(x,2)
    for j=1:size(x,1)
        pxy(x(j,i),y(j,i))=pxy(x(j,i),y(j,i))+1;
    end
end
pxy=pxy/numel(pxy)
```

After obtaining the histograms the entropy calculations described in section 3.3.2 can begin. Most importantly is to compute the mutual information between the two images. We are given the hyperspectral image and the down projected laser image. What we want to do is to adjust the laser image to align as much as possible with the hyperspectral image, since the coverage of the laser is much greater. To compute or measure how good alignment is, we can use the mutual information between the two images. Consider figure 3.14. If the hyperspectral image is the random variable X and the laser image is Y, then what we want to accomplish is to adjust Y, or $H(Y)$ to increase the mutual information $I(X,Y)$, which determines how aligned the images are, since the mutual information measures how good the mapping is between these two sets/systems. How good this mapping is determines how many common features can be extracted from both systems and it is crucial from the sensor calibration point of view.

To illustrate some examples, consider figure 4.8 and 4.9. The left image in figure 4.8 represents a down projected laser image using $\mathbf{P}_0$ where the alignment is pretty much as good as it gets, since the reference 3-D space and pixel coordinates are selected with high precision. Also depicted in figure 4.8 and 4.9 is the mutual information between the hyperspectral image and laser image. Note that the unit is bit/pixel, since $\log_2(\cdot)$ is being used. Now, let's assume that the projection matrix has been perturbed, resulting in the following projection matrix

$$\mathbf{P}_{\text{perturbed}} = \begin{bmatrix} 0.6492 & -0.0364 & 0.1850 & -0.2209 + 0.5 \\ -0.0347 & -0.6925 & 0.0661 & 0.0952 + 0.5 \\ -0.0001 & -0.0002 & 0.0007 & 0.0000 + 0.5 \end{bmatrix}$$

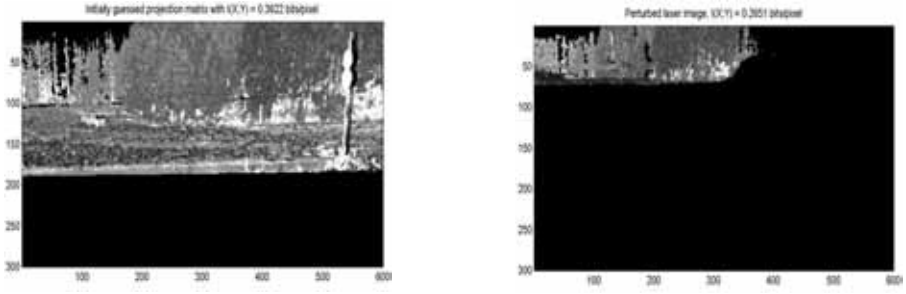

Figure 4.8: Using iteration to resolve perturbed projection matrix

i.e. only the external parameters are being perturbed. With this perturbation, the resulting down projected laser image is depicted in the right image in figure 4.8. The left image in figure 4.9 depicts the resulting image after one iteration and the right image in the same figure depicts the image after 6 iteration. The mutual information, $I(X;Y)$ for each images are given, from top left to lower right images:

$$I(X;Y) = [0.3622,\ 0.2651,\ 0.4828,\ 0.3573] \quad \text{bits/pixel}$$

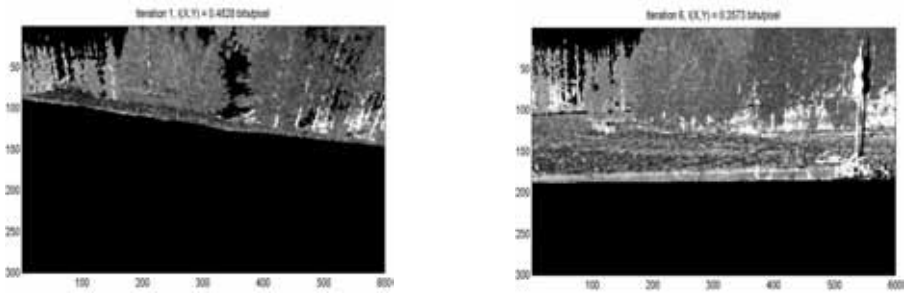

Figure 4.9: Using iteration to resolve perturbed projection matrix

Figure 4.10: Corresponding hyperspectral image using one band

Hence, the more pixels the more information can be acquired from the images. As figure 4.10 indicates, the more overlap we have in both pictures, the better mutual information can be acquired.

We have developed an algorithm that can withstand projection matrix perturbation, i.e. we can use different external setups for the sensors using our initially guessed projection matrix, and yet find a proper and robust solution for the projection matrix using a few iteration steps.

5 Results

A brief presentation of the results we obtained in this thesis will be given here. To demonstrate the results we simulate the situation were the sensors have been placed randomly during the data acquisition, which is simply done by perturbing the last column in the projection matrix $\mathbf{P}$. As the results show is that the algorithm developed here is outstanding in finding its way back from being perturbed to its nominal position. Provided is also a short movie which shows the strength in combining the sensors given in this thesis for anomaly detection.

5.1 Example 1

For this test the projection matrix used is

$$\mathbf{P}_{\text{perturbed1}} = \begin{bmatrix} 0.6492 & -0.0364 & 0.1850 & -0.2209 - 0.01 \\ -0.0347 & -0.6925 & 0.0661 & 0.0952 + 0.02 \\ -0.0001 & -0.0002 & 0.0007 & 0.0000 - 0.02 \end{bmatrix}$$

The originaly estimated laser image is depicted in figure 4.3 with the corresponding hyperspectral image given in figure 4.6. Using the perturbed projection matrix the resulting image is given below:

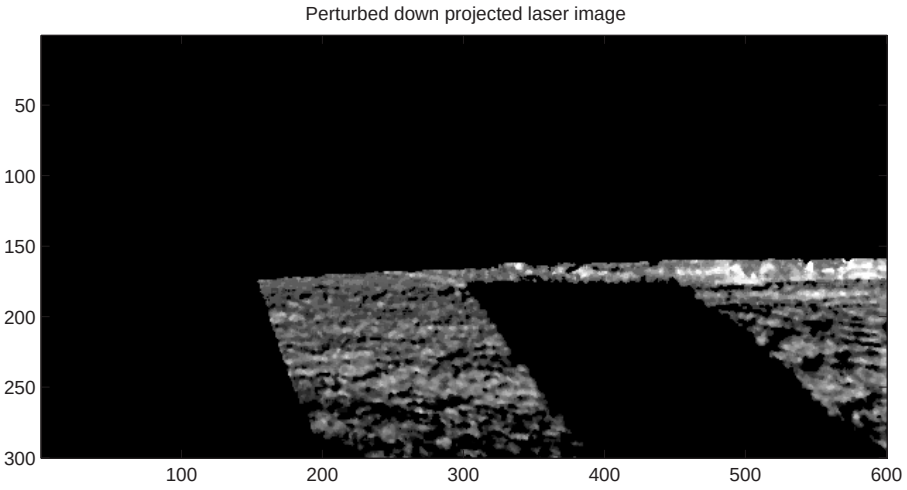

Figure 5.1: Laser image using $\mathbf{P}_{\text{perturbed1}}$

Using the iterative steps described earlier, the resulting images are depicted below.

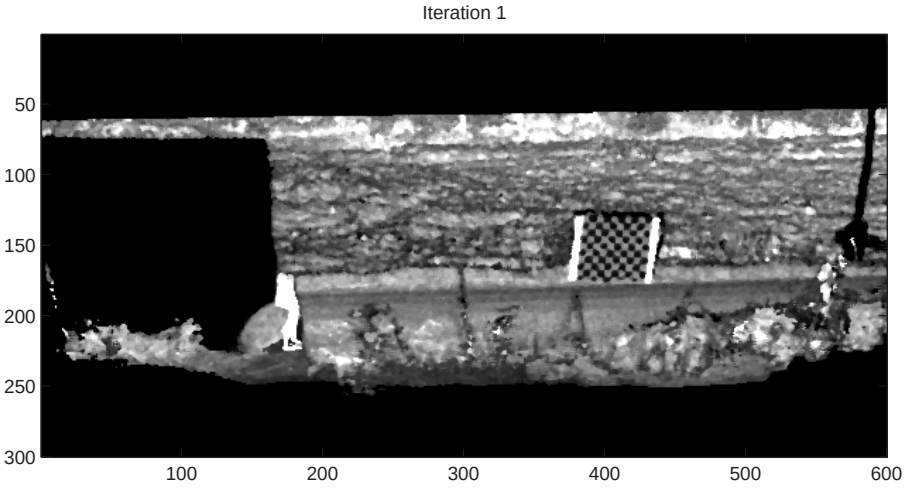

Figure 5.2: Iteration 1 for $\mathbf{P}_{\text{perturbed1}}$

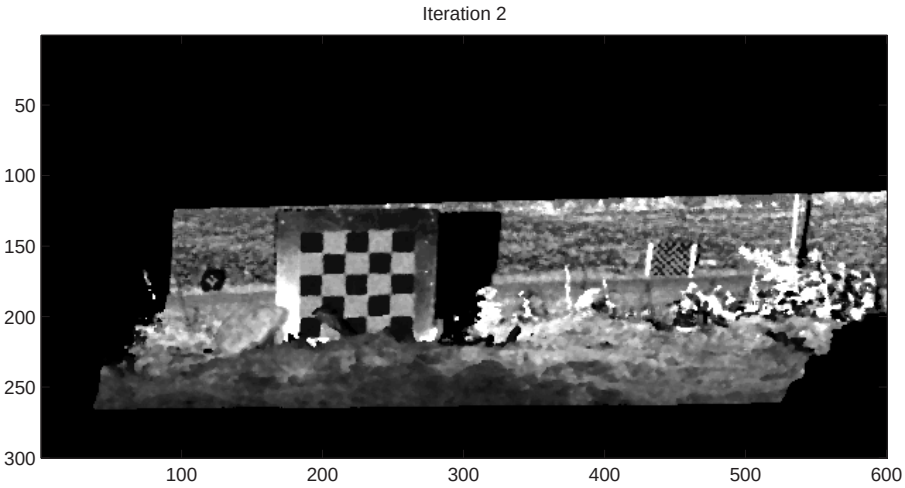

Figure 5.3: Iteration 2 $\mathbf{P}_{\text{perturbed1}}$

5.2 Example 2

The next test we used the projection matrix

$$\mathbf{P}_{\text{perturbed2}} = \begin{bmatrix} 0.6492 & -0.0364 & 0.1850 & -0.2209 + 0.01 \\ -0.0347 & -0.6925 & 0.0661 & 0.0952 - 0.01 \\ -0.0001 & -0.0002 & 0.0007 & 0.0000 + 0.01 \end{bmatrix}$$

yielding the following down projected laser image

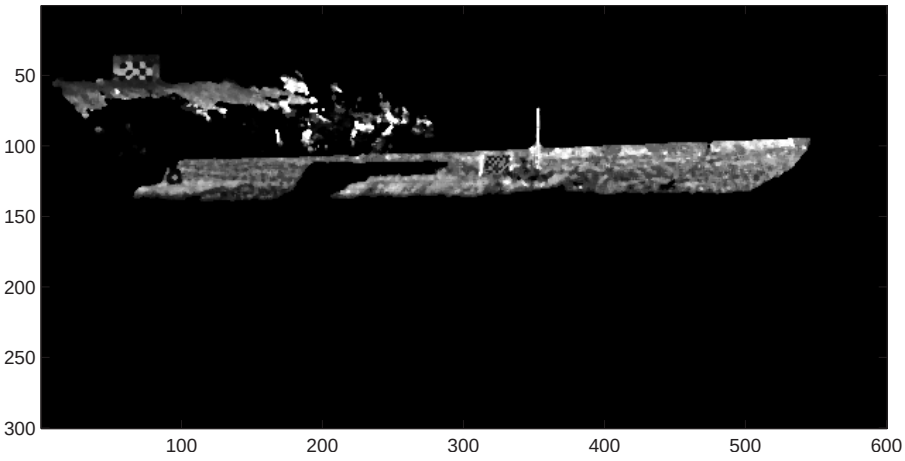

Figure 5.4: Laser image using $\mathbf{P}_{\text{perturbed2}}$

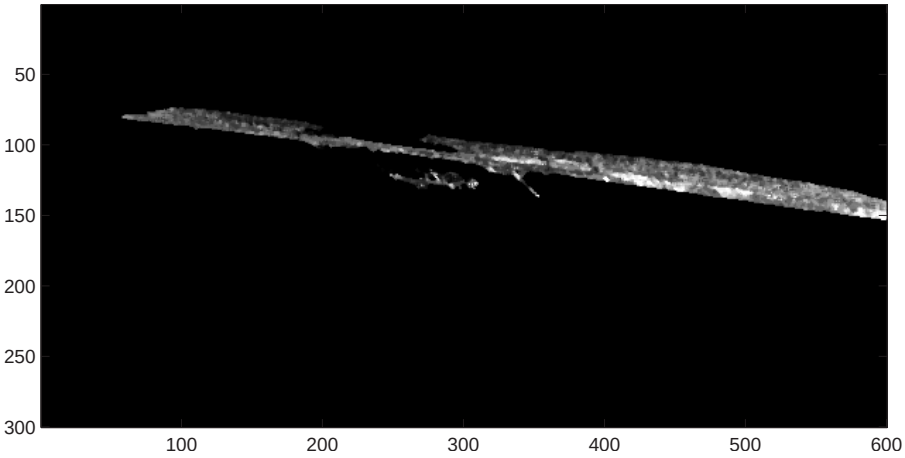

Figure 5.5: Iteration 1 for $\mathbf{P}_{\text{perturbed2}}$

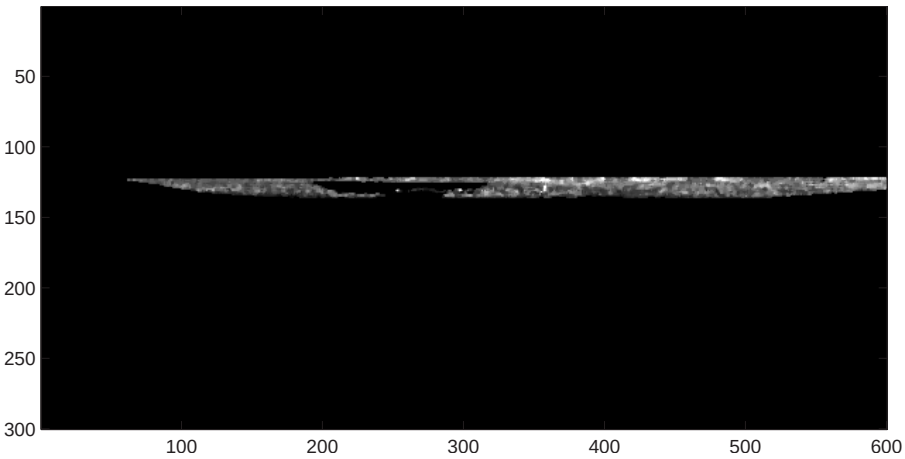

Figure 5.6: Iteration 2 for $\mathbf{P}_{\text{perturbed2}}$

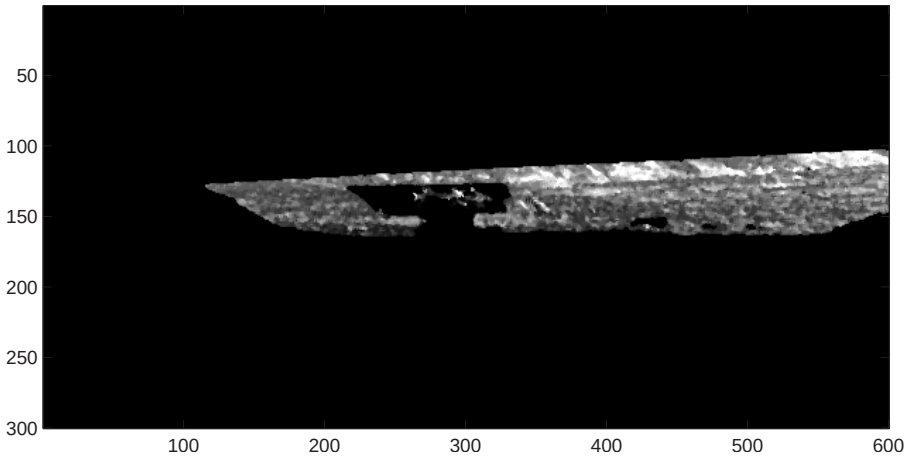

Figure 5.7: Iteration 3 for $\mathbf{P}_{\text{perturbed2}}$

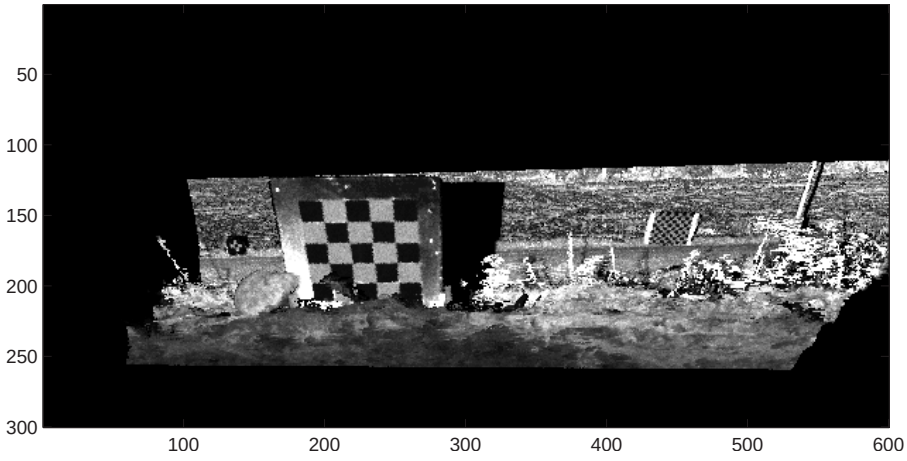

Figure 5.8: Iteration 4 for $\mathbf{P}_{\text{perturbed2}}$

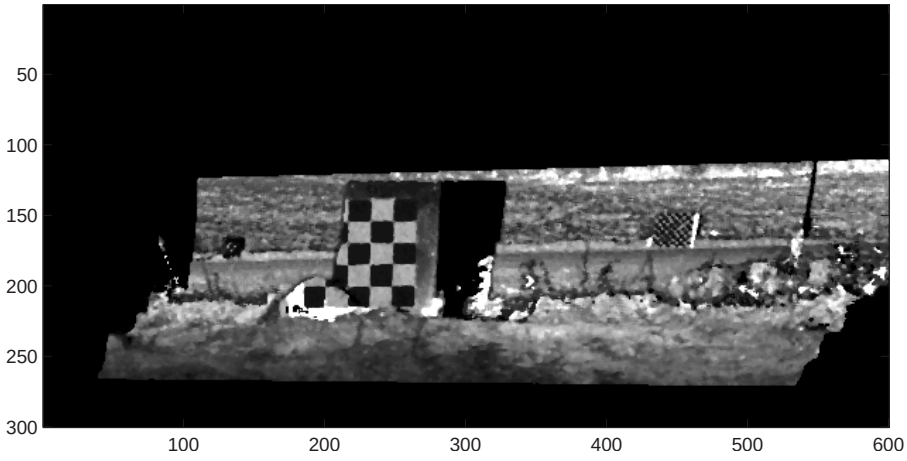

Figure 5.9: Iteration 5 for $\mathbf{P}_{\text{perturbed2}}$

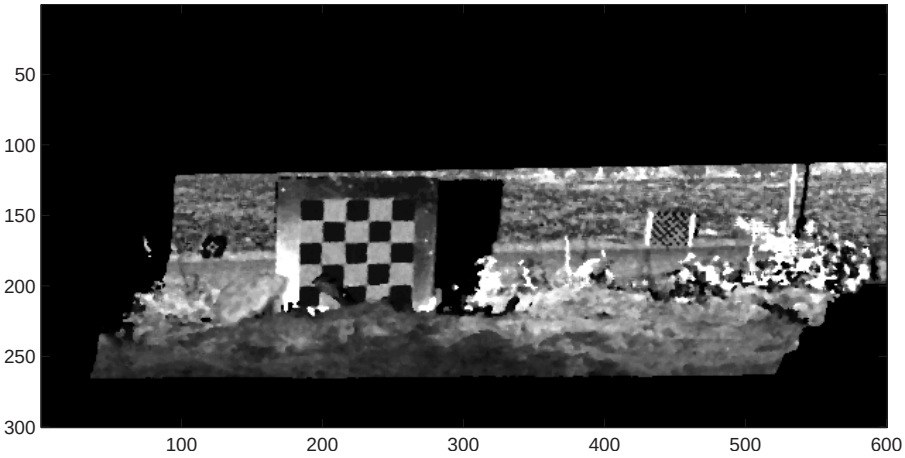

Figure 5.10: Iteration 6 for $\mathbf{P}_{\text{perturbed2}}$

6 Conclusions and recommendations

During this thesis work, we implemented a demonstration tool on how these two sensors can be fused for detection purposes. We have shown that this kind of fusion is possible to realize and the potential tremendously high. Since fusing the hyperspectral sensor with the laser was so successful, we believe that fusing the laser with another sensor, e.g., an IR-sensor, is highly possible. Hence, we can extract more intelligence from the scene and the chances to find targets in the scene will increase.

6.1 Recommendations for future work

This thesis is just a start in the fusion of different sensors. Since it is just for demonstration purposes, there are plenty of possible improvements that could be made. Stated below is a couple of suggestions and improvements that we think are possible from our experience and that will improve the kinds of fusion.

6.1.1 Data acquisition

When the data is to be acquired, there are some aspects that should be considered. For example, during this thesis I found out in the very end that the hyperspectral sensor was focused for two meters range, though we had objects lying as far as 100 meters away. Considering that, it is obvious that detecting objects far away was hard. Also, the sensors were not placed close enough to each other during the field-survey. This will result in the overlap region won't be as large as it could be, since the overlap region determines how many point correspondences that could be used for calibration purposes. As mentioned in section 4.1, having point correspondences well distributed along the image frame is crucial. But having them uniformly distributed along one image frame is not sufficient. They must also be uniformly distributed in a number of frames in order to make sure that our solution for the projection matrix is depth invariant, i.e. we can allow 3-D space coordinates to be spread out along the Z-axis. Hence, we must have reference points in varying depth and also, preferably, uniformly distributed along the image frame. This can be illustrated by considering figure 4.1. What we want is to have points spread out on the whole image frame and frames that can cover as much on the 3-D space Z-axis as we are measuring, i.e. if the most deep object is 90 meters away from the sensor, then we should have frames covering that depth to lock the projection matrix solution for that particular depth.

6.1.2 Data processing program

As mentioned in the beginning, the program we used for programming is Matlab. Matlab is a very good program for basic programming, but it is not as fast as we could hope for. When the program we designed performed the iteration

for optimizing and calibration of sensor data, it took as much as 6 minutes to do one loop. This processing time could be improved tremendously using different programming language such as C++, C, or Java, since all the operations used in our program are basically vector operations. The operation that took most processing time was the one looking up 3-D space coordinates for a certain pixel coordinate. For each coordinate, it has to run through all the pixel coordinates and return the corresponding spatail coordinates.

6.1.3 Calibration

The initial camera calibration is actually the most crucial step in the whole chain of events. Since the first camera calibration will yield the first guess of the projection matrix, which returns the down projection of the point cloud for corner point matching, it is extremely important that the point correspondences and the calibration itself are reliable.

As [4] mentions, taking pictures of a checkerboard from a few different orientiations will yield very good calibration results. However, in our case it is not interesting to take calibration images from different camera positions. In fact, it is not necessary at all. Taking images from one camera position is sufficient. Before getting deeper on that, let's answer the question why a camera calibration system needs to have checkerboard pattern from so many different orientations. Consider figure 6.1 below where a 3-D space coordinate is being mapped to a pixel.

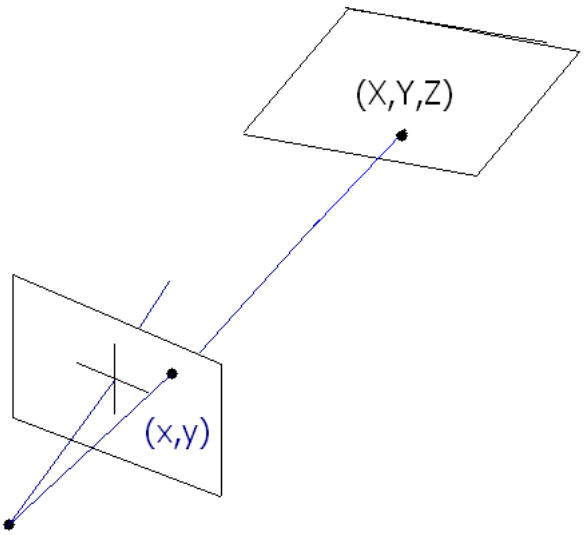

Figure 6.1: Linear transformation between 3-D space to image space

Suppose that the Z-coordinate is Z_0. If the Z-coordinate is moved closer or farther away from the image frame, i.e. $Z = Z_0 \pm \Delta Z$, it will still yield the same pixel coordinate. By taking multiple view images of the same object, we obtain the following situation

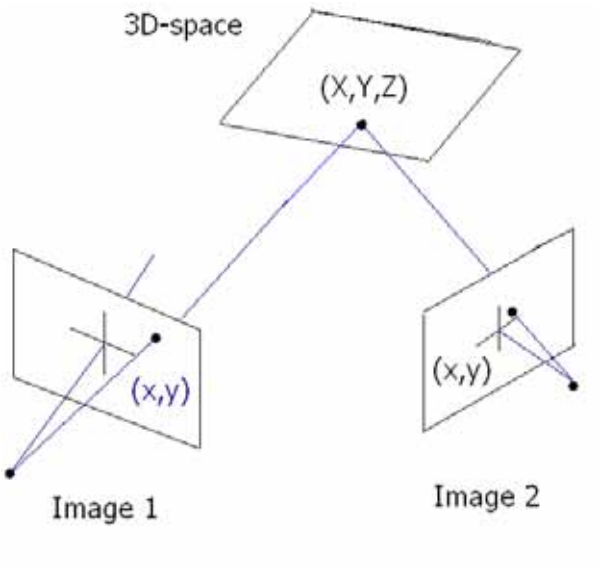

Figure 6.2: Simple triangulation

Now, there is a unique Z_0 for a given pixel coordinate. This is the only reason why mutliple view images are needed. For our sensors, however, it is not necessary, since the depth information is stored in the laser data. If we want to calibrate a sensor together with the laser, it has been shown here that it should be sufficient to use the algorithm provided here. But if we in the near future want to calibrate two other sensors that do not return any depth information such as IR cameras, we do need this triangulation scheme to obtain the depth information. Suppose we have a fixed setup between these two sensors. Taking multiple view images will not yield this triangulation, since the sensors mutual positions are the same. Instead, we need to have one of the sensors as the fixed one, e.g. the one with the smallest vision field. Then rotate the other sensor in different directions to obtain different views of the reference object will result in this triangulation. After getting sufficiently many views from the reference object, using calibration scheme proposed by [4] together with the Matlab toolbox provided by [15] should give satisfactory results.

In most of the litteratures and reports concerning camera calibration such as [14], [3], suggest that the projection matrix can be written as:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \end{bmatrix}$$

which basically states that p_{34} can be normalized to 1. This can have terrible consequences. It can be seen by inspection using (3.18) that p_{34} can be written as

$$\begin{aligned} p_{34} \quad = \quad & -\sin\theta\cos\psi - \cos\theta\sin\psi\sin\varphi \cdot X_0 - \sin\theta\sin\varphi \\ & +\sin\varphi\cos\theta\cos\psi \cdot Y_0 - \cos\varphi\cos\theta \cdot Z_0 \end{aligned}$$

Setting this element to 1 means that all these angles with the translation must be equal to one, which is not realistic since we do not know where the sensors are standing with respect to each other when the automatic calibration is being performed. Hence, if a similar camera calibration is to be performed, use instead the method described in section 3.1.5. To compute these long matrix multiplications, Matlab symbolic toolbox might come in handy. Using following commands, the above given equation can be generated:

```
syms phi theta psi sigmaXf sigmaYf gamma x0 y0 X0 Y0 Z0
Rx = [1 0 0; 0 cos(phi) sin(phi);0 -sin(phi) cos(phi)];
Ry = [cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0 cos(theta)];
Rz=[cos(psi) sin(psi) 0; -sin(psi) cos(psi) 0; 0 0 1]
Phi = [sigmaXf gamma x0; 0 sigmaYf y0; 0 0 1];
T = [1 0 0 -X0;0 1 0 -Y0; 0 0 1 -Z0];
P = Phi*Rx*Ry*Rz*T;
p34 = P(3,4);
```

6.1.4 More improvement measures

During this thesis we have used the mutual information between the hyperspectral image and the down projected laser image as a measure to analyse how good the mapping between these images is. But we feel that for further research in this field, other measures should be cooperated with the mutual information to get a more robust measure of the mapping.

Bibliography

[1] J. Ahlberg. *A Matlab Toolbox for Analysis of Multi/Hyperspectral Imagery*. Technical report FOI-R--1962--SE, FOI – Swedish Defence Research Agency, 2006.

[2] J. Ahlberg and I. Renhorn. *Multi- and Hyperspectral Target and Anomaly Detection*. Scientific report FOI-R--1526--SE, FOI – Swedish Defence Research Agency, 2004.

[3] S. Carlsson. *Geometric Computing in Image Analysis and Visualization*. Lecture notes, Department of Numerical Analysis and Computing Science, KTH, Stockholm, Sweden.

[4] Z. Zhang. *Flexible camera calibration by viewing a plane from unknown orientations*. Microsoft Research, Redmond, WA, USA.

[5] P. Kovesi. *Computer Vision*. Lecture notes, University of Western Australia, Perth, WA, Australia.

[6] S. Birchfield. *An Introduction to Projective Geometry (for computer vision)*. Stanford University, CA, USA.

[7] C. Harris and M. Stephens. "A combined edge and corner detector." *Proceedings of the 4th Alvey Vision Conference*, Manchester, UK, 1988, pp. 147–151.

[8] P. V. O'Neil. *Advanced Engineering Mathematics,* 5^{th} *Edition*. Brooks/ Cole Publishing Company, New York, 2005.

[9] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2004.

[10] P. E. Debevec. *Modeling and Rendering Architecture from Photographs*. Ph.D Thesis, University of California at Berkeley, CA, USA.

[11] B. P. Lathi. *Modern digital and analog communication systems, third edition*. Oxford University Press, New York. 2004.

[12] Matlab Optimization toolbox. http://www.mathworks.com

[13] M. R. Bax. *Real-time lens distortion correction: 3-D video graphics cards are good for more than games*. Stanford University, CA, USA.

[14] M. M. Seger. *Lite om kamerageometri*. Linköping University, Sweden, 2005.

[15] D. Scaramuzza. *Omnidirectional Camera Calibration Toolbox for Matlab*. Swiss Federal Institute of Technology, Lausanne, Switzerland.

[16] A. Wiklund. *Registration approaches for noisy 3-D data representing natural scenes*. Scientific report FOI-R--1994--SE, FOI – Swedish Defence Research Agency, 2006.

[17] O. Steinvall et al. *Grindad Avbildning - fördjupad studie*. Scientific report FOI-R--0991--SE, FOI – Swedish Defence Research Agency, 2003.

[18] T. Chevalier et al., *Årsrapport 3D-laser 2005*. User report FOI-R--1807--SE, FOI – Swedish Defence Research Agency, 2005.

[19] G. Sparr. *Linjär algebra*. Studentlitteratur, Lund, Sweden, 2003.

[20] G. Woods. *Digital Image Processing*. Addison Wesley, Singapore, 1993.

[21] A. Zisserman. *Single View and Two-View Geometry*. Robotics Research Group, University of Oxford, UK.

[22] T. Chevalier. *Mätrapport Samverkande sensorer Ströplahult Norra maj 2006*. Internal report FOI-D--0261--SE, FOI – Swedish Defence Research Agency, 2006.

[23] U. M. Ascher. *Lecture notes in Numerical Optimization*. University of British Columbia, Canada.
http://www.cs.ubc.ca/spider/ascher/542/chap11.pdf

[24] InnovMETRIC software. http://www.innovmetric.com

[25] C. Freyhult. *Visualisation and methods to support interacting 3-D laser radar and hyperspectral sensors*. Technical report to be published, Swedish Defence Research Agency (FOI), 2006.

[26] H. Ojanen. *Automatic Correction of Lens Distortion by Using Digital Image Processing*. Rutgers, NJ, USA, 1999.
http://www.math.rutgers.edu/ ojanen/

A Harris corner detector

A corner is defined as the intersection of two edges. An interest point is a point in an image which has a well-defined position. This means that an interest point can be a corner but also, for example, an isolated point of local intensity maximum or minimum, line endings, or a point on a curve where the curvature is locally maximum. As a consequence, if only corners are to be detected, it is necessary to do a local analysis of detected interest points to determine which of these are real corners.

Many proposed corner detectors are not usually very robust and often require large redundancies introduced to prevent the effect of individual errors from dominating the recognition task. However, an approach proposed by Harris and Stephens [7] has been proved to be very computationally efficient and robust.

Let $f(x, y)$ denote a grayscale 2-dimensional image. $\nabla f(x, y)$ is the local gradient of the image and $\nabla f(x, y) \cdot n$ is the gradient along the direction n. At a corner as we rotate n through all possible values, we should find two directions where the gradient goes through a maximum:

$$C_n = \frac{||\nabla f(x, y) \cdot n||}{|n|}$$

$$C_n^2 = \frac{n^T \nabla f \nabla f^T n}{n^T n} \tag{A.1}$$

where

$$\nabla f \nabla f^T = Q = \begin{bmatrix} (\frac{\partial f}{\partial x})^2 & (\frac{\partial f}{\partial x})(\frac{\partial f}{\partial y}) \\ (\frac{\partial f}{\partial x})(\frac{\partial f}{\partial y}) & (\frac{\partial f}{\partial y})^2 \end{bmatrix}$$

where Q is a smoothing operator.

As in edge detection, we require that noise to be eliminated from the images. This can be achieved by convolution with a Gaussian kernel (filter) or any appropriate alternative method. Using this expression on (A.1), we obtain

$$C_n^2 = \frac{n^T Q\, n}{n^T n}$$

which is the Rayleigh quotient and therefore is subject to the following bounds:

$$\lambda_1 \leq \frac{n^T Q\, n}{n^T n} \leq \lambda_2 \tag{A.2}$$

where λ_1 and λ_2 are the smallest and largest eigenvalues of the matrix Q. This means as n is varied through all possible values, C_n^2 is restricted within these eigenvalue bounds. Based on the magnitudes of the eigenvalues, the following inferences can be made on this argument:

1. If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ then there are no features of interest at this pixel.
2. If $\lambda_1 \approx 0$ and λ_2 is some large positive value, then an edge is found.

3. If λ_1 and λ_2 are both large, distinct positive values, then a corner is found.

Implementation of this routine usually look for a threshold in the following function:

$$M_c = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$
$$M_c = \det(Q) - \kappa\,\mathrm{trace}^2(Q)$$

Therefore, this implementation does not have to actually compute the eigenvalue decomposition of the matrix A, instead it is sufficient to evaluate the determinant and trace of Q to find corners. It is also common practice to set this tuneable parameter κ to values in the range $0.04 - 0.15$, which determines how 'edge-phobic' the algorithm is.

FOI is an assignment-based authority under the Ministry of Defence. The core activities are research, method and technology development, as well as studies for the use of defence and security. The organization employs around 1350 people of whom around 950 are researchers. This makes FOI the largest research institute in Sweden. FOI provides its customers with leading expertise in a large number of fields such as security-policy studies and analyses in defence and security, assessment of different types of threats, systems for control and management of crises, protection against and management of hazardous substances, IT-security an the potential of new sensors.

FOI
Swedish Defence Research Agency
P.O. Box 1165
SE-581 11 LINKÖPING

Tel: +46 13 37 80 00
Fax: +46 13 37 81 00

www.foi.se