dedication (optional)

Summary

Write your summary here...

Preface

This project report presents the work for the mandatory course *TTK4551*, taken during the fall of 2017. The course counts for 7.5 credit points of the two-year master's programme in cybernetics and robotics.

Table of Contents

List of Tables

List of Figures

Abbreviations

ADC	=	Analog-to-Digital Converter
AIS	=	Automatic Identification System
ASV	=	Autonomous Surface Vehicle
CNN	=	Convolutional Neural Net
CCD	=	Charge Coupled Device
INS	=	Inertial Navigation System
GNSS	=	Global Navigation Satellite System
FOV	=	Field Of View
RPS	=	Rotations Per Second
ROS	=	Robot Operating System
HDMI	=	High Definition Multimedia Interface
OS	=	Operating System
USB	=	Universal Serial Bus
IP	=	Internet Protocol
TCP	=	Transmission Control Protocol
NMEA	=	National Marine Electronics Association
CPU	=	Central Processing Unit
GB	=	Gigabyte
RAM	=	Random Access Memory
HDD	=	Hard Disk Drive
FPGA	=	Field Programmable Gate Array

Chapter 1

Introduction

1.1 Problem formulation

Autonomous surface vehicles (ASVs) operating in urban environments, including ferries, will need slightly different exteroceptive sensors than ASVs operating in the open sea. A lidar with a range of 100 meters may be more appropriate than a maritime radar with range of several kilometers. Furthermore, the complexity of the environment means that the rich information from optical cameras will be more important. In order to build a coherent world image, which, e.g., collision avoidance decisions can be based on, the data from these sensors must be fused, together with data from interoceptive sensor systems such as an inertial navigation system (INS).

The goal of the specialization project is to prepare the processing pipeline leading to camera-lidar fusion. A comparison of detections of interesting objects in the camera data with interesting objects in the lidar data will be performed, and suitable detection models for camera and lidar will be analyzed .

Hvis tid

The following subtasks are proposed for the project:

1. Installation of Velodyne lidar together with camera, and recording of time-synchronized data from both sensors.
2. Calibration of the sensors, including specification of world-frame-to-sensor-frame measurement models for both sensors.
3. Implementation of a detector based on a convolutional neural network (CNN) for the detection of boats in camera data.
4. Implementation of a detector based on intensity of reflected signal strength for the lidar.
5. Analysis of the extent to which lidar detections correspond to camera detections and vice versa.
6. Analysis of suitable detection models for lidar and camera.

1.2 Report outline

Describe what each chapter (section) presents

Chapter 2

Theory

2.1 Sensor Fusion

Robots and autonomous vehicles, be it a car or a ship, operate in unstructured environments, environments that are unpredictable. While robots working on a assembly line work in very structured, predictable environments, a vessel attempting to atonomously navigate a busy shipping lane has to rely on sensor inputs in order to make sense of its environment, and navigate safely. In sensing its environment, a vessel might use various sensors, such as radar(s), cameras, sonars or laser range finders. The different sensors typically give a incomplete and imperfect view of the surrounding world. Sensor data fusion is the process of combining such mutually complementary sensor information in such a way that a better understanding of the surroundings can be achieved [1]. The U.S. Joint Directors of Laboratories (JDL) Data Fusion Working Group developed a process model in 1985 for characterizing hierarchical levels of fusion processing, categorized fusion functions, and candidate algorithm approaches, and is the most widely used system for categorizing data fusion-related functions [2]. The model is illustrated in figure 2.1.

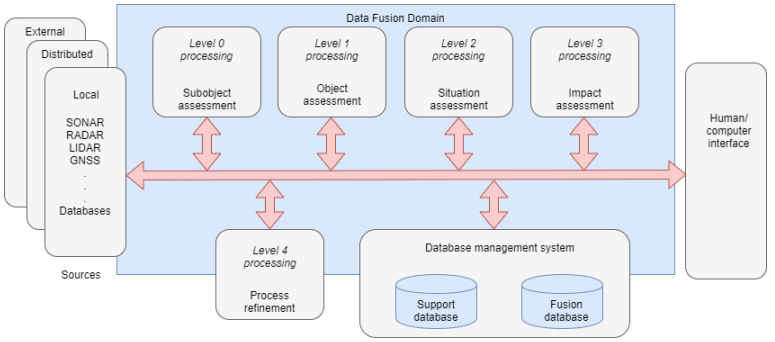


Figure 2.1: The Joint Directors of Laboratories data fusion model.

The JDL model defines the fusion levels as follows:

- Level 0: *Subobject assessment*. Preconditioning data to correct biases, perform spatial and temporal alignment, and standardize inputs.
- Level 1: *Object assessment*. Association of data (including products of prior fusion) to estimate an object or entitys position, kinematics, or attributes (including identity).
- Level 2: *Situation assessment*. Aggregation of objects/events to perform relational analysis and estimation of their relationships in the context of the operational environment.
- Level 3: *Impact assessment*. Projection of the current situation to perform event prediction, threat intent estimation, own force vulnerability, and consequence analysis.
- Level 4: *Process refinement*. Evaluation of the ongoing fusion process to provide user advisories and adaptive fusion control or to request additional sensor/source data.

The models different levels do not represent a process flow, but rather provides a structured and integrated view on the complete functional chain from distributed sensors, data bases, and human reports to the users and their option to act including various feedback loops at different levels. Although the model was developed with military applications in mind, it remains valid in civilian applications. Koch [1] identifies two characteristic features of sensor data fusion:

1. The available sensor data and context knowledge to be fused typically provide incomplete and imperfect pieces of information. The reasons for this are manifold, and are unavoidable in real-world applications. For dealing with these deficiencies, sophisticated mathematical methodologies and reasoning formalisms are applied.
2. Sensor data fusion is closely related to the practical design of surveillance and reconnaissance components for information systems. In implementing fundamental theoretical concepts, a systematic way of finding reasonable compromises between mathematical exactness and pragmatic realization issues, as well as suitable approximation methodologies are therefore inevitable. System aspects such as robustness and reliability even in case of unforeseeable nuisance phenomena, priority management, and graceful degradation are of particular importance in view of practicability.

This project aims to prepare the processing pipeline leading to fusion of measurements from a lidar with measurements from a camera. A lidar provides excellent range information, but has limits with regard to object recognition. A camera, on the other hand, is good for object recognition but has limits to the high resolution range information. The two sensors exhibit complementary properties, and combining (fusing) the measurements from both sensors can achieve more specific inferences about the surroundings than using a single, independent sensor. In order to associate measurements from the lidar and the camera to the external world, fusion with an INS is necessary, to be able to associate the (moving) coordinate frames of the sensors with a common world frame. Moreover, to be able to associate measurements from one sensor with the other, they need to be synchronized in time (temporally calibrated).

2.2 Computer Vision

Computer vision is a field of engineering and science concerned with extracting useful information from images. This has proved to be a challenging task, and it is still today an active field of research. Visual data is very complex, and the same object represented by two different images could be perceived very differently by a computer based on variations such as changes in illumination, partial occlusion of objects, changes in orientation, deformation and so on. Such variations are illustrated in figure 2.2. Despite these challenges, the recent advancements made within the field of deep learning, and deep convolutional neural nets in particular, has significantly improved the ability of computers to recognize objects in images. The ImageNet Large Scale Visual Recognition Challenge has been run annually since 2010, and is a benchmark in object category classification and detection on hundreds of object categories and millions of images [3]. The results of the top 5 classification errors from the challenges up until 2016 is shown in figure 2.3.

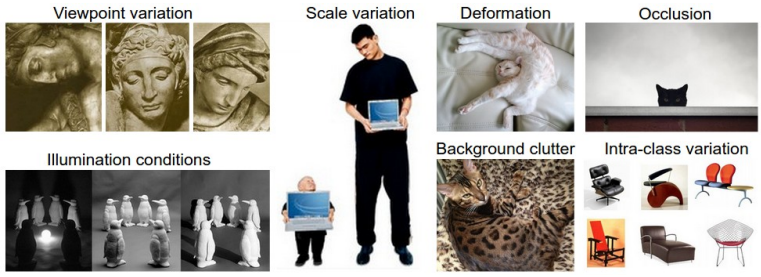


Figure 2.2: Challenges related to object classification in images.[1]

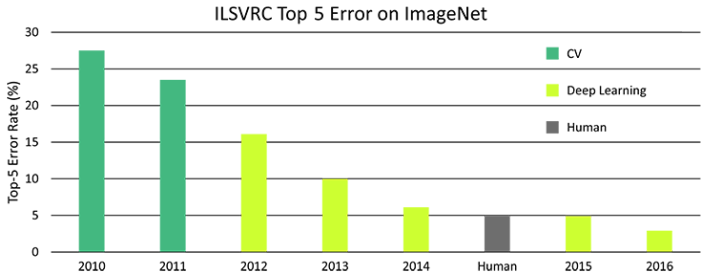


Figure 2.3: ImageNet top 5 errors over time.[2]

As seen in figure 2.3, deep learning algorithms dominate the field of object category classification and detection, even outperforming humans with state-of-the-art deep convo-

[1] Image courtesy: http://cs231n.github.io/classification/

[2] Image courtesy: https://www.dsiac.org/resources/journals/dsiac/winter-2017-volume-4-number-1/real-time-situ-intelligent-video-analytics

lutional neural nets. This section will introduce some necessary basic image processing operations, and give a brief introduction to convolutional neural nets used for image object classification and detection.

2.2.1 Spatial image filtering

The term spatial in spatial image filtering refer to the image plane itself, and spatial image processing involves direct manipulation of the pixels themselves in the image, in contrast to other methods operating in a transform domain such as the frequency plane. A spatial filter consists of a *neighborhood* (also called a mask, kernel, template or window) (typically rectangular), and a *predefined operation* that is performed on the pixels in the image encompassed by the neighborhood [4]. Filtering the image creates a new pixel with coordinates equal to the center of the neighborhood, with value given by the predefined operation performed on the pixels in the neighborhood. Spatial filtering of an image of size $M \times N$ with a filter of size $m \times n$ is given by

$$g(x,y) = w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \tag{2.1}$$

where $a = \frac{m-1}{2}$ and $b = \frac{n-1}{2}$, $w(s,t)$ is the filter mask, with its center at $w(0,0)$ aligned with the pixel at location (x,y). $g(x,y)$ is the response of the filter at location (x,y). $*$ denotes the convolution operator. x and y are varied such that each pixel in the image is visited. In other words, spatial filtering of an image with a mask of size $m \times n$ corresponds to convolving the image with the mask. The filter and the corresponding image pixels for a given (x,y) are illustrated in figure 2.4. The filter mask used in this illustration is 3×3 pixels.

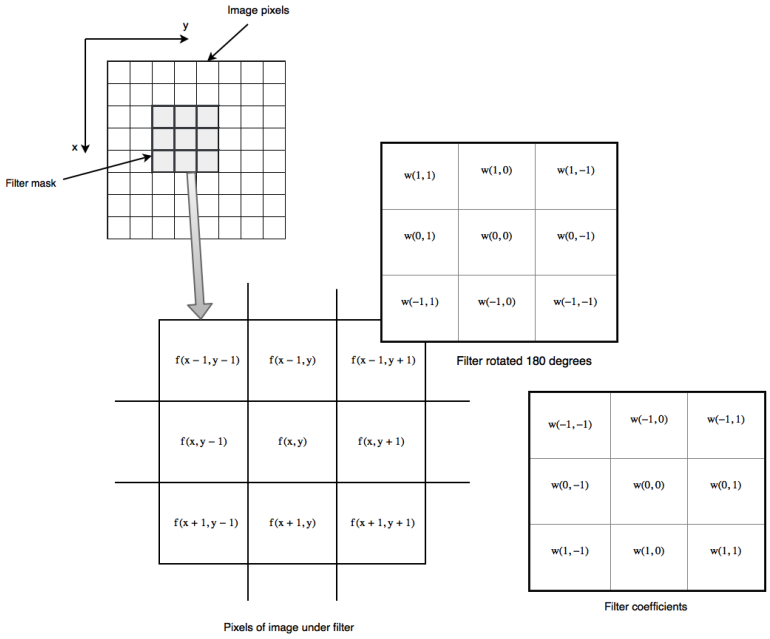


Figure 2.4

Convolution between the image and the filter mask corresponds to correlating the image and the filter mask rotated 180 degrees.

2.2.2 Convolutional Neural Nets for Object Recognition

What is a neural net, brief description of convolutional neural nets.

2.3 Target tracking

Necessary? Will not implement tracking at this stage anyway.

2.3.1 Clustering

2.3.2 PDAF (or other tracking method)

Chapter 3

Sensor Models

3.1 Image Formation

Camera systems have found a wide array of applications since its invention, and are today widely used in industry. Application areas range from surveillance (CCTV), to image acquisition in scientific exploration of the earth, oceans and space. In recent years, there has been a drive towards utilizing cameras for automatic object detection and recognition, made possible through advances in computing power and algorithms. There is a drive towards more and more autonomous operation of vehicles, and camera systems play an integral part in the sensing of the environment necessary for safe autonomous operation. This section presents how an image is formed in a digital camera, and how to relate image coordinates to world coordinates through transformations. At the end of this section a methodology widely used for camera calibration is presented. The theory presented in this section is by no means exhaustive, for a more in-depth treatment of the theory behind image formation see [4]–[6].

3.1.1 The Pinhole Camera Model

Pinhole camera, thin lens model, thick lens model, refraction, snell's law, paraxial (or first-order) geometric optics, depth of field/depth of focus, field of view, abberations...

The pinhole camera model was first proposed by Brunelleschi in the early fifteenth century [5]. Consider a rectangular box with a translucent plate at one end. At the opposite end there is a small pinhole, small enough so that exactly one ray of light would pass through the translucent plate, the pinhole, and some scene point. The distance between the pinhole and the translucent plate is called the *focal length*, denoted f. This is of course impossible in reality, as the pinhole will have a finite, though small size. As such, each point in the image plane will collect light from a cone of rays. A real camera will also be equipped with lenses, which complicates the simple model of rays passing through a hole. Still, the pinhole camera model is mathematically convenient, and often provides

an acceptable approximation of the imaging process. The pinhole perspective projection model is illustrated in figure 3.1.

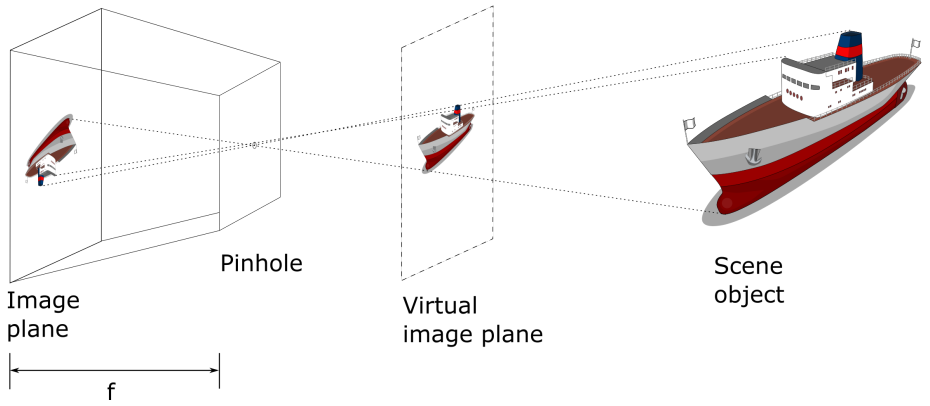
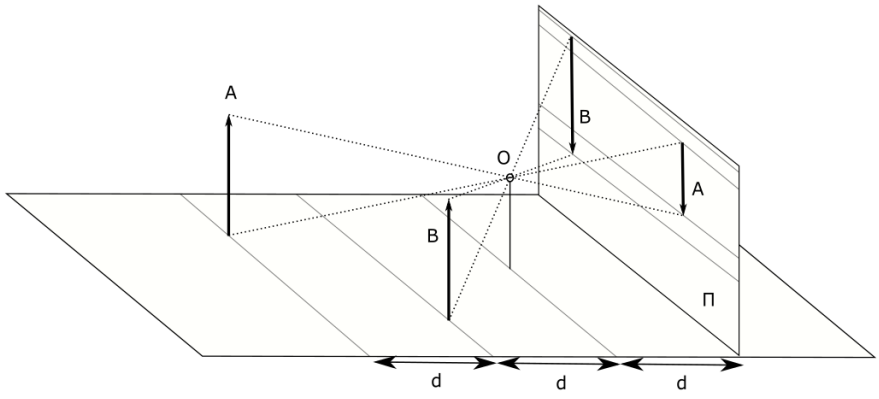
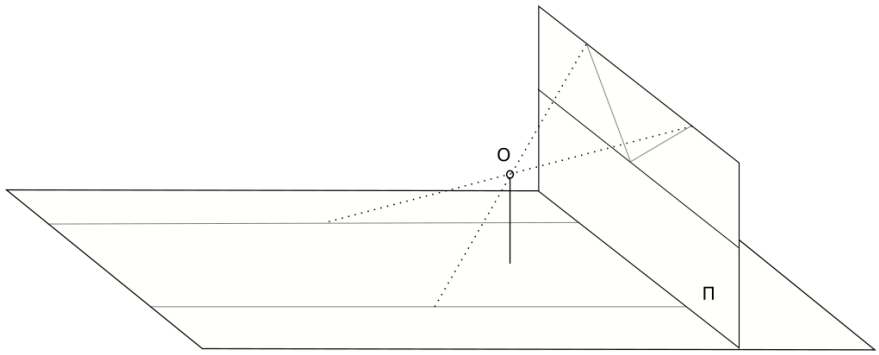


Figure 3.1: The pinhole camera model

The pinhole perspective creates inverted images, so it is sometimes convenient to consider a virtual image plane lying in front of the pinhole, at the same distance from the pinhole as the actual image plane. The virtual image is not inverted, but otherwise identical to the actual image. There are some effects of this perspective projection which deserves some consideration. One such effect is the fact that the apparent size of objects depend on their distance from the pinhole. Consider two objects which are identical. If one object is placed farther away from the pinhole relative to the other, it will appear smaller than the closer object. Another effect of perspective projection is parallel lines in appearing to converge on a horizon line in the image plane. These effects are illustrated in figure 3.2.



(a) Close objects appear larger than distant ones.



(b) Parallel lines appear to converge on a horizon line.

Figure 3.2: Perspective effects.

3.1.2 Digital Image Aquisition

A digital image is formed by electromagnetic radiation illuminating a sensor arrangement. The electromagnetic radiation gathered by the sensors typically falls within the spectrum

of visible light (with a wavelength approximately between 400 and 700 nanometers), but can also be in the infrared or ultraviolet spectrum, or in the X-ray spectrum as is the case for X-ray detectors in medical applications. The sensed wavelength depends on the application, and in the scope of this project only the visible spectrum is considered. For simplicity, the electromagnetic radiation falling within the visible spectrum will simply be adressed as *light*. The images are generated by the combination of an illumination source and the reflection or absorption of energy from that source by the elements of the scene being imaged. A digital camera is a *passive* sensor, as in it detects input generated by the physical environment. Incoming energy is transformed into a voltage by the combination of input electrical power and sensor material that is responsive to the energy being detected. The sensor outputs a voltage waveform as a response, and this voltage is sampled and quantized, producing a digital value.

Individual sensors are arranged in a 2D CCD array. The response of each individual sensor is proportional to the integral of the light energy projected onto the sensor [4]. Color images are produced by placing a color filter consisting of red, blue and green in front of the sensor array. The typical color filter setup, known as a Bayer filter [7] is shown in figure 3.3.

(a) The Bayer filter. **(b)** Profile of sensor.

Figure 3.3: The Bayer filter overlaid a pixel array.[1]

The voltage output by each pixel may be read and quantized by a ADC producing a digital value, usually in the range 0-255 (8-bit).

3.1.3 Intrinsic and Extrinsic Parameters

The world and camera coordinate systems are related by a set of physical parameters. These parameters can be separated into *intrinsic* parameters, which relates the image coordinate system to the normalized coordinate system presented in figure 3.4, and *extrinsic* parameters, which relate the cameras coordinate system to a fixed world coordinate system and specify its position and orientation in space.

[1] Images courtesy: https://en.wikipedia.org/wiki/Bayer_filter

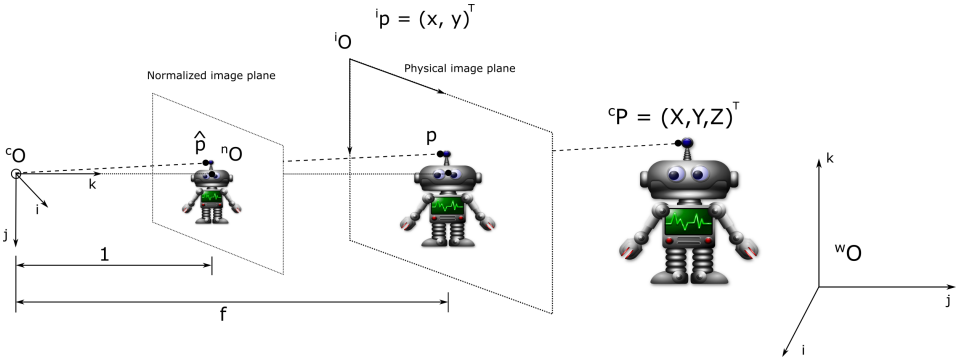


Figure 3.4: Coordinate systems

In figure 3.4, ${}^{n}O$ denotes the origin of the normalized image plane, ${}^{i}O$ is the origin of the physical image plane, ${}^{w}O$ is the origin of the world frame, and ${}^{c}O$ is the origin of the camera 3D frame. Since ${}^{c}O$, ${}^{c}\hat{p}$ and ${}^{c}P$ in figure 3.4 are collinear, we have that $\overrightarrow{{}^{c}O\hat{p}} = \lambda \overrightarrow{{}^{c}OP}$ for some constant λ such that

$$\begin{cases} \hat{x} = \lambda X \\ \hat{y} = \lambda Y \iff \lambda = \frac{\hat{x}}{X} = \frac{\hat{y}}{Y} = \frac{1}{Z} \\ 1 = \lambda Z \end{cases} \tag{3.1}$$

which gives the following relation between the normalized plane coordinates ${}^{n}\hat{p}$ and the coordinates of the point ${}^{c}P$

$$\begin{cases} \hat{x} = \frac{X}{Z} \\ \hat{y} = \frac{Y}{Z} \end{cases} \tag{3.2}$$

Homogeneous Coordinates and Rigid Transformations

Homogeneous coordinates are convenient for representing geometric transformations by a matrix product. Consider a point P in some right-handed coordinate frame with origin O and the x, y and z axis given by the unit vectors $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ respectively. The point can be specified by

$$\overrightarrow{OP} = X\mathbf{i} + Y\mathbf{j} + Z\mathbf{k} \tag{3.3}$$

The nonhomogeneous coordinate vector P is the vector $(X, Y, Z)^T$ in $\mathbb{R}^3$, while its homogeneous counterpart is the vector $(X, Y, Z, 1)^T$ in $\mathbb{R}^4$. The change of coordinates between two (arbitrary) Euclidean coordinate systems a and b can be represented by a rotation matrix ${}^{a}\mathcal{R}_b$ and a translation vector $\mathbf{t}$ in $\mathbb{R}^3$ as

$${}^{a}\mathbf{P} = {}^{a}\mathcal{R}_b^b\mathbf{P} + \mathbf{t} \tag{3.4}$$

The notation used for the rotation matrix describes the direction of the transformation, where the superscript denotes the resulting coordinate frame, while the subscript denotes

the coordinate frame the vector to be transformed is represented in, as ${}^{\text{to}}\mathcal{R}_{\text{from}}$. Using homogeneous coordinates, the same transformation can be written as

$$ {}^{a}\mathbf{P} = \mathcal{T}\,{}^{b}\mathbf{P}, \quad \text{where} \quad \mathcal{T} = \begin{bmatrix} {}^{a}\mathcal{R}_{b} & \mathbf{t} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \tag{3.5} $$

where ${}^{a}\mathbf{P}$ and ${}^{b}\mathbf{P}$ are now vectors in $\mathbb{R}^4$. There are several ways to parameterize the rotation matrix $\mathcal{R}$, such as by Euler angles or quaternions. There are many textbooks which go into great detail on parameterizations and properties of rotation matrices, some examples being [8], [9] and [10]. Going into such details is beyond the scope of this text.

Intrinsic Parameters

Using homogeneous coordinates, equation 3.2 can be rewritten as

$$ {}^{n}\hat{\mathbf{p}} = \frac{1}{Z}\begin{bmatrix}\mathbf{I}^{3\times 3} & \mathbf{0}^{3\times 1}\end{bmatrix}{}^{c}\mathbf{P} \tag{3.6} $$

where $\mathbf{I}^{3\times 3}$ is the 3-by-3 identity matrix, and ${}^{n}\hat{\mathbf{p}} \triangleq \begin{bmatrix}\hat{x} & \hat{y} & 1\end{bmatrix}^{T}$ is the homogeneous coordinate vector of the projection $\hat{p}$ of the point P into the normalized image plane, and ${}^{c}\mathbf{P}$ is the homogeneous coordinate vector of the point P in the camera coordinate frame. The following section relates the homogeneous coordinates of the normalized image plane to those of the physical image plane. The physical retina of a camera is located at a distance $f \neq 1$ from the pinhole (this assumes that the camera is focused at infinity, so that the physical distance from the pinhole to the image plane equals the focal length), and the physical image point p is usually expressed in pixel units. In addition to this, pixels may be rectangular instead of square, introducing two additional scale parameters k and l, such that

$$ \begin{cases} x = kf\frac{X}{Z} = kf\hat{x} \\ y = lf\frac{Y}{Z} = kf\hat{y} \end{cases} \tag{3.7} $$

We have that f represents a distance, in meters say, and a pixel has dimensions $\frac{1}{k} \times \frac{1}{l}$, where k and l are expressed in $\frac{\text{pixel}}{m}$. The parameters can be replaced by a magnification $\alpha = kf$ and $\beta = lf$ expressed in pixels. In general the origin of the physical image plane is in a corner c of the physical image plane (often the upper left corner, but this may vary) and not at it's center, and the center of the CCD matrix typically does not coincide with the image center c_0. This adds two more parameters x_0 and y_0 that define the position of c_0 in pixel units in the physical image coordinate system. Equation 3.7 is then replaced by

$$ \begin{cases} x = \alpha\hat{x} + x_0 \\ y = \beta\hat{y} + y_0 \end{cases} \tag{3.8} $$

This assumes that the angle between the image axes equals exactly 90 degrees. In reality, however, the image plane axes may be skewed, such that the angle between the image plane axes is $\theta \neq 90^\circ$. In this case, equation 3.8 is replaced by

$$ \begin{cases} x = \alpha\hat{x} - \alpha\cot\theta\hat{y} + x_0 \\ y = \frac{\beta}{\sin\theta}\hat{y} + y_0 \end{cases} \tag{3.9} $$

In matrix form, equation 3.9 can be written as

$$\mathbf{p} = \mathcal{K}\hat{\mathbf{p}}, \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathcal{K} \stackrel{\text{def}}{=} \begin{bmatrix} \alpha & -\alpha\cot\theta & x_0 \\ 0 & \frac{\beta}{\sin\theta} & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

The matrix $\mathcal{K}$ is called the *calibration matrix* of the camera. Putting the equations 3.6 and 3.10 together, we obtain

$${}^{i}\mathbf{p} = \frac{1}{Z}\mathcal{K}\begin{bmatrix}\mathbf{I}^{3\times3} & \mathbf{0}^{3\times1}\end{bmatrix}{}^{c}\mathbf{P} = \frac{1}{Z}\mathcal{M}{}^{c}\mathbf{P}, \quad \text{where} \quad \mathcal{M} \stackrel{\text{def}}{=} \begin{bmatrix}\mathcal{K} & \mathbf{0}^{3\times1}\end{bmatrix} \tag{3.11}$$

The parameters α, β, θ, x_0 and y_0 are called the *intrinsic parameters* of the camera, typically obtained through a calibration procedure like the one described in section A.1.

Extrinsic Parameters

Describe relation between camera coordinate frame and (some) world frame. Equation 3.11 is written in a coordinate frame rigidly attached to the camera. However, for applications like autonomous vehicles, where the vehicle itself moves in some inertial coordinate system, we can express equation 3.11 in some world coordinate system using a rigid homogeneous transformation. The change of coordinates between the camera frame and the world frame can be expressed as presented in equation 3.5:

$${}^{c}\mathbf{P} = \begin{bmatrix}\mathcal{R} & \mathbf{t} \\ \mathbf{0}^{T} & 1\end{bmatrix}{}^{w}\mathbf{P} \tag{3.12}$$

Here, ${}^{w}\mathbf{P}$ represents the point P in some world coordinate frame w. Substituting this into equation 3.11 yields

$${}^{i}\mathbf{p} = \frac{1}{Z}\mathcal{M}{}^{w}\mathbf{P}, \quad \text{where} \quad \mathcal{M} = \mathcal{K}\begin{bmatrix}\mathcal{R} & \mathbf{t}\end{bmatrix} \tag{3.13}$$

Knowing $\mathcal{M}$ determines the position of the camera's optical center in the world coordinate frame w. A rotation matrix $\mathcal{R}$ is defined by three independent parameters (such as Euler angles), and together with the translation vector $\mathbf{t}$ defines six *extrinsic parameters* that define the position and orientation of the camera relative to the world coordinate frame. The depth Z in equation 3.13 is dependent on $\mathcal{M}$ and ${}^{w}\mathbf{P}$, which can be seen directly from the equation. If we denote the three rows of $\mathcal{M}$ as $\mathbf{m}_1^T$, $\mathbf{m}_2^t$ and $\mathbf{m}_3^T$, it follows that $Z = \mathbf{m}_1^T \cdot {}^{w}\mathbf{P}$. It's important to note that the matrix $\mathcal{M}$ is only defined up to scale, and multiplying $\mathcal{M}$ by a arbitrary constant $\lambda \neq 0$ does not change the resulting image coordinates since

$$\begin{cases} {}^{i}x = \frac{\mathbf{m}_1^T \cdot {}^{w}\mathbf{P}}{\mathbf{m}_3^T \cdot {}^{w}\mathbf{P}} \\ {}^{i}y = \frac{\mathbf{m}_2^T \cdot {}^{w}\mathbf{P}}{\mathbf{m}_3^T \cdot {}^{w}\mathbf{P}} \end{cases} \tag{3.14}$$

and the constant λ cancels out. The perspective projection matrix $\mathcal{M}$ can be written explicitly as a function of its five intrinsic parameters, the rows $\mathbf{r}_1^T$, $\mathbf{r}_2^T$ and $\mathbf{r}_3^T$ of the rotation

matrix $\mathcal{R}$ and the coordinates of of the vector $\mathbf{t} = \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T$ as

$$\mathcal{M} = \begin{bmatrix} \alpha \mathbf{r_1} - \alpha \cot\theta \mathbf{r}_2^T + x_0 \mathbf{r}_3^T & \alpha t_1 - \alpha \cot\theta t_2 + x_0 t_3 \\ \frac{\beta}{\sin\theta} \mathbf{r}_2^T + y_0 \mathbf{r}_3^T & \frac{\beta}{\sin\theta} t_2 + y_0 t_3 \\ \mathbf{r}_3^T & t_3 \end{bmatrix} \tag{3.15}$$

The intrinsic parameters are typically found through a calibration routine, while the extrinsic parameters depend on the location of the camera frame relative to the world frame the user chooses. A detailed explanation of camera calibration can be found in appendix A.1.

3.2 Lidar

A lidar is an *active* electro-optical sensor that sends out a laser pulse, and subsequently measures the parameters of the return signal bounced off some object. The lidar is known under several names, some examples being ladar, lidar, LIDAR, LADAR or laser radar. The term lidar is the most common, and will be used throughout this report. Lidars have seen a growing range of applications in recent years, with the current drive towards autonomous, driverless cars as a significant driving factor. In a lidar, a waveform generator generates a laser waveform. Depending on the type of lidar, the setup can include a single laser or a master oscillator with multiple lasers or laser amplifiers. There are many types of lidars, however this report focuses on a particular type of 3D lidar with a rotating detector array, which measures azimuth and elevation angle, and range. An example of such a lidar is shown in figure 3.5.

The measurement is performed by the laser pulse being guided through transmit optics, traversing some medium, typically atmosphere, to a target. The laser pulse bounces off the target, and traverses the media again until receive optics captures the reflected pulse, guiding it to a detector or a detector array [11]. The laser and detector arrays rotate, taking multiple measurements as it scans the full 360 degree field of view. The range to a target can be determined based on the travel time of the laser pulse by $R = \frac{c}{2}(t_{rx} - t_{tx})$, where c is the speed of light in the intervening medium, t_{tx} and t_{rx} is the transmission and reception time of the laser pulse, respectively. The laser travels twice the distance to the target between transmission and reception time, hence the resulting range is halved. Relativistic effects are assumed negligible. The range resolution is determined by $\Delta R = \frac{c}{2B}$, where B is the system bandwidth, typically given by the transmit signal bandwidth.

Figure 3.5: The Velodyne VLP-16 lidar.

3.2.1 Velodyne VLP-16 Lidar Specifications

The lidar used in this project is the Velodyne VLP-16, which is a small, real-time rotating lidar which steams data over a TCP/IP connection when it is powered up. The lidar has multiple return modes, where it can report either the strongest return signal, the last return signal, or both. The default return mode is to report the strongest return, and this mode is what is used in this project. The Velodyne VLP-16 features 16 laser/detector pairs mounted in a rotating housing, rapidly spinning to produce 360° 3D image.

Some features of the VLP-16 are [12]:

- Horizontal FOV of 360°.
- Weight of 830 grams.
- Rotational speed of 5-20 RPS (adjustable).
- Vertical FOV of 30° (+15° to -15°).
- Vertical angular resolution of 2°.
- Horizontal angular resolution of 0.1°-0.4°.
- Range of up to 100 meters (range depends on application).
- Accuracy ± 3 cm (typical).

- 903 nm wavelength laser.

3.2.2 Lidar Sensor Model

A lidar is an optical detector, similar to a digital camera in that it responds to the intensity of the light hitting the detector, generating a voltage equal to the square of the intensity of the impinging light [11]. A rotating 3D lidar returns the intensity of the measured signal along with the range calculated from the travel time, as well as the azimuth (rotation) angle and the vertical angle of the point. The lidar has its own reference frame (in sperical coordinates), and it keeps track of its own orientation, giving the azimuth angle of a point relative to some reference angle (typically determined by the manufacturer), and the vertical angle is referenced to the horizontal plane in the lidar frame of reference. The coordinates of a point in the lidar frame therefore consists of two angles, α and β, as well as the distance R from the lidar to the point. Figure 3.6 illustrates this.

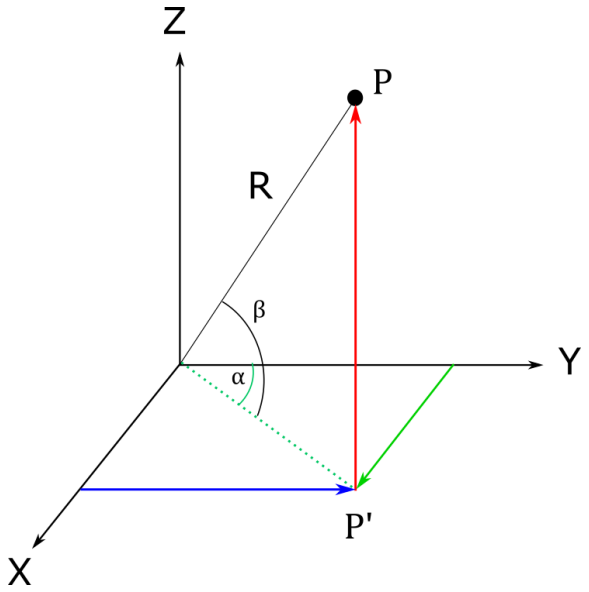


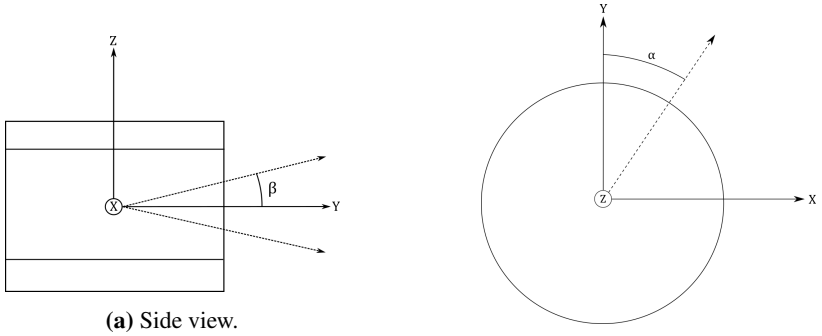
Figure 3.6: Lidar spherical coordinate system, with equivalent cartesian coordinates.

In figure 3.6, α is the azimuth angle, relative to the y-axis. β is the vertical/elevation angle, and R is the distance from the lidar to the point P. From figure 3.6, using basic trigonometry, we get that the transformation from spherical to cartesian coordinates is given by

$$\begin{aligned} X &= P' \sin\alpha = R\cos\beta\sin\alpha \\ Y &= P' \cos\alpha = R\cos\beta\cos\alpha \\ Z &= R\sin\beta \end{aligned} \tag{3.16}$$

The range returned by the Velodyne VLP-16 lidar is measured along a beam at known azimuth angles α, and the 16 laser/detector pairs are mounted vertically at equidistant

angles from -15° to 15°. When the lidar is rotating at a frequency of 10 Hz the lasers fire at every $0.2°$, with a total of 1800 laser firings for a full rotation. The theoretical maximum number of range measurements for a full $360°$ scan is therefore $\frac{360}{0.2} \times 16 = 28800$ measurements, where every single laser beam is reflected and detected at the lidar.



(a) Side view.

(b) Top view.

Figure 3.7: The lidar cartesian coordinate frame, with angles from the spherical coordinate frame drawn in.

A single reflection from the lidar can be represented in a cartesian coordinate frame centered in the lidar, shown in figure 3.7, using the transformation given in equation 3.16 as:

$$\mathbf{P}_{ij}^{l} = \begin{bmatrix} R_{ij} \cos\beta_i \sin\alpha_j \\ R_{ij} \cos\beta_i \cos\alpha_j \\ R_{ij} \sin\beta_i \end{bmatrix} \tag{3.17}$$

Here, α_j is the j^{th} azimuth angle in the scan, and β_i is the angle of the i^{th} vertical laser. The superscript l denotes the cartesian frame centered in the lidar. R_{ij} is the range measured at the given azimuth and elevation angle. The measurement from the lidar also includes a measure of the returned signal strengt, given as an integer in the range 0-255. If we denote the returned signal strength for the ij^{th} measurement as $\rho_{ij} = [0, 255] \in \mathbb{N}$, and augment the measurement vector with the returned signal strength, we can write a single measurement as

$$\mathbf{z}_{ij}^{l} \overset{\text{def}}{=} \begin{bmatrix} \mathbf{P}_{ij}^{l} \\ \rho_{ij} \end{bmatrix} = \begin{bmatrix} R_{ij} \cos\beta_i \sin\alpha_j \\ R_{ij} \cos\beta_i \cos\alpha_j \\ R_{ij} \sin\beta_i \\ \rho_{ij} \end{bmatrix} \tag{3.18}$$

where the *measurement* $\mathbf{z}_{ij}^{l}$ denotes the position of the reflection point $\mathbf{P}_{ij}^{l}$ along with the strength of the reflected signal. This model assumes that the range measurement R_{ij} reported by the lidar is exact, as well as that the angles β_i and α_j are known with perfect accuracy, which is not the case. There is uncertainty in the measurements, but this is assumed negligible in order to have a tractable measurement model.

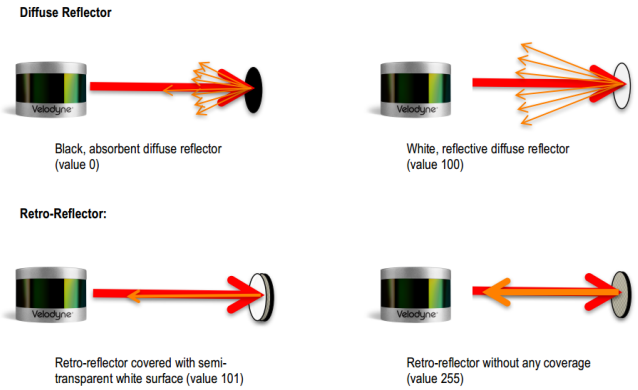


Figure 3.8: Reflector types used in calibrating the VLP-16 lidar. Image taken from [12].

3.2.3 Lidar Calibration

In order to use the measurements in conjunction with other measurements, such as those from a camera, they must be transformed to some common coordinate frame. As described in section 3.1.3, the measurement from the lidar can be transformed to another frame of reference by a rotation followed by a translation. If a navigation system gives the translation $\mathbf{t}_{wl}^{w}$ and the rotation $\mathcal{R}_{l}^{w}$ from the lidar frame to the world frame, the lidar measurement can be written in the world frame as

$$\mathbf{z}_{ij}^{w} = \begin{bmatrix} \mathbf{t}_{wl}^{w} + \mathcal{R}_{l}^{w}\mathbf{P}_{ij}^{l} \\ \rho_{ij} \end{bmatrix} \tag{3.19}$$

which is the lidar measurement model for a single point reflection, giving the 3D point coordinates alongside the returned signal strength as a 4×1 vector.

The returned signal strength is a measure of the reflectivity of the surface of the object reflecting the laser beam, and the Velodyne VLP-16 lidar is factory-calibrated using commercially available reflectivity standards and retro-reflectors [12]. The calibration data is stored in a calibration table within the FPGA in the VLP-16. Figure 3.8 illustrates the different reflector types used in the calibration.

3.2.4 The Lidar Point Cloud

For an entire 360° scan, the measurement returned by the lidar can be represented as an aggregation of single point measurements as

$$\mathbf{Z}^{w} = \begin{bmatrix} \mathbf{z}_{11}^{w} & \mathbf{z}_{12}^{w} & \dots & \mathbf{z}_{1j}^{w} \\ \mathbf{z}_{21}^{w} & \mathbf{z}_{22}^{w} & \dots & \mathbf{z}_{2j}^{w} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{z}_{i1}^{w} & \mathbf{z}_{i2}^{w} & \dots & \mathbf{z}_{ij}^{w} \end{bmatrix} \tag{3.20}$$

where, as before, i denotes the index of the vertical laser (from 1 to 16 for the velodyne VLP-16), and j is the index of the asimuth angle in the right-open interval $[0°, 360°[$.

When the Velodyne VLP-16 rotates at 10 Hz, j ranges from 1 to 1800. $\mathbf{Z}^w$ represents a *point cloud* made by a full rotation of the lidar, decomposed in the cartesian world frame.

Chapter 4

Method

4.1 Sensors, Hardware and Processing Pipeline

The physical sensor setup used in the data collection is illustrated in figure 4.1. A HP zbook 15 running Ubuntu 16.04 with ROS was used as a data collection platform.

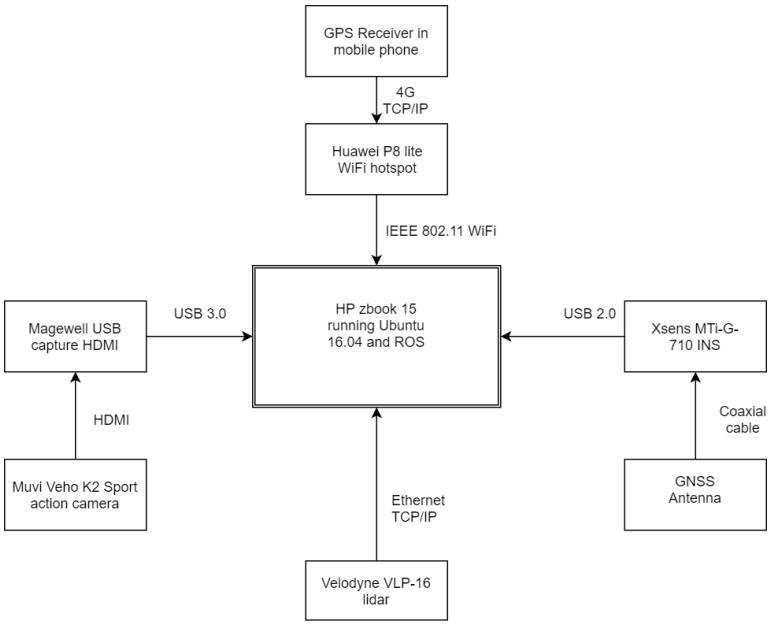


Figure 4.1: Sensor setup with physical connections.

The camera used to gather image data is a commercial off-the-shelf action camera by Muvi, named Veho K2 Sport. The reasoning for using such a camera was the affordable

price, relatively high resolution, and a wide angle of view. A serious drawback with such a camera for computer vision applications, however, is the fact that it only sends images as a steady stream over its HDMI interface, with little options for the user to control the capture of individual frames. Moreover, HDMI is not

Kilde p at hdmi suger for real time CV

The specifications of the computer used in the data gathering is listed in table 4.1.

Model	HP zbook 15
CPU	Intel Core i7-4800MQ 8×2.7 GHz
Memory	8 GB RAM
Graphics	lala
Storage	250 GB Solid State HDD
Operating System	Linux Ubuntu 16.04 LTS (Xenial)

Table 4.1: Computer specifications.

Overview of the sensor setup, hardware used, and signal flow through the system.

4.2 Faster R-CNN

Introduce the framework of Faster R-CNN

4.2.1 Training and validation data

Describe the datasets used in training and validation (Espen Tangstads work basically)

4.2.2 Implementation Aspects

Steps needed to run the Matlab implementation from https://github.com/ShaoqingRen/faster_rcnn. Compiling Caffe with cuda for GTX1070, the link between caffe and Matlab (mex).

4.3 ROS Implementation

In order to capture, organize, timestamp and save the collected data, the open-source robotics framework of ROS was used. ROS, which is an abbreviation for Robot Operating System, is not, as the name might imply, a operating system, but a framework running on top of a traditional OS. The framework of ROS is based on several philosophical aspects [13]:

- *Peer to peer*: A ROS system consists of several small programs (called nodes) connected to each other, continuously exchanging messages. Messages travel directly from one node to another.

- *Tools-based*: Complex ROS systems can be created from many small, generic programs. ROS does not have a integrated development and runtime environment, and tasks such as (but not limited to) navigating the source code tree, visualizing the system interconnections, generating documentation, and logging data are performed by separate programs.
- *Multilingual*: ROS software modules can be written in any language for which a *client library* has been written. Client libraries exist for C++, Python, LISP, Java, JavaScript, MATLAB, Ruby, Haskell, R, and others. This provides flexibility for the programmer, in that he or she can choose the language they are most familiar with, or is best suited for the task, when creating new functionality. ROS achieves this multilinguality by enforcing a convention for serializing messages being passed between nodes.
- *Thin*: ROS conventions encourage developers to create standalone libraries and then wrap those libraries so that they can send and receive messages to and from other ROS modules. This allows reuse of software outside ROS, and simplifies automated testing using continuous integration tools.
- *Free and open source*: The core of ROS is released under the BSD license [14], which allows commercial and noncommercial use.

In ROS, `roscore` is a service that provides connection information to nodes so that they can transmit messages to and from one another. Each node connects to `roscore` on startup to register details of the messages it publishes, and the details of the messages to which it wishes to subscribe. `roscore` is not a server in the traditional client/server sense, all messages are sent peer-to-peer between nodes. The `roscore` service is only used by nodes to know where to find their peers. Due to this fact, every ROS system needs a running `roscore`. Upon startup, every ROS node expects its process to have an environment variable `ROS_MASTER_URI`, containing a string in the format `http://hostname:portnumber`, to tell it where the `roscore` service is running, and which port it is accessible on. The connection between nodes, and between individual nodes and `roscore` is illustrated in figure 4.2.

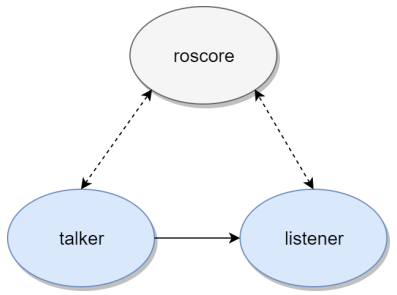


Figure 4.2: Roscore and its connection to other nodes in the system.

The ephemeral connection between nodes and `roscore` is illustrated by dashed lines, indicating periodic calls to `roscore` to find peers, while the peer-to-peer message pass-

ing is illustrated by the solid line between the nodes. The main ROS client libraries are geared towards a UNIX/Linux platform, mainly due to the dependency on large collections of open-source software dependencies . Ubuntu Linux is a supported distribution, while others, such as Fedora Linux, Mac OS, and Windows, are designated as experimental and are supported by the community. Due to the simplicity of installation and ”out-of-the-box” functionality of ROS on Ubuntu Linux, the most recent long-term supported version of ROS, ROS Kinetic, was installed on a laptop running Ubuntu 16.04 LTS, which was used as a platform for running the various sensor drivers and collecting the data.

kilde

4.3.1 ROS Nodes and Data Flow

An overview of the implemented sensor driver and data processing nodes is illustrated in figure 4.3.

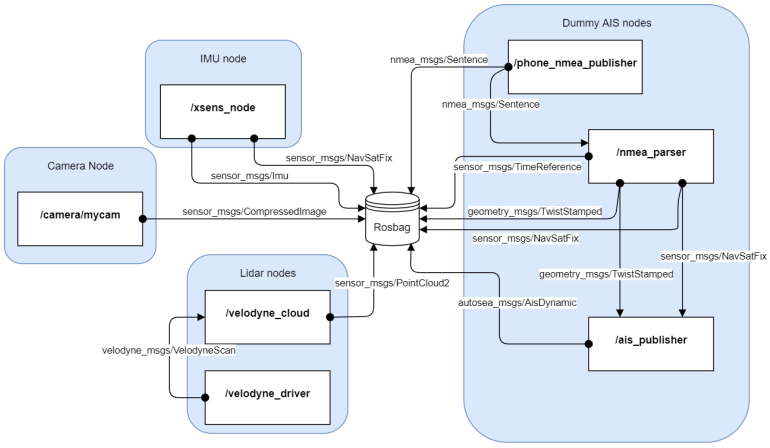


Figure 4.3: ROS nodes and topic flow.

The lidar nodes, camera node and IMU node are all based on open-source ROS packages freely available from the ROS community. The camera driver used is the `libuvc_camera` driver. This node reads the image stream from the camera over the USB port, providing a ROS interface for cameras meeting the USB Video Class standard [15]. This node publishes messages containing an image corresponding to a single frame of the video stream, with a time stamp corresponding to the arrival time of the image frame over USB. The source code is available at the ROS device drivers repository [16].

The lidar nodes are in the `velodyne_driver`-package from the ROS device drivers repository [16]. This package provides two nodes, the first is `/velodyne_driver` for collecting all TCP packages sendt over ethernet from the lidar and publishing a message containing the raw data from a full 360-degree scan from the lidar. The second node, `/velodyne_cloud`, transforms the raw data into a point cloud given in the frame of reference the user chooses (default `/velodyne`, the sensor frame) in XYZ-coordinates. In addition to the coordinates of each point, the corresponding intensity of the reflected

signal strength and the ring data is published [17].

The point cloud is then published as a ROS message. In addition to the coordinates of each point, the corresponding intensity of the reflected signal strength and the ring data is also included in this message [17].

The driver for the IMU is the `xsens_driver` package [18], which communicates with the Xsens IMU over USB, and publishes messages containing the measurements from the IMU and GNSS position measurements [18].

The dummy AIS nodes in figure 4.3 are nodes for receiving and processing GNSS-information received over the internet from a mobile phone, developed at NTNU by the Autosea-project . These nodes are used for receiving and storing the position ground truth for the vessel used in the experiments along with all the other data. They provide raw data as an NMEA-0183 sentence, which is a standard defined by the National Maritime Electronics Association for communication between marine electronics, as well as speed over ground and GNSS position data. The details of messages passed between nodes in the system is given in table 4.2.

siter autosea et eller annet

Topic	Content
`nmea_msgs/Sentence`	GNSS Data as a String representing a NMEA0183 sentence
`sensor_msgs/NavSatFix`	Navigation Satellite fix specified using WGS 84 ellipsoid
`geometry_msgs/TwistStamped`	Linear and angular velocity
`autosea_msgs/AisDynamic`	AIS message generated from GNSS data
`sensor_msgs/TimeReference`	External time reference not actively synchronized with system time (i.e. GNSS time)
`sensor_msgs/PointCloud2`	Message containing point cloud data from the lidar
`velodyne_msgs/VelodyneScan`	Message containing raw data from a 360deg lidar scan
`sensor_msgs/CompressedImage`	Message containing a compressed image in .JPEG format
`sensor_msgs/Imu`	Message containing raw measurements from the IMU

Table 4.2: ROS Topics used in data collection.

4.3.2 Rosbag

ROS provides a built-in set of tools for recording and playing back ROS topics, `rosbag` [19]. This toolset stores the topics of interest, specified as parameters in the command-

line, to a file of the ROS bag format, with extension .bag. This toolset provides a simple method for storing data from experiments, and the playback function provides the option to simulate sensor inputs to the system without actually reading sensors. A basic use case, where for example two topics named `/imu_data` and `/camera/image` are to be recorded, the proper command line input is

```
$ rosbag record -o filename /imu_data /camera/image
```

where `filename` is the user-specified filename for the .bag file. ROS appends the filename with the system time automatically. Rosbag was used in recording all experimental data.

4.3.3 MATLAB implementation

MATLAB provides ROS interfaces via the Robotics System Toolbox, making it possible to use ROS topics directly in MATLAB.

4.4 Camera Calibration

As explained in section 3.1.3, one needs the cameras intrinsic parameters in order to associate pixel coordinates with world coordinates. MATLABs Computer Vision System Toolbox includes the Camera Calibrator application, which was used in calibrating the camera. The Camera Calibrator application implements the calibration routine by Zhang [20], described in appendix A.1 [21]. The steps taken for calibrating the camera using this application can be summarized as follows:

1. Prepare a checkerboard calibration pattern with known dimensions.
2. Capture 20-40 images of the checkerboard with the camera, from varying orientations.
3. Initiate the calibration.
4. Evaluate calibration accuracy.
5. Adjust parameters to improve accuracy (if necessary).
6. Export parameters as a cameraParameters object.

The accuracy in the calibration routine can be improved by removing outlier images, which contribute more to the reprojection errors than others. Figure 4.4 shows an example of a blurred image, which due to the error introduced to keypoint detection by the blurryness of the image causes a larger reprojection error.

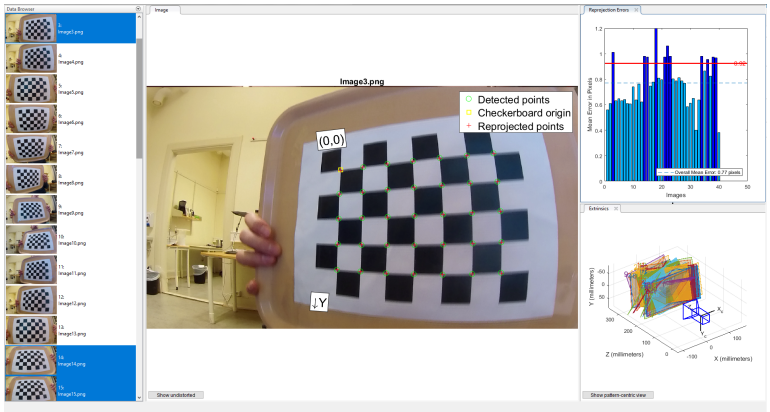
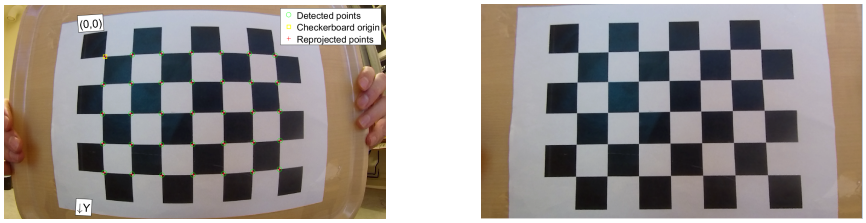


Figure 4.4: Camera Calibration app showing an outlier image.

The outliers can be removed by adjusting the threshold for mean pixel reprojection error shown in the upper right corner of figure 4.4. The calibration app also computes the extrinsic parameters and displays the calibration pattern planes in 3D, as seen in the lower right corner in figure 4.4. Having calibrated the camera, the radial distortion in the images can be removed, as seen in figure 4.5, where a unprocessed image with the calibration pattern key points overlaid can be seen next to the undistorted version.



(a) Distorted image. **(b)** Undistorted image.

Figure 4.5: Distorted checkerboard image, along with the undistorted version.

The numerical values for the intrinsic matrix was found using this routine as

$$\mathcal{K} = \begin{bmatrix} \alpha & -\alpha\cot\theta & x_0 \\ 0 & \frac{\beta}{\sin\theta} & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1068.7558 & -1.9413 & 893.7339 \\ 0 & 1066.6539 & 516.2405 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

4.4.1 Time Synchronization

Time delay between camera and imu, Kalibr calibration routine.

Chapter 5

Experiments

In order to get data to analyse and evaluate the proposed methodology, a series of experiments was performed using a small boat in the channel near Ravnkloa in Trondheim. The sensors were mounted on a rigid mount, and the boat performed a series of maneuvers in front of the sensors. The data was saved in rosbags during the experiments, to be post-processed in MATLAB after the data gathering.

Chapter 6

Analysis

Analyze data from experiments, present results achieved.

Chapter 7

Discussion

Discuss the results from the previous chapter, with a critical eye.

Chapter 8

Conclusion

Conclusion, and suggestion of further work.

Bibliography

[1] W. Koch, *Tracking and Sensor Data Fusion: Methodological Framework and Selected Applications*. Springer, 2014.

[2] A. Steinberg and C. Bowman, "Revisions to the jdl data fusion model," in *Handbook of Multisensor Data Fusion - Theory and Practice*, M. E. Liggins, D. L. Hall, and J. Llinas, Eds., CRC Press, 2008, ch. 3, pp. 45–66.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson, 2008.

[5] M. Forsyth and J. Ponce, *Computer Vision, A Modern Approach*. Pearson, 2012.

[6] R. Szeliski, *Computer Vision, Algorithms and Applications*. Springer, 2011.

[7] B. Bayer, *Color imaging array*, US Patent 3,971,065, 1976. [Online]. Available: https://www.google.com/patents/US3971065.

[8] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2002.

[9] J. Wittenburg, *Kinematics: Theory and Applications*. Springer, 2016.

[10] D. S. Bernstein, *Geometry, Kinematics, Statics and Dynamics*. Princeton University Press, 2012.

[11] P. McManamon, *Field Guide to Lidar*. SPIE, 2015.

[12] I. Velodyne LiDAR, *User manual and programming guide, vlp-16*, English, version 63-9243 Rev B Mar 16, Velodyne LiDAR, Inc., 49 pp., March, 2016.

[13] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS*. O'Reilly, 2015.

[14] *The 3-clause bsd license*, https://opensource.org/licenses/BSD-3-Clause, Accessed: 2017-12-06.

[15] O. S. R. Foundation, *Libuvc_camera - ros wiki*, http://wiki.ros.org/libuvc_camera?distro=kinetic, Accessed: 2017-12-06.

[16] *Ros device drivers*, https://github.com/ros-drivers, Accessed: 2017-12-06.

[17] ——, *Velodyne - ros wiki*, http://wiki.ros.org/velodyne?distro=kinetic, Accessed: 2017-12-06.

[18] ——, *Xsens_driver - ros wiki*, http://wiki.ros.org/xsens_driver, Accessed: 2017-12-06.

[19] *Rosbag - ros wiki*, http://wiki.ros.org/rosbag, Accessed: 2017-12-09.

[20] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, 2000.

[21] I. MathWorks, *Estimate geometric parameters of a single camera - matlab*, https://se.mathworks.com/help/vision/ref/cameracalibrator-app.html, Accessed: 2017-12-08.

[22] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, 1993.

[23] Z. Zhang, "A Flexible New Technique for Camera Calibration," Microsoft Corporation, Tech. Rep., Dec. 1998.

[24] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[25] G. Wei and S. De Ma, "Implicit and explicit camera calibration: Theory and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 469–480, 1994.

[26] D. C. Brown, "Close-range camera calibration," *PHOTOGRAMMETRIC ENGINEERING*, vol. 37, no. 8, pp. 855–866, 1971.

Appendix A

Appendix

A.1 Geometric Camera Calibration

Camera calibration is a necessary step in computer vision applications in order to extract metric information from 2D images. The intrinsic and extrinsic parameters of a camera can be estimated from the image positions of scene features whose geometry in 3D space is known with very good precision [5]. The calibration setup usually consists of two or three orthogonal planes with a known calibration pattern (such as a chessboard). In fact, if the 3D reference points lie in the same plane, calibration will not work [22]. This setup, however, requires an expensive calibration apparatus, and an elaborate setup. Zhang proposed an alternative, flexible approach that doesn't require knowledge of the 3D space geometry of the calibration setup [20]. The technique described by Zhang only requires the camera to observe a planar pattern shown at at least two different orientations. The calibration procedure performed in this project is based on this work, and a introduction to the theory is presented in this section.

A.1.1 Perspective Projection between the Model Plane and Its Image

In order to be consistent with the notation used in Zhang's paper, let's rewrite equation 3.13 as

$$s\tilde{\mathbf{m}} = \mathbf{A}\begin{bmatrix}\mathcal{R} & \mathbf{t}\end{bmatrix}\tilde{\mathbf{M}}, \quad \text{where} \quad \mathbf{A} = \begin{bmatrix}\alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1\end{bmatrix} \tag{A.1}$$

where $\tilde{\mathbf{m}} = \begin{bmatrix}u & v & 1\end{bmatrix}^T$ is the image plane coordinates, $\tilde{\mathbf{M}} = \begin{bmatrix}X & Y & Z & 1\end{bmatrix}^T$ as the 3D point coordinates expressed in the world frame, s is an arbitrary scale factor, $\begin{bmatrix}u_0 & v_0\end{bmatrix}^T$ describe the position of the image center in pixel units, α and β are the scale factors in the u and v axes, and γ describes the skew between the two image axes. Now, without loss of generality, assume that the model plane is located at $Z = 0$, and let $\mathbf{r}_i$ denote the ith

column of $\mathcal{R}$. Equation A.1 can then be expressed as

$$s\begin{bmatrix}u\\v\\1\end{bmatrix} = \mathbf{A}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\0\\1\end{bmatrix} = \mathbf{A}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix} \tag{A.2}$$

By slight abuse of notation, let $\tilde{\mathbf{M}}$ still represent a point on the model plane, but since Z is always zero, we write $\tilde{\mathbf{M}} = \begin{bmatrix}X\\Y\\1\end{bmatrix}$. With this, a point on the model plane and its corresponding image coordinates are related by a projective transformation (also known as a *homography*) matrix $\mathbf{H}$:

$$s\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{M}}, \quad \text{where} \quad \mathbf{H} = \mathbf{A}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix} \tag{A.3}$$

The 3×3 matrix $\mathbf{H}$ is only defined up to a scale factor. Denoting $\mathbf{H} = \begin{bmatrix}\mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3\end{bmatrix}$, and knowing that since $\mathbf{r}_1^T\mathbf{r}_2 = 0$, we have

$$\mathbf{h}_1^T(\mathbf{A}^{-1})^T\mathbf{A}^{-1}\mathbf{h}_2 = 0 \tag{A.4}$$

and

$$\mathbf{h}_1^T(\mathbf{A}^{-1})^T\mathbf{A}^{-1}\mathbf{h}_1 = \mathbf{h}_2^T(\mathbf{A}^{-1})^T\mathbf{A}^{-1}\mathbf{h}_2 \tag{A.5}$$

These equations are the two basic constraints on the intrinsic parameters, given a single projective transformation. Due to the fact that a projective transformation has 8 degrees of freedom, and there are six extrinsic parameters, one can only obtain two constraints on the intrinsic parameters [20].

A.1.2 Solving the Calibration Problem

Let

$$\begin{aligned}\mathbf{B} = (\mathbf{A}^{-1})^T\mathbf{A}^{-1} &= \begin{bmatrix}B_{11} & B_{12} & B_{13}\\B_{12} & B_{22} & B_{23}\\B_{13} & B_{23} & B_{33}\end{bmatrix}\\ &= \begin{bmatrix}\frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma-u_0\beta}{\alpha^2\beta}\\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2}+\frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma-u_0\beta)}{\alpha^2\beta^2}-\frac{v_0}{\beta^2}\\ \frac{v_0\gamma-u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma-u_0\beta)}{\alpha^2\beta^2}-\frac{v_0}{\beta^2} & \frac{(v_0\gamma-u_0\beta)^2}{\alpha^2\beta^2}+\frac{v_0^2}{\beta^2}+1\end{bmatrix}\end{aligned} \tag{A.6}$$

$\mathbf{B}$ is symmetric, defined by a 6D vector

$$\mathbf{b} = \begin{bmatrix}B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33}\end{bmatrix}^T \tag{A.7}$$

Defining the i^{th} column vector of $\mathbf{H}$ as $\mathbf{h}_i = \begin{bmatrix}h_{i1} & h_{i2} & h_{i3}\end{bmatrix}^T$, we can rewrite equation A.4 as

$$\mathbf{h}_i^T\mathbf{B}\mathbf{h}_j = \mathbf{v}_{ij}^T\mathbf{b} \tag{A.8}$$

where

$$\begin{aligned}\mathbf{v}_{ij} =&[h_{i1}h_{j1} \;\; h_{i1}h_{j2} + h_{i2}h_{j1} \;\; h_{i2}h_{j2} \\ & h_{i3}h_{j1} + h_{i1}h_{j3} \;\; h_{i3}h_{j2} + h_{i2}h_{j3} \;\; h_{i3}h_{j3}]^T\end{aligned} \tag{A.9}$$

The two constraints on the intrinsic parameters can then be written as two homogeneous equations in $\mathbf{b}$ as

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{A.10}$$

If n images of the model plane are observed, we can stack n equations as equation A.10 and get

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{A.11}$$

where $\dim(\mathbf{V}) = 2n \times 6$. For $n \geq 3$ we will generally have a unique solution b given as the eigenvector og $\mathbf{V}^T\mathbf{V}$ associated with the smallest eigenvalue, defined up to a scale factor. Once $\mathbf{b}$ is estimated, one can find all the intrinsic parameters in $\mathbf{A}$ from $\mathbf{b}$ by

$$\begin{aligned} v_0 &= \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B22 - B_{12}^2} \\ \lambda &= B_{33} - \frac{B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \\ \alpha &= \sqrt{\frac{\lambda}{B_{11}}} \\ \beta &= \sqrt{\frac{\lambda B_{11}}{B_{11}B22 - B_{12}^2}} \\ \gamma &= -\frac{B_{12}\alpha^2\beta}{\lambda} \\ u_0 &= \frac{\gamma v_0}{\alpha} - \frac{B_{13}\alpha^2}{\lambda} \end{aligned} \tag{A.12}$$

Now that $\mathbf{A}$ is known, the extrinsic parameters can easily be computed from equation A.3 as

$$\begin{aligned} \mathbf{r}_1 &= \lambda\mathbf{A}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda\mathbf{A}^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda\mathbf{A}^{-1}\mathbf{h}_3 \end{aligned} \tag{A.13}$$

where $\lambda = 1/||\mathbf{A}^{-1}\mathbf{h}_1||$. The parameters can be further refined through maximum likelihood inference. Assuming that the image points are corrupted by independent and identically distibuted (i.i.d.) noise, the maximum likelihood estimate can be obtained by minimizing

$$\sum_{j=1}^{n}\sum_{i=1}^{m} ||\mathbf{m}_{ij} - \hat{\mathbf{m}}(\mathbf{A}, \mathcal{R}_i, \mathbf{t}_i, \mathbf{M}_j)||^2 \tag{A.14}$$

where $\hat{\mathbf{m}}(\mathbf{A}, \mathcal{R}_i, \mathbf{t}_i, \mathbf{M}_j)$ is the projection of point $\mathbf{M}_j$ in image i as described in equation A.3.

A.1.3 Radial Lens Distortion

In the previous sections lens distortion models has not been discussed. A typical camera, however, will to a varying degree exhibit lens distortion, especially radial distortion. Figure A.1 illustrates positive and negative radial distortion, known as barrel distortion and pincushion distortion, respectively. Zhang only considered radial distortion in his work, the argument being that more elaborate models not only did not improve the result, but could also produce numerical instability [23]. The interested reader is referred to [24]–[26] for a more thorough description of distortion models considering effects as decentering and thin prism distortion in addition to radial distortion.

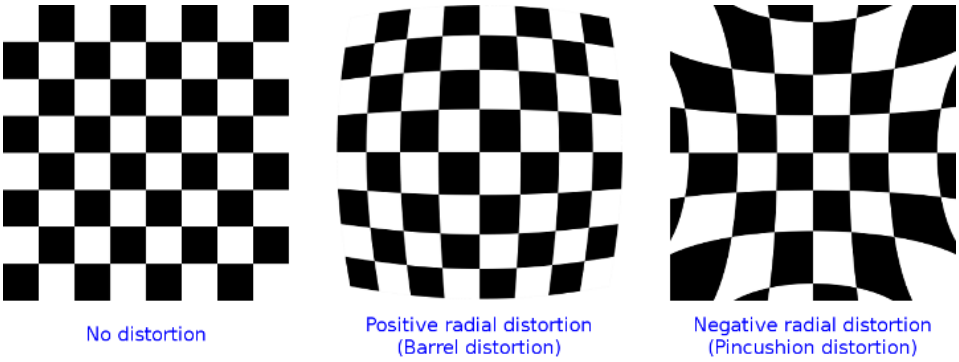


Figure A.1: Examples of radial distortion. [1]

Considering only radial distortion, let (u, v) represent the ideal (distortion-free) pixel image coordinates, and $(\breve{u}, \breve{v})$ the corresponding observed real coordinates. Then, using equation A.1 and assuming $\gamma = 0$, we have [23]

$$\breve{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{A.15}$$

$$\breve{v} = v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{A.16}$$

where k_1 and k_2 are coefficients of the radial distortion. This can be rewritten as

$$\begin{bmatrix} (u - u_0)(x^2 + y^2) & (u - u_0)(x^2 + y^2)^2 \\ (v - v_0)(x^2 + y^2) & (v - v_0)(x^2 + y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \breve{u} - u \\ \breve{v} - v \end{bmatrix} \tag{A.17}$$

In a similar fashion as for equation A.11, for m points in n images, this can be stacked to obtain $2mn$ equations in matrix form as $\mathbf{Dk} = \mathbf{d}$, $\mathbf{k} = \begin{bmatrix} k_1 & k_2 \end{bmatrix}^T$. The linear least-squares solution is given by

$$\mathbf{k} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{d} \tag{A.18}$$

Once k_1 and k_2 are estimated, equation A.14 can be extended to estimate the complete set of parameters including k_1 and k_2 by minimizing

$$\sum_{j=1}^{n}\sum_{i=1}^{m} ||\mathbf{m}_{ij} - \breve{\mathbf{m}}(\mathbf{A}, k_1, k_2, \mathcal{R}_i, \mathbf{t}_i, \mathbf{M}_j)||^2 \tag{A.19}$$

[1] Image courtesy: https://docs.opencv.org/2.4.13.2/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

where $\breve{\mathbf{m}}(\mathbf{A}, k_1, k_2, \mathcal{R}_i, \mathbf{t}_i, \mathbf{M}_j)$ is the projection of the point $\mathbf{M}_j$ in image j according to equation A.3, followed by distortion according to equations A.15 and A.16. This is a nonlinear minimization problem, which is solved by the Levenberg-Marquardt algorithm. This requires an initial guess for $\mathbf{A}$ and $\mathcal{R}_i, \mathbf{t}_i,\ i = 1 \dots n$ which can be obtained by solving equation A.11. k_1 and k_2 can be estimated by solving A.18, or simply set to 0 initially [23]. Typically, the rotation matrix $\mathcal{R}$ returned from this procedure will not be a true rotation matrix (where the rows and columns are orthogonal unit vectors) due to image noise. A method for estimating the best rotation matrix from a general 3×3 matrix is given in [23].

kilde p dette?

A.1.4 Calibration Procedure

In Zhang's paper, the recommended calibration procedure is summarized as [20]:

1. Print a calibration pattern and attach it to a planar surface
2. Take a few images of the model plane under different orientations by moving either the plane or the camera
3. Detect the feature points in the images
4. Estimate the five intrinsic parameters and all the extrinsic parameters using the solution of equation A.11
5. Estimate the coefficients of the radial distortion by solving equation A.18
6. Refine all parameters by minimizing equation A.19