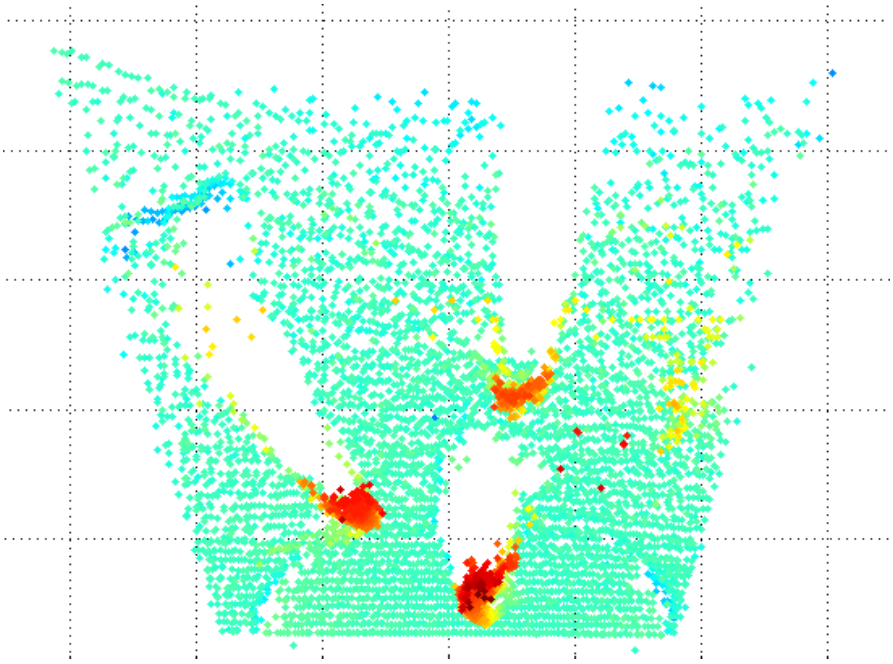
CHALMERS



.

# Evaluation and implementation of 3D imaging sensors for close range object detection from a truck safety application perspective

*Master of Science Thesis*

OLLE BRÄNNLUND
DAVID ÖST

Department of Signals and Systems

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2013
Report No. xxxx

# Contents


**1 Introduction** **1**
- 1.1 Background . . . 1
- 1.2 Goals . . . 2
- 1.3 Limitations . . . 3

**2 The sensors - placing, theory and evaluation** **4**
- 2.1 Placing . . . 4
- 2.2 Time of flight sensor . . . 7
  - 2.2.1 Theory . . . 7
  - 2.2.2 Issues with time of flight cameras . . . 9
  - 2.2.3 Advantages with time of flight cameras . . . 11
  - 2.2.4 Measurement accuracy evaluation . . . 12
- 2.3 Stereo camera . . . 14
  - 2.3.1 Theory . . . 15
  - 2.3.2 Issues with stereo vision . . . 17
  - 2.3.3 Advantage with stereo vision . . . 17
- 2.4 Camera models . . . 17
  - 2.4.1 TOF sensor . . . 18

**3 Choice of algorithms** **20**
- 3.1 Interpreting input data . . . 20
- 3.2 Tracking algorithms . . . 22
  - 3.2.1 Data association . . . 22

**4 Data processing** **24**
- 4.1 Overview of the process . . . 25
- 4.2 Camera calibration . . . 26
  - 4.2.1 Estimating the real world orthonormal vectors in the (implicit) sensor space . . . 26
  - 4.2.2 Correcting camera vectors (for B70 TOFS specifically) . . . 26
- 4.3 Live processing . . . 27
  - 4.3.1 Unwrapping (for TOFS only) . . . 27
  - 4.3.2 Converting depth image to 3D coordinates . . . 28
  - 4.3.3 Classifying pixels . . . 28
  - 4.3.4 Occupancy and unknown grid . . . 29

**5 Tracking Algorithms** **31**
- 5.1 Algorithms based on Kalman filtering . . . 31
  - 5.1.1 Introduction to the Kalman filter . . . 32
  - 5.1.2 Detection . . . 33
  - 5.1.3 Target states . . . 34
  - 5.1.4 Kalman filtering matrices . . . 34
  - 5.1.5 Target-detection association . . . 36
  - 5.1.6 How the detections are formed for the different cases . . . 40
- 5.2 Tracking - Particle based . . . 46

5.2.1 Target state forming . . . 48
5.2.2 Target association . . . 48

**6 Results** **50**
6.1 Evaluation . . . 50
6.1.1 Detection evaluation . . . 54
6.1.2 Tracking evaluation . . . 56
6.1.3 Speed evaluation . . . 62

**7 Conclusions and discussion** **64**
7.1 Evaluation of sensors . . . 64
7.1.1 Field of view . . . 64
7.1.2 Depth resolution . . . 65
7.2 Tracking . . . 65
7.3 Classification . . . 65
7.4 General conclusions and thoughts . . . 66

**Abstract**

In order to improve current safety functions in trucks and construct new ones, such as automatic braking, start inhibition or warning systems that for example indicate when obstacles are present in blind spots, truck manufacturers are interested in new sensor technology that can perform the task of environment perception.

This project, carried out at Chalmers and at the department for Safety Functions & Electronics at Volvo Group Trucks Technology, has aimed to examine two new kinds of sensors that are, or will be shortly, available in packaging designed for automotive implementations, namely stereo vision and time of flight cameras. Focus has been on object detection and tracking in the blind spots in front of the truck and to the sides around the cabin, in which pedestrians can be difficult for the driver to spot.

The two 3D perception approaches are evaluated through experiments and a literature study. Furthermore, the project has included designing and implementing appropriate data processing algorithms for the large amount of information that the sensors provide, together with implementing and evaluating a few different tracking algorithms which include the Kalman filter and a particle filter. A significant share of the work has been dedicated to detection-to-track association and the issue of how to deal with occlusions. We propose a couple of different approaches to solving this which are also implemented and tested, both in a controlled environment and in real world scenarios.

We have found that both sensor types have a promising future in active safety applications, but they have some issues that need to be addressed to ensure a robust functionality in all kinds of environments and weather conditions. One of the tracking algorithms proposed performs well in our experiments and could be suitable for future implementation depending on the requirements on robustness.

# Chapter 1

# Introduction

This project has been carried out at the department for Safety Functions & Electronics at Volvo Group Trucks Technology as a research project in the field of 3D cameras for active safety applications. Active safety in trucks is the umbrella term for a few different safety functions in trucks that autonomously monitor the environment, feed the information to a central processing unit that assesses threats and eventually if needed issues warnings to the driver or in extreme cases even takes over control of the truck. In an active safety system, knowledge of the surroundings is central as a basis for decision making, such as for automatic braking or start inhibition.

This section presents the background of the project, why 3D sensors are interesting from an automotive safety perspective and what we aimed to accomplish with the project.

## 1.1 Background

A significant risk source in truck driving is the limited close-range field of view (FOV). Because of the elevated driver position and the size of the cabin there are large blind spots directly in front of the cabin and to the sides, mainly on the passenger side. Figure 1.1 shows all of the blind spots. Besides the driver's need for extra eyes in these areas there are existing safety and comfort applications in trucks that could utilize information of the close-range region, such as automatic cruise control (which does not work in congested traffic since the distances are too small to be measured accurately by the current sensor). Currently, the most common sensors used by these systems are radar and RGB-cameras. However, these do not fully cover the mentioned regions so to do this an additional short-range sensor is required.

Trucks must, by law, be equipped with mirrors that reflect certain areas that would otherwise be hidden to the driver, but accidents still occur despite these mirrors being in place. The problems with mirrors are that they demand attention from the driver and they create new blind spots due to their lack of transparency. A camera or depth sensor, on the other hand, does not need to be placed in the drivers FOV, and signal processing hardware can detect hazardous situations and assist the driver.

The risks that come with the blind spots are especially prominent in urban environments where trucks drive among cyclists and pedestrians. For example, 43 % of all cyclist fatalities in London traffic involved freight vehicles in the period 1992 to 2006 [16]. Roughly half of these, or on average eight per year, occurred when the vehicle was making a left turn, which means they where in an area only visible to the driver indirectly through mirrors, as as the driver position is on the right side in British vehicles. According to [4] "15-25 % of the victims in heavy truck accidents are unprotected road users, i.e. pedestrians, cyclists and motorcyclists. Many of these accidents occur in low speed and limited visibility is one of the main causes.". It is clear that the blind spots is a problem that deserves attention.

This is where the 3D sensors come into the picture. New trucks are already equipped with sensors that perceive the environment, processing units that perform cognition and decision making and a human-machine interface that supplies the driver with warnings and information. The computer can also directly control the vehicle, e.g. by braking or accelerating. The scope of this project is hence to expand the perception and cognition parts of the system, which basically means doing research on sensors and detection and tracking

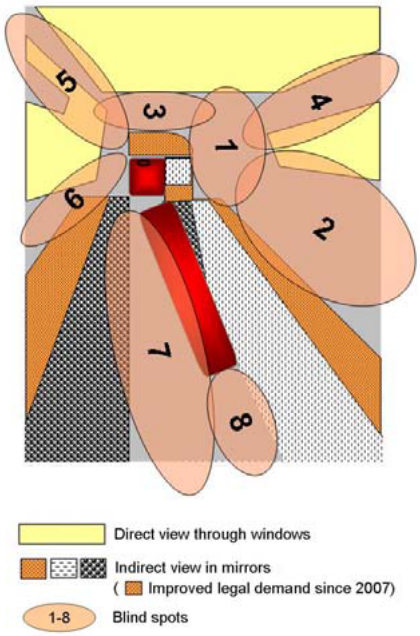



Figure 1.1: The rough size and locations of the blind spots of a truck. Area region of interest in this project has mainly been region 3 and the region between this and the truck, and also the dotted areas to the side of the truck. The dotted areas are covered by mirrors in today's trucks. (Image borrowed from [4].)

algorithms.

3D cameras are becoming more and more prevalent on the market as the technology is improved and production costs are reduced. With the depth data at hand the task of object detection becomes rather straightforward compared to 2D RGB or grey scale images. Therefore 3D cameras may be well suited for the environment perception part of an active safety system, and in extension even autonomous vehicles, and the purpose of this project is to investigate how this can be done and how well they can perform.

## 1.2 Goals

The main goal of the research has been to investigate the performance of primarily one type of depth sensor: time of flight (TOF) cameras. This includes implementing appropriate data processing and tracking algorithms and to evaluate how well this type of system could perform the task of object detection in the aforementioned blind spots. As a branch of the project a simple stereo camera has been constructed, which we have compared to the TOF sensor at hand, a Fotonic B-70. Even though we have to deal with sensor-specific problems in our implementation, it is the inherent problems and benefits of the time of flight technology that are of interest to Volvo and that will be the basis for our analysis and conclusions.

In order to reach this goal the following questions have been formulated which we attempt to answer in the project. They pinpoint some of the key aspects of depth sensor hardware and application that need to be investigated:

- What is the sensor's field of view and how should it be positioned to best cover the blind spots?
- Analysis of resolution; what precision does the sensor boast at different ranges?
- What is the sensors performance in different lighting conditions?
- Does the depth map need to be fused with an RGB image?

- Can a reliable object detection in the blind spots be done in real time?
- Can the sensor be used for classification, e.g. differentiation between various types of objects?
- What kind of tracking system is appropriate for a 3D camera?
- Is any additional information needed from other sensors on the truck to perform any of the tasks mentioned above?
- How can the output data be described so that bayesian tracking methods, decision making and data fusion can be employed at the next level of signal processing in an active safety system?
- Considering FOV and processing time, what types of accidents can be prevented?
- What is the theoretical upper bounds on how well object detection can be performed concerning e.g. speed, FOV and accuracy?

## 1.3 Limitations

In our research we will not treat the following areas:

- HMI - the human-machine interface will not be researched or dealt with in any way. We are only concerned with the perception and cognition parts of the system, i.e. collecting input data and interpreting it.
- Optimisation - optimisation of the algorithms with respect to execution speed will not be done more than show that the algorithms realistically could work in real time.
- Other sensor types - we will limit our analysis to optical ranging sensors, with focus on TOF cameras, but also including stereo cameras to some extent. Other sensors are not researched or tested.

# Chapter 2

# The sensors - placing, theory and evaluation

The 3D imaging technology that we have briefly studied is primarily time of flight (TOF, the abbreviation TOFS will also be used meaning Time Of Flight Sensor) cameras, but we have also briefly studied stereo cameras. Both types are available on the market, although not specifically designed for truck implementation[1]. During our research we have had access to a Fotonic B70 TOF camera. A stereo camera has been constructed from two low-end standard web cameras. It is not up to par with industrial stereo cameras but has still offered some insight into the technology.

TOF and stereo systems differ completely from each other in how they acquire depth maps, TOF cameras measure the time it takes for an emitted pulse of light to be reflect back onto the sensor, while stereo cameras use triangulation to determine the distance to objects in a pair of simultaneously captured intensity (i.e. grey scale) images from different positions. This needs to be taken into account later on when constructing the detection algorithms.

This chapter will give an introduction to the technologies and shed light on some of their advantages, but also some issues that need attention in a future implementation. We start off with a short analysis of where such a camera could be placed and what it could accomplish there.

## 2.1 Placing

To determine where the optimal mounting position of the camera is, there are a number of factors that need to be considered: which areas are the most difficult to see from the drivers perspective, the camera FOV, whether the camera needs to be shielded from rain, dirt or sunlight etc..

Given the specifications of the Fotonic B70 -the TOF camera which was available for testing during the project- and the task of detecting objects in the near vicinity of the truck, it was found that an as high position as possible with the camera pointing downwards would be the most appropriate, since this covers as large an area as possible. The position most elevated while being somewhat protected from the elements and dirt was found to be above the windshield, beneath the sunshield. Figure 2.1 illustrates the mounting position. To also cover the sides of the truck an additional sensor with similar or better specifications would be needed. In theory, with the camera positioned high up in a front corner of the truck, see Figure 2.2, both the front of the truck and one of the sides could be covered by the single sensor, however this would demand a broader FOV (and possibly higher sensor resolution to compensate for this) and is therefore not appropriate for the B70 camera. In the proposed set-up the FOV just barely covers the width of the truck with a margin of about 50 centimetres to each side.

[1] However, one automotive supplier has recently released a stereo vision system for cars that can detect pedestrians and other vehicles [11] and a Japanese car manufacturer is on the verge of putting cars on the market with another stereo vision system [13].

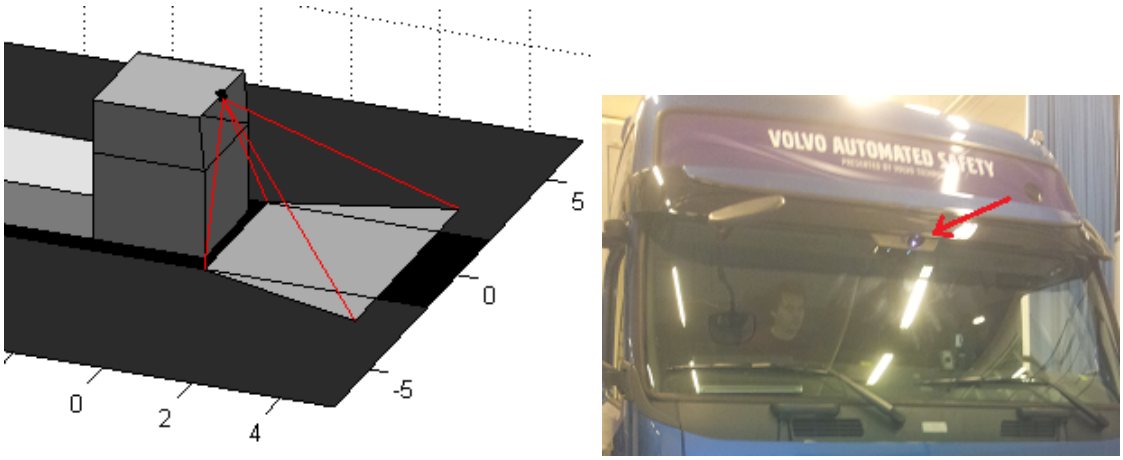



Figure 2.1: An example of where a camera could be placed. Left picture: the red lines show the edges of the field of view, which represents the Fotonic B70's FOV. Right picture: this position is located under the sun shield which means it is somewhat shielded from direct sunlight and also dirt and rain.

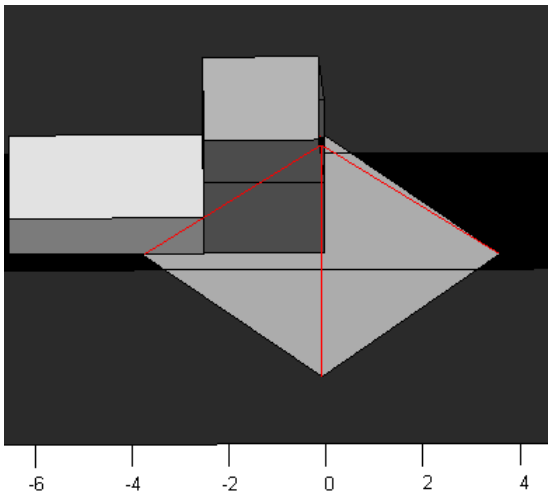



Figure 2.2: An example of where a camera with a broader FOV than the cameras available during the project could be placed in order to cover both the areas in front of and to the side of the camera.

The functionality that can be achieved depending on the FOV of the camera can simply be analyzed by looking at the stopping distance for different vehicle speeds and reaction times of the system. The distance travelled from the time of first detection of an object until the vehicle is brought to a halt is the minimum forward range at which object detection and path prediction is required.

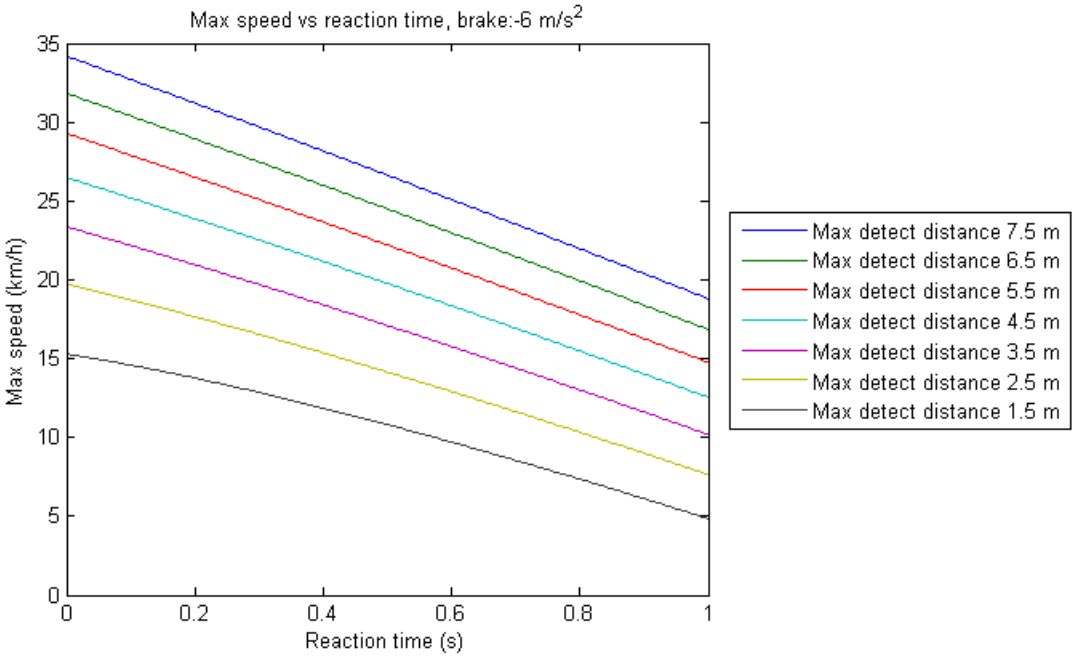



Figure 2.3: The vehicle speeds that can be reduced to zero given different ranges at which objects are detected, as a function of the reaction time of the detection system. This stems from the stopping distance at various initial speeds. The graph can be interpreted as, for example, given a detection range (maximum) of 6.5 metres and a reaction time, from the instance when the object was 6.5 metres away, of 0.6 seconds the highest speed at which a successful stop can be achieved (by braking with 6 $m/s^2$) is $\approx$ 23 km/h. Alternatively this could be interpreted as: at 23 km/h and the object appears 6.5 metres in front of the truck, the system needs to detect and react to the object within 0.6 seconds.

Judging by Figure 2.3 and the theoretical placing proposed in Figure 2.1 a system based on the B70 sensor could enable effective automatic braking at speeds up to roughly 20 km/h. However, in reality, reliable detection can not be performed in the far end of the cameras FOV due to unreliable measurements at those distances, which is later shown in the evaluation (Figure 6.5).

## 2.2 Time of flight sensor

Figure 2.4: Fotonic B70, the time of flight sensor used in this project.

Time of flight ranging is based on the known speed of light. By finding the time it takes for a light pulse to travel from the light source to the sensor via a reflecting object, the distance to the object can be calculated.

### 2.2.1 Theory

In its simplest form, the TOF camera measures distances by measuring the time delay of emitted light pulses reflected back onto the sensor. The time shift is found by integrating the received light during two different time intervals. This can be realised by a photo diode converting the received light to a current and the electrons being stored in different capacitors during the different time intervals. The difference in voltage between the capacitors indicates when the light pulse was received in relation to when the switch between the different capacitors took place. This process is depicted in Figure 2.5. Background light will affect the current from the photo diode resulting in an offset in the signals, and this offset can be found by repeating the procedure but without emitting any light. If the background light is too strong, however, the capacitors saturate and thus the distance can not be determined. The whole measurement process is repeated several times to form one distance measurement to make it more reliable - similarly as an averaging filter reduces noise.

TOF distance measuring can alternatively, and probably more accurately, be done by modulating the emitted light intensity as a continuous wave instead of pulses, see Figure 2.6 (or an arbitrary periodic function), and measuring the received intensity at several different times, which will depict a phase shifted wave plus background light. This is how the B70 works and the method is described in detail in [1] and [2]. The following section is, however, analogous for both TOF techniques.

Besides the depth map, the sensor measures the total reflected light, referred to as *active brightness*, providing a monochrome image of the scene.

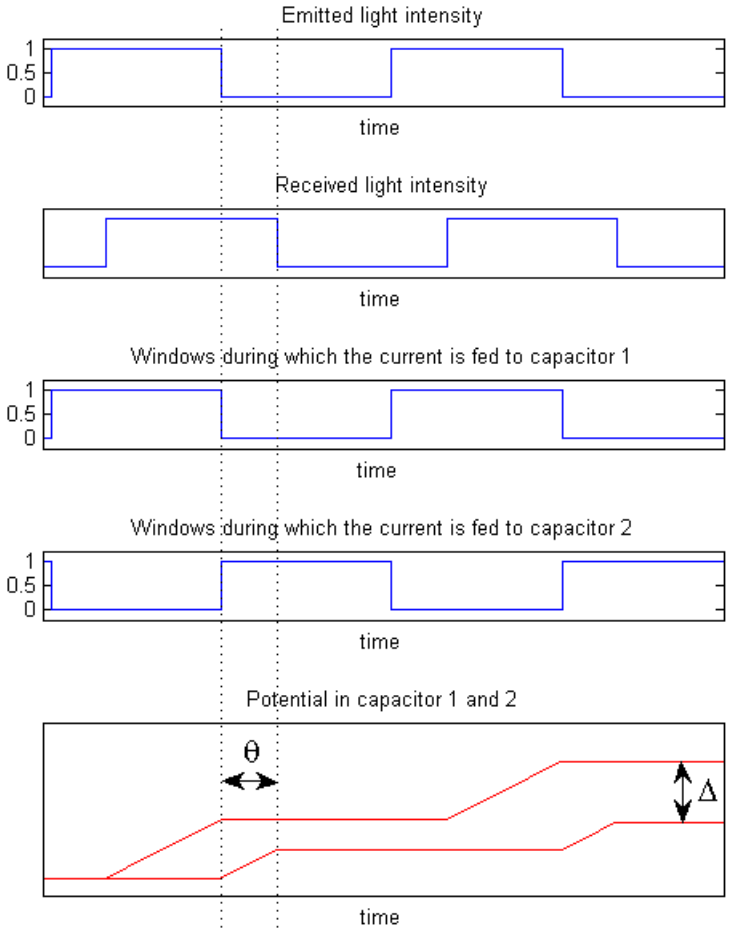



Figure 2.5: The simplest measurement principle of a TOF camera pixel. Each pixel consists of a photo diode connected to two different memory elements, i.e. capacitors. The difference in voltage, $\Delta$, between the two capacitors is a function of the time shift (or delay), $\theta$, of the reflected light which in turn depends on the distance travelled by the emitted light. Background illumination is disregarded here.

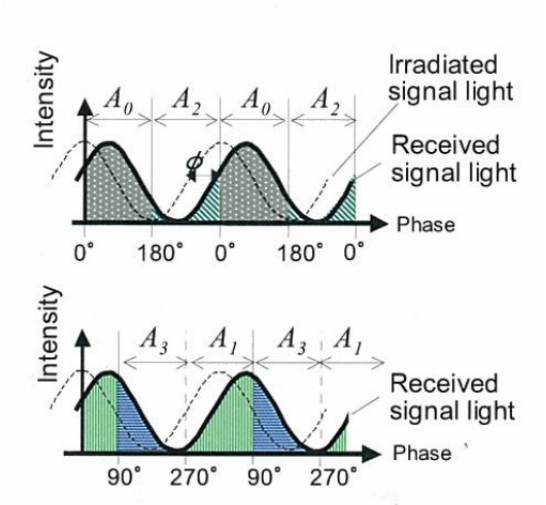



Figure 2.6: This figure shows an example of how the emitted light intensity can be modulated and what is reflected (background illumination is disregarded). This image is borrowed from [1].

### 2.2.2 Issues with time of flight cameras

The TOF ranging method suffers from a few inherent limitations, of which the most important ones are described below.

#### Wrapping

The emitted light intensity is modulated (continuously or as pulses) with a frequency $f_{mod}$, corresponding to the (intensity) wavelength $\lambda$. Since the light pulse travels back and forth from the camera via a reflecting object, the maximum distance that can be measured without ambiguities is $R_{max} = \lambda/2$. The periodicity of the emitted light intensity, and the measuring thereof, give rise to the ambiguity, a phenomena often called wrapping. This is due to the fact that the reflected light from an object beyond $R_{max}$ will reach the sensor after the consecutive intensity measurement has been initiated which results in a distance measurement, $r$, that is the true distance minus a multiple of $R_{max}$, causing objects to appear closer than they are in reality:

$$r = mod(R_{true}, R_{max}). \tag{2.1}$$

Figure 2.7 shows the wrapping by plotting the (filtered) distance values of in the depth image, with the camera at a slanted angle towards a flat white surface. One can see that the measurements are practically unusable beyond the second wrapping (pixels above 90).

There are ways to determine in which distance range the true distance for a measurement lies, e.g. employing multiple frequency modulation and phase unwrapping algorithms [17], this may however not be applicable to the pulse modulation technique (which could therefore be a reason for why one would prefer a continuous intensity modulation).

#### Interference

Interference may be a problem if this type of sensor would become common in future road vehicles. In the case where oncoming traffic passes close by, and the FOV overlaps another sensors FOV, the illumination of the passer-by could interfere with the camera's illumination, either directly or by reflection, causing false distance measurements. By varying the modulation frequencies in different cameras the problem may be reduced, but information on how big a difference in frequency that is required has not been found[2].

[2]Volvo cars have a laser ranging device and interference has been experienced with laser-based speed measuring instruments, such as the ones used by police [`http://nrk.no/nyheter/distrikt/troms_og_finnmark/1.8404362`, found 2013-01-23].

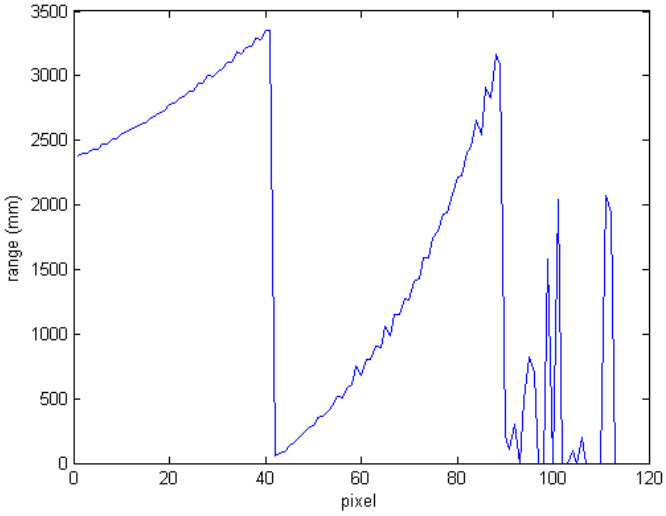



Figure 2.7: Depth measurements of a flat surface, taken from one depth image, showing the wrapping phenomena (occurs at $\approx 3300$ mm and 6600 mm). The true range values vary monotonously from $\approx 2.3$ m to $\approx 8.7$ m. What is interesting here is not the specific relationship between pixel and distance value (which is different for every geometrically different scene, camera position and angle) - this is described in section 2.2.4 - but simply that the measurement values do not form a monotonously increasing curve, as the true distances do. Instead they are subjected to equation 2.1.

### Background light and external light

As previously mentioned, background light may pose problems for the sensor, compare Figure 2.11 and 2.12 which show a higher uncertainty in the measurements in the presence of stronger background light. Often the background light is soft and evenly distributed over the scene. However, cases that may give rise to more serious issues, such as false detection, may occur when light with relatively high intensity reaches the sensors significantly and thus lowering the signal-to-noise ratio. This could be, for example, reflection of the sun in puddles of water or vehicle wind shields. Figure 2.8 shows how the sensor perceives a puddle in which the sun is reflected toward the sensor. The puddle lies in the ground plane, but the sensor gives values between $\approx 500 - 1500$ mm above ground and could perhaps be interpreted as a car. However, without having put extensive work into the problem we believe this issue could be mitigated by measuring local standard deviation of the depth measurements, which for a puddle reflecting the sunlight seems to be relatively high compared to an object reflecting general background light combined with the sensor's light. Removing pixels, and their surrounding, with brightness above a value is also a technique we have briefly experimented with that showed promising results.

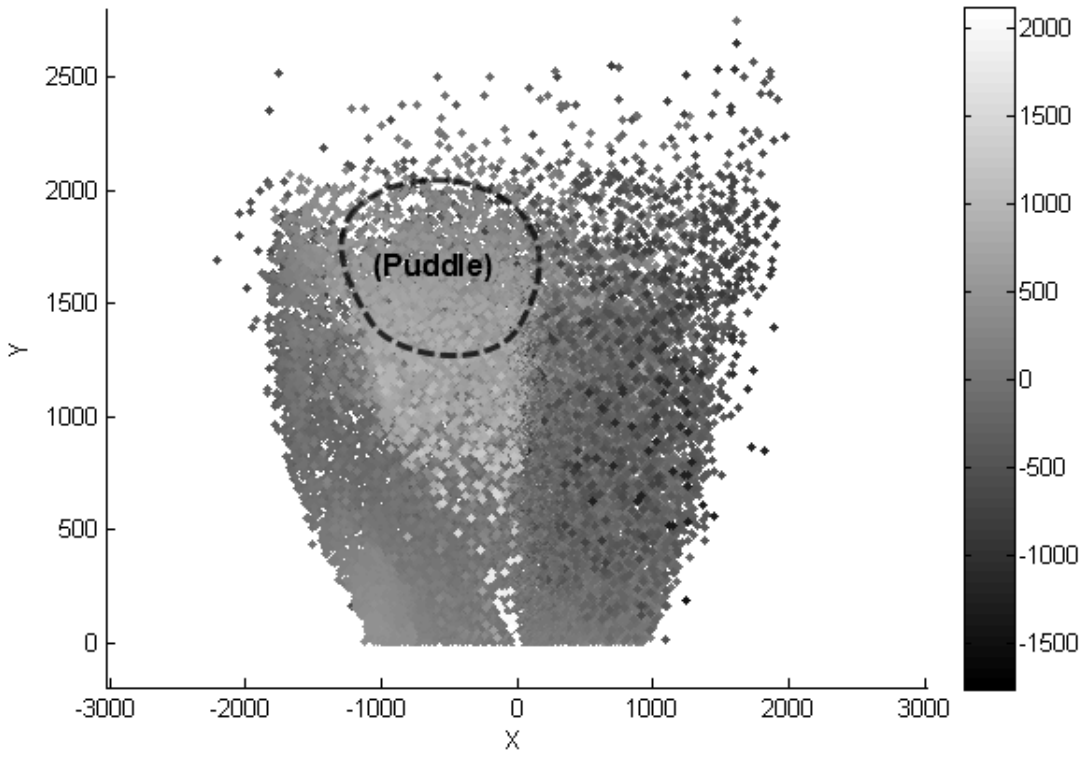



Figure 2.8: 3D coordinates of measurements when measuring flat ground with a small pool of water with sunlight reflecting in it. The brightness of the gray scale indicates the height above ground for each of the coordinates.

### 2.2.3 Advantages with time of flight cameras

#### Speed

The Fotonic B70 boasts a frame rate of 54 frames per second (FPS) with a resolution of 160x120 pixels, and there are cameras that are even faster and with higher resolution, at least under development - up to 100 FPS and with 100 000 pixels[3]. A higher frame rate means a shorter delay before new objects can be detected, increased accuracy of object speed estimation, greater certainty of objects existence or non-existence over time etc..

#### Accuracy

Inside the maximum distance range and for most materials, the depth measurements are relatively accurate, with a standard deviation in the range of a few centimetres at distances within meters although increasing in conditions with more background light. See the following section for a quantitative analysis.

#### Mechanically robust

Compared to stereo cameras and laser scanning devices the TOF camera can be made more robust in it's construction since it has no moving parts and nor is it particularly sensitive to the relative placement of the sensor and light source, as is the case with the two cameras used in stereo vision. The light source and sensor can be mounted in separate places as long as the reflected light can reach the sensor, however calibration will be needed to compensate for the offset in radial distance to the scene.

[3] `http://www.all-electronics.de/texte/anzeigen/46428/Mehr-Funktionalitaet-integrieren`

### 2.2.4 Measurement accuracy evaluation

This section will contain a brief investigation of how accurate the Fotonic B70 is in different conditions.

To investigate the accuracy of camera in the best of conditions, it was once more places straight towards a black object at different distances and using different shutter times, and 100 frames captured for each set-up. From these values, taken only from the black object, a standard deviation and a mean are calculated. This procedure was done twice; indoors, and outdoors in a shadowed area with more infrared background light. For the outdoor measurements, the number of zero-valued brightness pixels is large, especially for the two larger distances, and those range measurements are thus removed. Plotting the remaining measurements, for all of the set-ups, the mean value on the horizontal axis and the standard deviation the vertical, gives a plot like the one seen in Figure 2.9 for indoors, and Figure 2.10 for outdoors. Note the scale of the $y$-axis being different for the outdoor and indoor series of graphs. What is also interesting is the significant increase of fluctuation for the outdoor data. The standard deviation increases over ten times for the three smallest distances, while the fraction zero-valued pixels becomes so large that measurements for the two largest distances become almost totally unreliable. Distributions of measurements of a black object inside and outdoors can be seen in Figure 2.11 and 2.12, respectively.

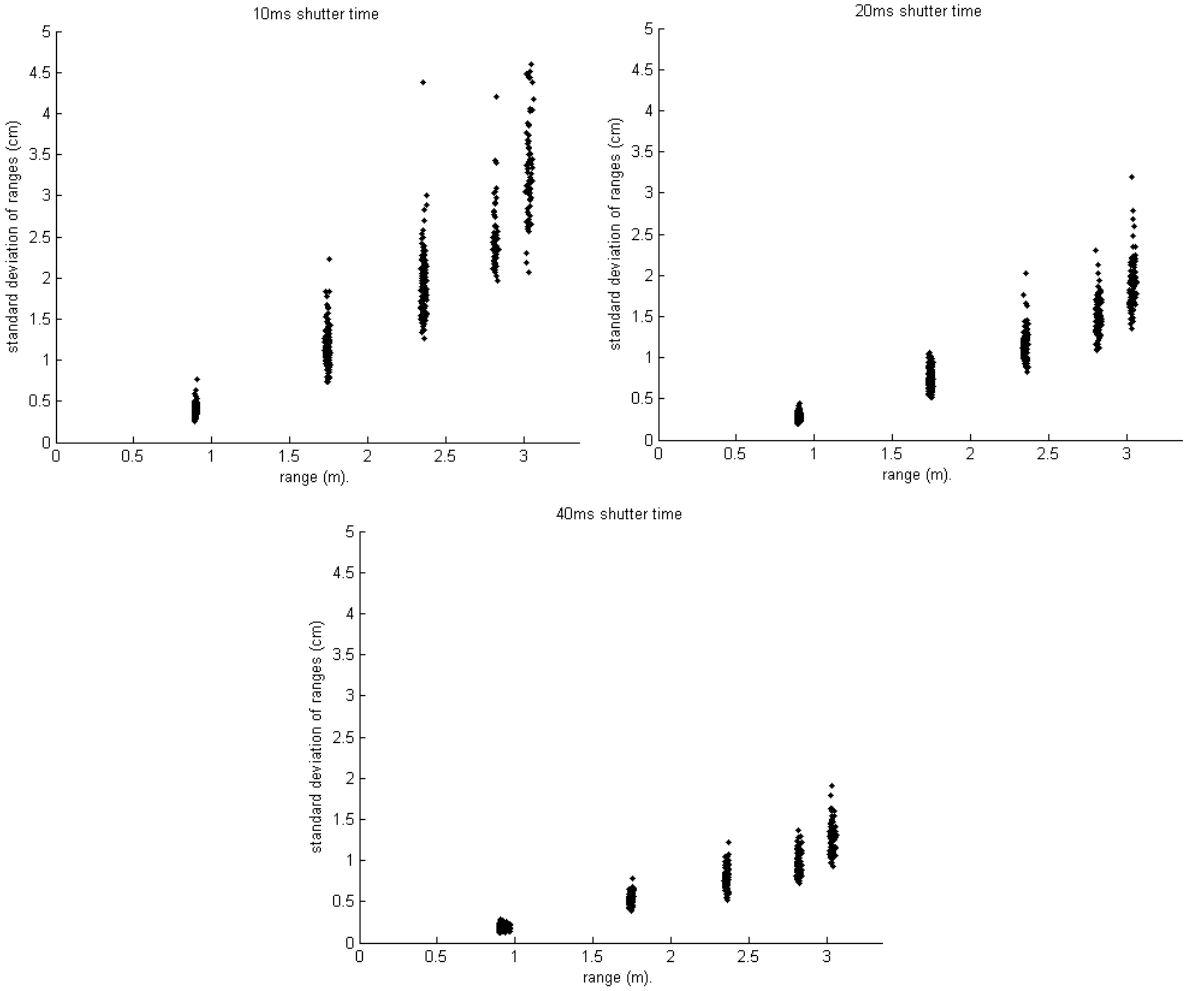



Figure 2.9: The standard deviation for each pixel at different distances to a flat (static) wall formed from a sequence of 100 depth images, for three different shutter times, indoors.

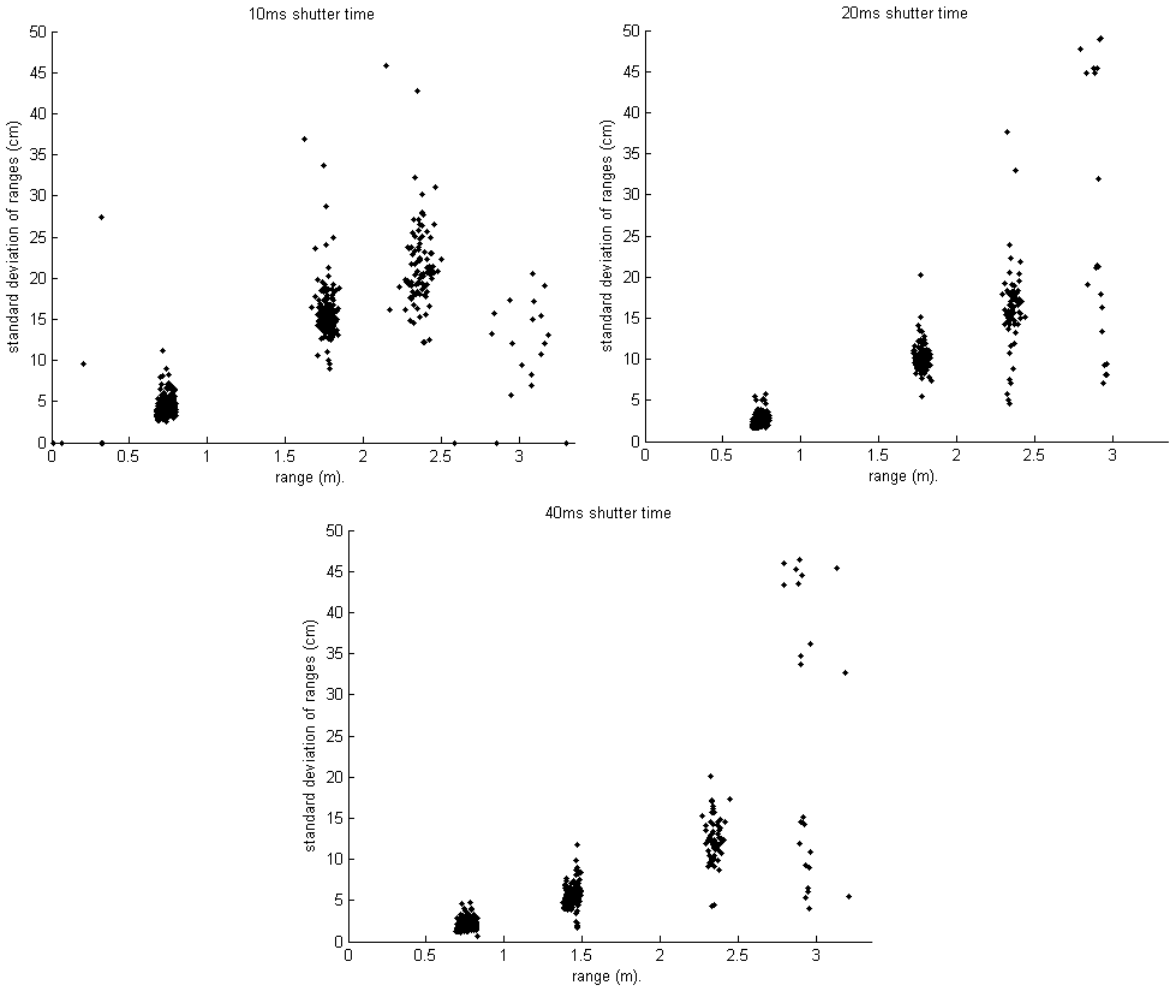



Figure 2.10: The standard deviation for each pixel at different distances to a flat (static) wall formed from a sequence of 100 depth images, for three different shutter times, outdoors.

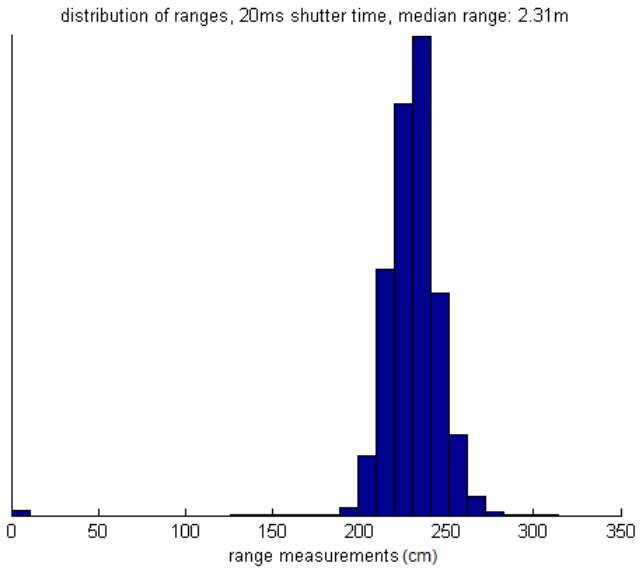



Figure 2.11: An example of a distribution of ranges around its mean values, measured indoors. The standard deviations for different ranges and with different sensor settings indoors are shown in Figure 2.9.

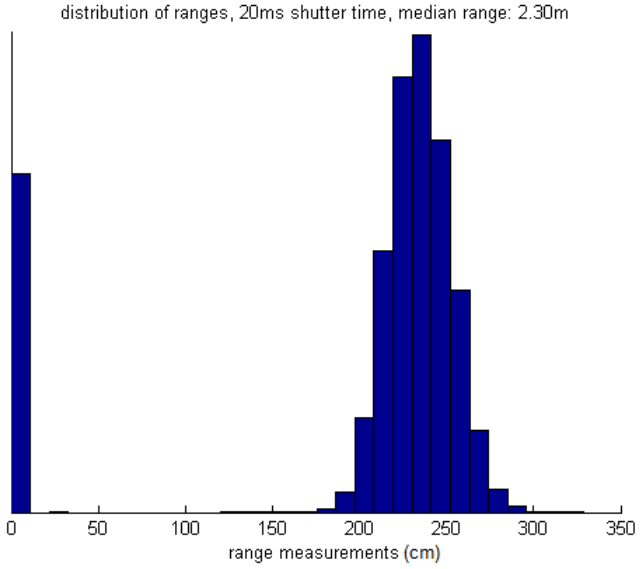



Figure 2.12: An example of a distribution of ranges around its mean values, measured outdoors. Note that the standard deviation has increased somewhat and the number of zero-valued measurements has increased significantly, compared to Figure 2.11. The standard deviations for different ranges and with different sensor settings outdoors are shown in Figure 2.10.

## 2.3 Stereo camera

Since studying stereo cameras have been a secondary objective in our work, we have taken a more qualitative approach. Nonetheless, the technology may play an important role in vehicle environment perception in the

future [18].

### 2.3.1 Theory

Stereo vision systems use triangulation to find the distance to objects. Triangulation uses the difference between the angular postion of an object relative to two different points along a straight line. The angles, $\alpha$ and $\beta$, can be measured with e.g. cameras or telescopes. Assuming that the distance between the measuring devices, referred to as the baseline, is known the distance to the object can be calculated from the respective angles between the measuring devices and the baseline according to equation 2.2:

$$d = \frac{b}{1/\tan(\alpha) + 1/\tan(\beta)} \tag{2.2}$$

where $d$ is the distance to an object, $b$ is the distance between the cameras, also called baseline of the cameras, and $\alpha$ and $\beta$ are the angles to the object from the two cameras respectively [19].

When using cameras for this task they need to be initially calibrated and their images rectified. Calibration is done to ensure that the cameras produce images where lines in the scene have the same angle in both sets of images. Rectified images means that a straight line in the scene remains straight in both images. Furthermore, to simplify the process of calculating the depths, a point in the scene should be located at either the same $x$ or $y$ coordinate (column or row) in the image, depending on whether the camera pair is placed on top of or beside each other.

The depth information is acquired by comparing pixels in one image to the corresponding pixels in the other, i.e. pixels that show the same part of the scene. This is referred to as finding the stereo correspondence. It is performed for windows of pixels, small parts of the image, at the same time. The correspondence for each pixel is therefore the correspondence of each pixel together with its neighbourhood of pixels within the window. Figure 2.13 contains an image pair taken by the stereo camera.

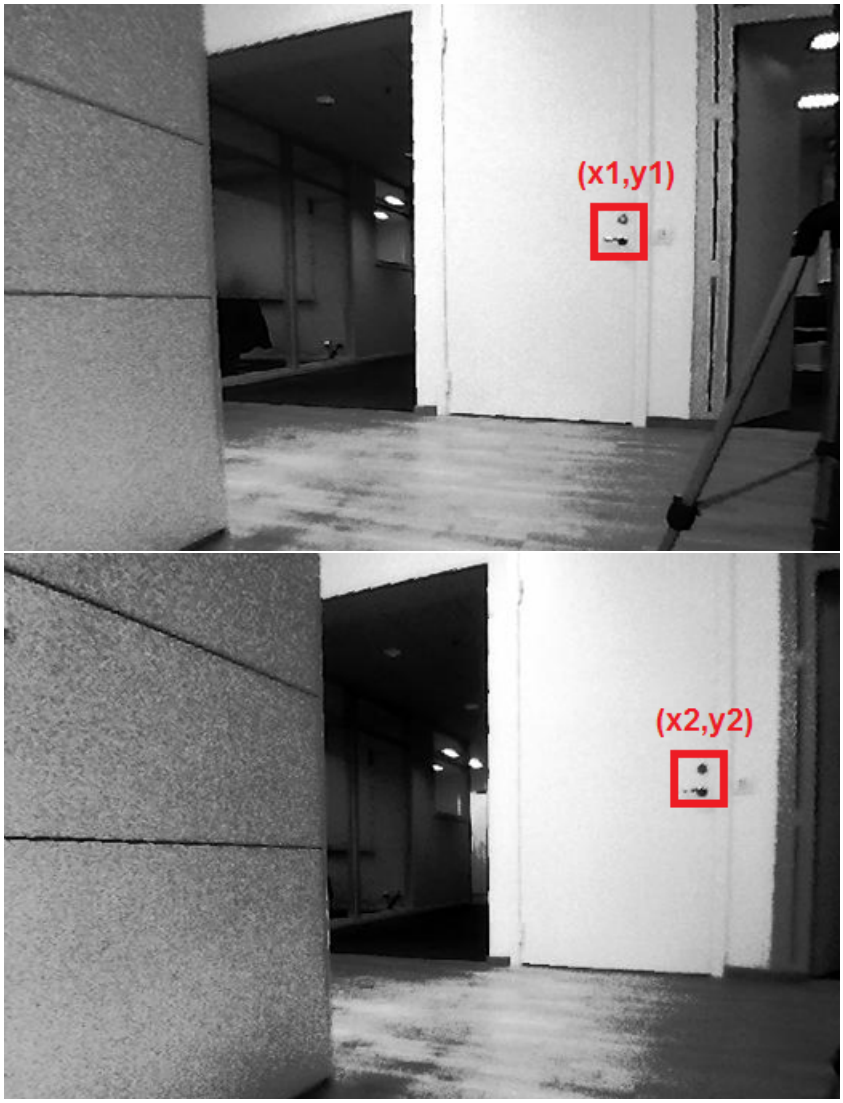



Figure 2.13: In this pair of images $y1 = y2$, since the cameras are side-by-side, while $x2 - x1$ gives the disparity.

A large window means that it is more likely to find a unique best correspondence, which makes a correct depth measurement possible, while the resolution of the depth map suffers. If the window is too large the result can be that no good correspondence is found since the two images will differ due to their disparate perspectives which means that objects appear differently. Thus smaller objects may not be measurable.

The pixel association is performed by calculating the correlation of windows from the first image with the second image along the axis of the two cameras displacement, with the step size of one pixel. The number of iterations that are performed sets the maximum disparity that can be found, i.e. the minimum distance that can be measured (objects close to the cameras have a higher disparity). The disparity, $d_{i,j}$, between two corresponding pixels is the pixel distance between them:

$$d_{i,j} = x_2 - x_1 \quad \texttt{or} \quad d_{i,j} = y_2 - y_1 \tag{2.3}$$

depending on if the cameras are mounted along the images $x$- or $y$-axis. The disparity is then used to calculate the depth, through a transfer function found empirically, i.e. another calibration. However the relation between the disparity and the distance can be derived from equation 2.2 if the exact value of the baseline and the cameras relative angles are known together with the angular resolution (pixel/deg) of the cameras.

### 2.3.2 Issues with stereo vision

#### Incorrect pixel association

In the usual case when there are objects in front of each other in a scene, parts of the occluded object may be visible from the perspective of one camera and not from the others. This means that some of the pixels in one image will not have matches in the other and the distance to the (partly) occluded object can not be determined.

Conversely, there may be several equally good matches if there are patterns in the scene, e.g. a fence or brick wall, but also a homogeneously coloured surface. If this is the case, the algorithm can only guess which are the corresponding pixels in the images. For this reason, algorithms can use a uniqueness measure of the matchings enabling unsure matchings to be rejected. Instead, no measurement is provided and one has to rely on that the edges of the object can be distinguished from the background.

#### Computationally heavy

In contrast to TOF cameras, the sensor hardware is relatively simple, our stereo camera, for example, was built from two web cameras. The price is that it requires depth acquisition algorithms that place high demands on the processing hardware. However, the demands are not impossible to meet today; high speed stereo correspondence computation, reaching frequencies of above 100 Hz for 640x480 pixels, has been realised on hardware costing around 120 USD [3]. It should be mentioned, on the other hand, that in this specific implementation the maximum disparity was 64 pixels which is rather low, suitable only for short ranges.

#### Calibration sensitive

As mentioned earlier, the stereo system needs to be calibrated. The disparities which yield the depths depend both on relative position and angle of the cameras. Depending on how much relative movement of the cameras that can occur, the system will need to be calibrated, more or less, often. If the rig is completely rigid calibration will only have to be done once (and if the relative positions and angles can be exactly replicated, the same calibration parameters can be re-used for new systems). To quantify the possible error coming from change in relative angle one can look at the triangulation equation 2.2. By assuming that the object is equidistant to both cameras and that $\alpha = \beta$, differentiating with regard to $\alpha$ gives us

$$d'(\alpha) = \frac{b(1 + \tan^2(\alpha))}{2}. \tag{2.4}$$

Setting the baseline to 22 cm, which is appropriate for maximum ranges of more than 10 meters with $\approx$ 500 pixels along the horizontal FOV [20], when $\alpha$ is 88° the error becomes $d'(88) \approx 1.58$ meters/degree (compared to the corresponding distance of 3.15 meters).

### 2.3.3 Advantage with stereo vision

The main advantage seems to be its versatility. Any two, preferably of same type, cameras can be used for the task and therefore it is easy to accomplish the desired FOV and resolution.

Additionally, stereo vision systems do not have to rely on an active light source and work in nearly all lighting conditions, depending on the light sensitivity of the cameras. Provided that the truck has its headlights on the system will work in nighttime. Furthermore, they have no wrapping issues and nor will they suffer from interference from other sensors.

## 2.4 Camera models

For each depth image pixel, we define a corresponding vector in which direction their light source is assumed to lie. While the norm of these vectors are different, depending on which camera or whether the range value is wrapped, the direction of the camera vectors remain for the different models. The vectors' horizontal and vertical angles relative to the camera center are assumed to be evenly distributed in the FOV. Horizontally,

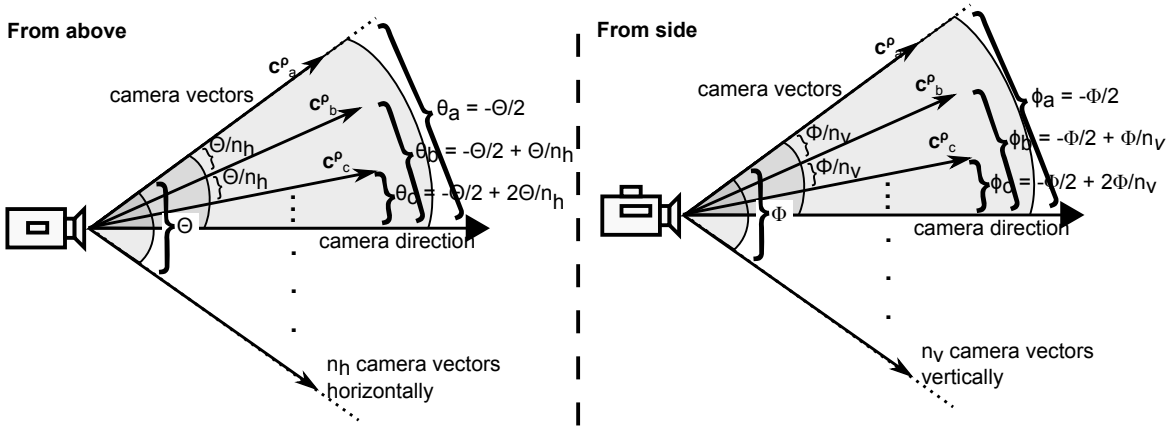



Figure 2.14: A depiction of the the horizontal and vertical camera vector angles.

the angle of the camera vector is denoted $\theta_i$ and spanning $[-\frac{\Theta_h}{2}, \frac{\Theta_h}{2}]$, while $\phi_i$ is short for the vertical angle and similarly spans $[-\frac{\Phi_v}{2}, \frac{\Phi_v}{2}]$. A depiction of the camera vector orientation can be found in Figure 2.14.

### 2.4.1 TOF sensor

The Fotonic B70's depth image, e.g. as seen in Figure 2.15 shows that there is need for two different depth camera models to describe the depth value shown within and outside the first wrapping; the equi-depth curves are clearly horizontal lines for real-world ranges below the maximum range, and further away the ranges seem to more reflect their actual distances to the camera, if one compensates for the wrapping. From the observed unwrapped depth profile we assume that the ranges are the real-world distance in the camera direction, or in other words: the length of the projection of the camera-source vector onto the camera direction vector. In this case, a picture is surely needed for clarification, see Figure 2.16.

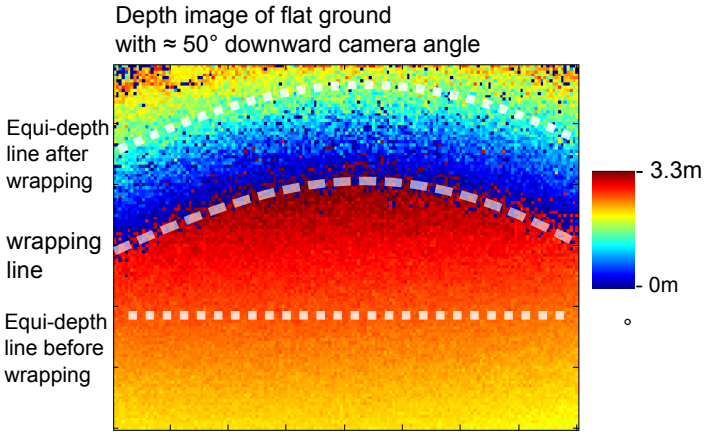



Figure 2.15: A depth image of flat ground at a slanted angle. It clearly demonstrates the wrapping and the two different camera models.

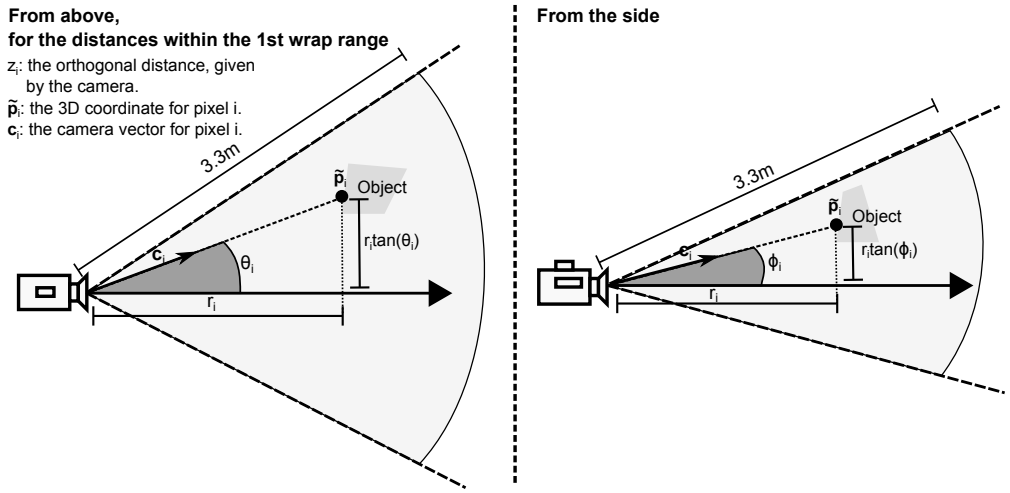



Figure 2.16: For ranges within the first wrapping, the figure demonstrates the relation between a 3D coordinate $\widetilde{\mathbf{p}}_i$ from its source pixel with range value $r_i$, the camera vector $\mathbf{c}_i$ and its horizontal angle from the camera vector. The relationship is the same for the vertical angle of a camera vector.

Due to these different depth profiles, it is of need for different sets of camera vectors to translate the depth values to real-world 3D coordinates. The vectors for the unwrapped distances which give an accurate representation are:

$$\begin{aligned} \delta x_i &= \tan(\theta_i) \\ \delta y_i &= \tan(\phi_i) \\ \delta z_i &= -1 \end{aligned}$$

$$\mathbf{c}_i = (\delta x_i, \delta y_i, \delta z_i)$$

While the underlying model for the wrapped distances is uncertain, we use normalized vectors as a rough first approximation:

$$\mathbf{c}_i^{\rho} = \|\mathbf{c}_i\|_2^{-1} \mathbf{c}_i \tag{2.5}$$

# Chapter 3

# Choice of algorithms

As briefly mentioned in the introduction, we need robust algorithms that can adequately detect and track objects given the data provided from the 3D camera and they need to run in real-time. Given the large amount of data in the depth (and active brightness) images provided by the sensors, the task is to reduce this data to a list of targets and their relevant attributes (e.g. position, size, velocity) which in turn can be communicated to, for example, the decision making part of the safety system.

The purpose of this chapter is to describe the basis for the algorithm designs and to motivate the choices made in key aspects of ditto. The algorithms are then described in depth in the two subsequent chapters.

## 3.1 Interpreting input data

A way to achieve fast algorithms is to reduce the data in each input image to a form of smaller size. The approach which, to the authors, seemed most sensible was to transform the high resolution depth and brightness images to a two-dimensional *occupancy grid*. The basic idea is to express the field of view projected on the ground as a grid of binary values where each indicates the existence of an object at that position. The fundamental assumption for this is that the trucks environment can be reduced to a plane in which objects can exist, and their relevant movement occurs, without losing important information.

The occupancy grid is formed by counting the number of detected spatial points that lie within each grid cell (disregarding the $z$ dimension (i.e. vertically)), i.e. forming a spatial two-dimensional histogram. The histogram is mapped to a binary occupancy grid by setting each cell to 1 if the bin count is above a specified threshold and 0 if it is below. Figure 3.1 depicts how an object is represented in an occupancy grid.

This translation has a linear algorithmic complexity; each pixel of the input images value needs only to be counted once.

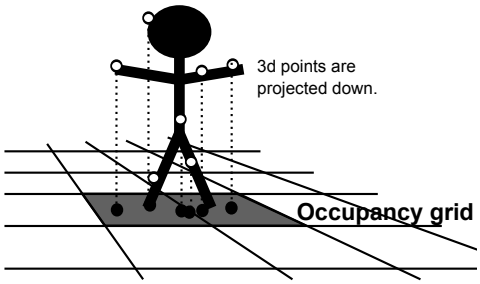



Figure 3.1: A depiction of how an occupation grid works.

As an example of how much the quantity of data can be reduced, if we assume that the physical resolution of 10 cm per pixel is accurate enough to describe the objects of interest: with a FOV of e.g. 5x4 meters, the occupancy grid will be 50x40 cells where each cell can be represented by one bit. Compared to an input

depth image of 160x120 pixels with two bytes per pixel (2 x 8 bits) the data is thus reduced by roughly 1500 times. This quantisation of data will impact the position resolution and hence the speed resolution of detected objects negatively, but the consequences are expected to be modest because the position is typically defined as the center of mass which can have sub-pixel precision for objects occupying more than one cell.

An important difference between the camera set-up investigated in this project and the systems used for similar applications is that the ground is in the entire field of view because of the downward angle at which the 3D camera is mounted. This implies that where an object is absent ground is instead detected (ideally), which is not true for an horizontal set-up where there could simply be lack of information if no object is within range. One can utilize this additional information by attributing measurements to objects with more certainty; if two occupied cell areas are separated by what is certain to be ground one can determine with higher confidence that they are separate objects, see Figure 3.2. Our proposal is to utilize the ground information by clustering the occupancy grid cells into *detections*.

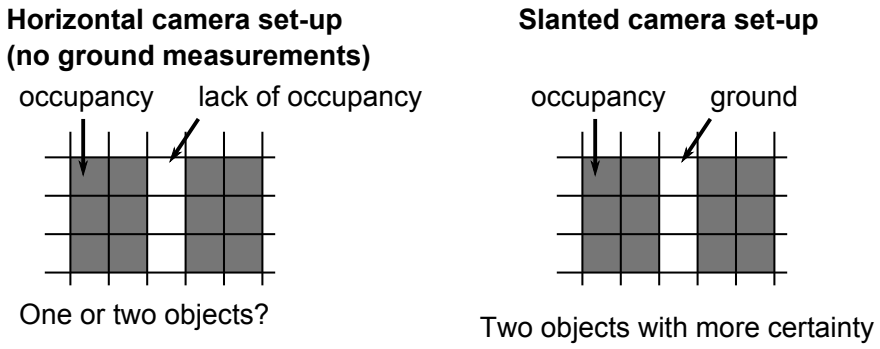



Figure 3.2: Ground information increases the certainty about the extent of an object.

The major drawback of a 2D occupancy grid, compared to, for instance a 3D occupancy grid, is the high propensity for different objects to coalesce. Two objects can seem to be one by simply having enough points close enough for them to occupy the same grid cell. This can certainly be an issue when the trajectories of pedestrians or bikes need to be estimated; if a target moves close to another they may become indistinguishable in the occupancy grid rendering the tracker uncertain of the objects trajectories.

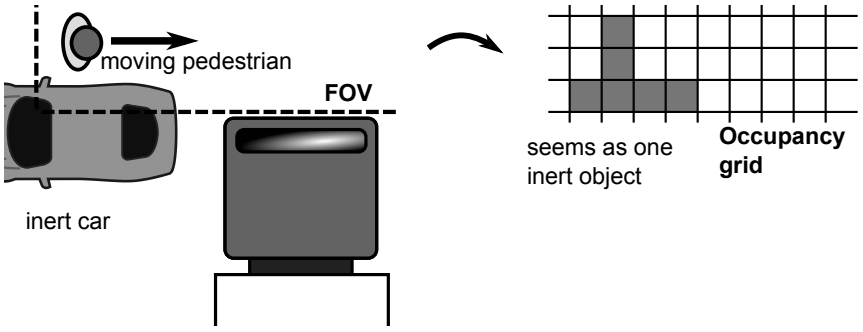



Figure 3.3: An example of how an occupancy grid could represent a traffic scene.

This can be remedied somewhat by increasing the resolution of the grid (at the cost of increasing computational load), but not fully since objects can overlap physically. Hence a smart method of separating objects that appear as adjacent on the grid is needed. Aside from the particle filter described in 5.2 which can distinguish different objects in neighbouring cells, we have developed two other approaches, which are described in 5.1.6.2.

## 3.2 Tracking algorithms

The choice of 2D occupancy grid as the core of the data representation, and the coalescence issues and their importance to the application lead us to the question of how to track objects, how to identify coalescing objects and how to separate them.

We have investigated three approaches to interpreting the occupancy grid, where one is a particle filter based on the work reported in [5] and the other two are based on the Kalman filter. The latter have a lot in common; the main difference is the target representation.

The first approach of the two Kalman filter-founded will be referred to as the *box-based* variant. It represents objects as boxes with an orientation in the two-dimensional occupancy grid. This seems reasonable because a box can be represented by a handful of variables while the shapes of many traffic-related objects are approximated quite well in the $x, y$-plane as boxes; pedestrians, cars and bikes all fill up quite convex areas when seen from above.

In the second variant, denoted *cell-based* we make no assumptions of the shape of a target; the boxes of the previous algorithm are replaced with the number of cells occupied by the object and orientation is disregarded. This is possibly an even better fitting representation of pedestrians due to the fact that their shape can vary if only the surface facing the camera is perceived. This means that the appearance of a given object depends on its orientation relative to the camera.

The mentioned variables, longitudinal and lateral position $(x, y)$, together with the velocities of the targets $(\dot{x}, \dot{y})$ in these dimensions , collected into the *state vector*

$$\mathbf{x}_{pos} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

, and

$$\mathbf{x}_{size} = \begin{bmatrix} w \\ d \\ h \end{bmatrix} \text{ or } \begin{bmatrix} n_c \\ h \end{bmatrix}$$

where $w$ is the width, $d$ the depth, $h$ the height of the target, and $n_c$ the number of cells in the occupancy grid it consists of. These state variables are then estimated and tracked. As established in section 2, the data is quite noisy in the presence of sunlight (even when cloudy), and the tracking should preferably filter it somehow. Effective filtering algorithms usually need an underlying model of the data, from which one can make assumptions and do predictions. Appropriately complex models for object size and position are thus of need.

The assumption that an object is constant in size seems to reflect the measurements well for most types of objects except pedestrians, but using another size model for pedestrians is considered unnecessarily complicated, and a constant model is indeed linear. Regarding the motion of a pedestrians, it can, naturally, be non-linear, but for the vast majority of cases, especially in uncongested areas, people move in straight lines. We therefore also consider a linear motion model to suffice.

A filter algorithm widely used since the 60's is the Kalman filter which optimally estimates the state of a linear system from data with normally distributed error. As the measurement noise for the TOF sensor at hand is similar to a normal distribution, judging by Figure 2.9 and 2.10, and the target position and size are estimated from these measurements, the approximation made is that these estimates also will fluctuate as a normal distribution. Thus the Kalman filter is a good fit; it is optimal for the models and noise and has a long and proven track record.

### 3.2.1 Data association

Data association is the procedure of associating tracks with measurements so that the history for each track can be used together with the new (noisy) information to form an estimate of the true state. There are a number of methods often mentioned in literature. The nearest neighbour (NN) association is simple; associating each target with the closest measurement. The distance can be measured in several ways; for

example, it can be expressed in probabilities or euclidean distances. The particle filter we implement in this project uses NN for association.

Neither *probabilistic data association* (PDA) or *joint probabilistic data association* (JPDA), two other frequently mentioned methods, seem appropriate. This is mainly due to the fact that they make an assumption that the number of measurements are larger than, or equal to, the number of targets. This in turn informs the basic structure of those algorithms. Our choice to form the tracking around our concept of *detections* makes, however, the number of detections always smaller than, or equal to, the number of targets. Here, it is the detections that need to be split up, not many measurements to be associated with a target, which is the general case for the mentioned algorithms. Furthermore, straightforwardly implementing these algorithms while using the cells of the occupancy grid as individual measurements is probably considered a by-the-book approach (and in hindsight, seems more fruitful than it did initially), but has one apparent disadvantage. These approaches would not take into consideration if cells were connected or not, and thus a measurements from separate connected areas (i.e. detections) could be associated with a single target - even if ground was determined to be between the connected areas. The connectedness could of course be taken into consideration somehow, but the issue was not pursued further than this.

Our decision was to pick a combinatorial approach where, which basically means that a likelihood is associated for each combination of data association, where multiple targets can be associated with a single detection. The potentially huge search space is shortened by a method called *branch-and-bound* which is described later. This data association algorithm was used for the two Kalman based tracking algorithms mentioned above.

# Chapter 4

# Data processing

The information received from the sensors are depth maps, i.e. images containing a distance measurement per pixel. The TOF camera also provides an active brightness image of the scene, a grey level image of the light that reaches the sensor. A typical stereo camera can also provide a grey level image, however this is not needed to determine the certainty of the measurements as for the TOF camera. Instead the uniqueness measure is integrated into the cameras internal data processing removing uncertain pixels before the depth map is provided.

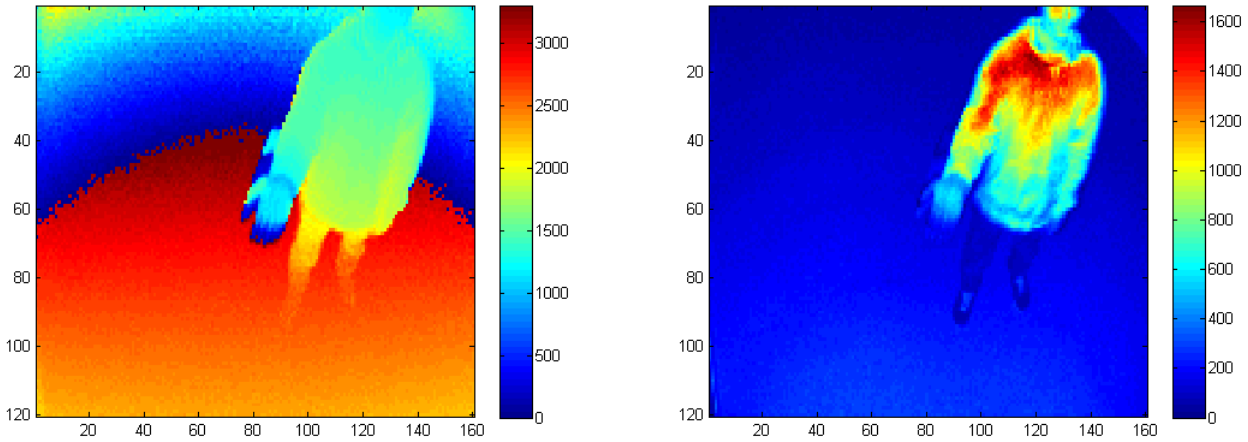



Figure 4.1: Samples of a depth (left) and active brightness image (right) of a person standing on a flat ground taken with the B70 TOF camera. The wrapping can be seen in the depth image as the ground changes from dark red to dark blue.

To detect objects in a depth map, along with their location and size, the distances need to be translated into 3D coordinates. When this is done, points with a significant height above an approximated ground level can be extracted, as they are potential object points, and then clustered into cohesive objects. The rest of the points are either regarded as ground or as invalid detections.

## 4.1 Overview of the process

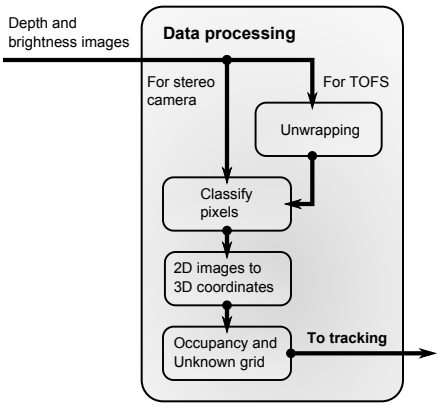



Figure 4.2: A schematic of the data processing.

The data processing algorithm is designed to take in the raw depth and brightness images sequentially and return a list of objects and their (statically measurable) properties including class. To give the reader a birds-eye view of this process, it will here be broken down and described very briefly, while the in-depth descriptions are given in the following sub-sections.

First off, to produce correctly orientated 3D data, with the $z$-axis perpendicular to the ground, the mapping between the arbitrarily rotated camera coordinate system and the real world coordinates needs to be found. This can be done by using the camera to measure flat ground when it is mounted in its position. In such a depth image a ground plane fit is found. The mapping is then derived from the ground plane's base vectors.

In the general case the depth images can be mapped to 3D space by using the camera model described in 2.4. For the Fotonic B70, however, there is a distortion in the normalised camera vectors $\mathbf{c}_i^\rho$ that needs to be corrected.

Now the set-up is done and the input data can be processed sequentially. At first, the wrapped depth image pixels are unwrapped, an operation that is exclusive to the TOFS. Then the depth image is converted to 3D coordinates by using the two sets of camera vectors for the wrapped and unwrapped pixels respectively. The resulting coordinates are then adjusted according to the ground calibration by translation and projecting them on the ground plane's base and normal vectors.

The detected points in 3D space reflect the real-world points of all possible types of objects and we have chosen to split them into three categories: object, ground or uncertain depending on their altitude from ground and brightness. Points with low active brightness are regarded as uncertain since the measurements are based on little reflected light.

The occupancy grid is then formed. To define which areas are lacking (valid) sensor data a similar grid which we call an unknown grid is also formed. In this grid, cells which have too few valid measurements within them, e.g. areas occluded by objects or areas containing surfaces with vary low reflectivity. It is these two grids accompanied by a height value per grid cell, defined as the maximum height value for each cell, that comprise the data which the tracking then is based on.

## 4.2 Camera calibration

### 4.2.1 Estimating the real world orthonormal vectors in the (implicit) sensor space

In order to represent the input as correctly rotated spatial points, where flat ground lies in the $xy$-plane in the vehicle space, the position and angles of the camera relative to the ground are needed. Lacking this information the rotation and translation of the measured space can be done if the vectors spanning the ground plane in the sensor space are known.

#### 4.2.1.1 Ground plane

If, as we assume, the camera has at all times a fixed position and direction relative to the ground plane, then a static ground plane suffices as ground model. This plane is derived from a depth image of flat ground. The math is fairly straightforward; let the ground plane be defined by its normal vector $\widetilde{\mathbf{n}}$ so that the following equation:

$$\widetilde{\mathbf{n}} \cdot (\mathbf{x} - \widetilde{B}) = 0 \tag{4.1}$$

holds for all points $\mathbf{x}$ in the plane with $\widetilde{B}$ being a arbitrary reference point also in the plane. Now a good fit to the observed data can be found by solving the system of equations, where $\mathbb{G}$ is the set of ground points and $\mathbb{W}$ is the set of wrapped points of ground data:

$$\widetilde{\mathbf{n}} \cdot (\widetilde{\mathbf{p}_i} - \widetilde{B}) = 0 \ , \ \forall \widetilde{\mathbf{p}_i} \in \mathbb{G} \backslash \mathbb{W} \tag{4.2}$$

where the normal is estimated to be the vector that minimizes the squared error of the model to the training data. A point, $\widetilde{B}$, on the ground is required and the mean of all $\mathbf{p}_i$ is a good choice for this because the ground is assumed to be flat and sensor noise is assumed to be gaussian with zero mean, shown in 2.

Two orthonormal base vectors spanning the plane are found by first taking two additional points on the plane with arbitrary distinct $x$ and $y$ values and with them the $z$ values are found from the plane equation (4.1). Vectors are formed by subtracting the points with $\widetilde{B}$, and are then orthogonalised with the Gram-Schmidt method and normalised, giving $\hat{\mathbf{e}}_1$, $\hat{\mathbf{e}}_2$.

The base and normal vectors can now be used as a transform from the sensor space to the truck space:

$$T(\widetilde{U}) = \ [\hat{\mathbf{e}}_1 \quad \hat{\mathbf{e}}_2 \quad \widetilde{\mathbf{n}}]^T \ (\widetilde{U} - \widetilde{0}) \tag{4.3}$$

where $\widetilde{0}$ is the point in the sensor space which is transformed to the origin in the new space. This point is found by choosing the new origin to be the point on the ground directly beneath the camera, i.e the camera position in the sensor space projected onto the plane:

$$\widetilde{0} = [\ \hat{\mathbf{e}}_1 \quad \hat{\mathbf{e}}_2\ ]\big([\ \hat{\mathbf{e}}_1 \quad \hat{\mathbf{e}}_2\ ]^T(\widetilde{C^o} - \widetilde{B})\big) + \widetilde{B} \tag{4.4}$$

which is the camera coordinate in the sensor space (chosen to be $\mathbf{0}$, for simplicity) projected onto the ground plane.

The actual distance to the ground, $G_i^{gt}$ for each pixel $i$, is simply formed from an average over a number of captured frames to reduce noise.

### 4.2.2 Correcting camera vectors (for B70 TOFS specifically)

For some reason the Fotonic B70 TOF camera vectors, when applied to wrapped pixels after unwrapping, $\mathbf{c}_i^\rho$, render inaccurate geometries. Therefore they need to be adjusted slightly so that they convert the depth image to rectified spatial 3D coordinates. Assuming that multiplying each vector with a specific factor would correct the camera vectors, we simply use the measured ground and the theoretical ground distances as a basis for the correction. To make the camera vectors map the measured depth values to the ground plane when transformed to spatial coordinates we multiply them with a factor $\frac{G_i^\rho}{G_i^{gt}}$ giving:

$$\mathbf{c}_i^f = \frac{G_i^\rho}{G_i^{gt}} \mathbf{c}_i^\rho$$

so $G_i^{gt}\mathbf{c}_i^f = G_i^\rho \mathbf{c}_i^\rho$, i.e. mapping the distorted spatial coordinates to the ground plane.

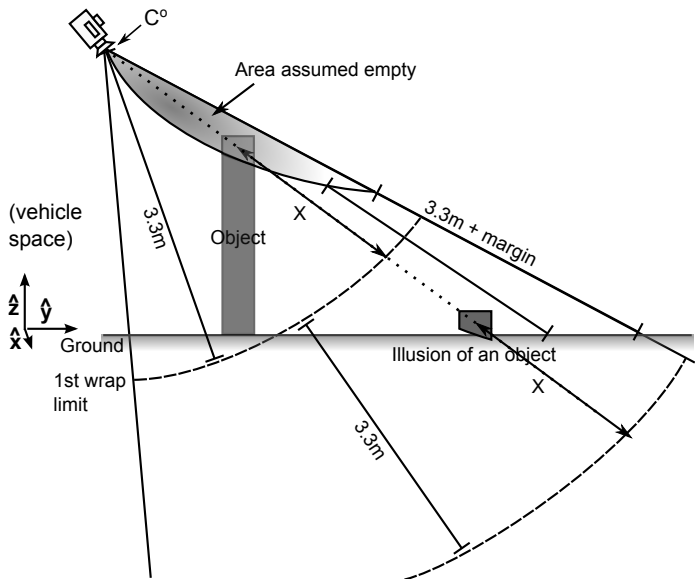



Figure 4.3: Image showing how the unwrapping is done. Since the distance to the ground is known and objects can not exist beneath the ground there is a volume within the FOV that can be unwrapped with a realistic outcome. The idea behind this is that all measurements, and only these, that after unwrapping are still above the known ground plane are unwrapped. This implies that areas of the FOV where the ground is within the wrapping distance will never be unwrapped, while measurements within the "area assumed empty" will always be unwrapped. As mentioned, 3.3 meters is the wrapping distance of the Fotonic B70.

## 4.3 Live processing

### 4.3.1 Unwrapping (for TOFS only)

As mentioned in section 2 the TOF sensor provides ambiguous range measurements, which is visible in Figure 2.7. The measurements therefore need to be so-called unwrapped, i.e. shifted by one or more ambiguity ranges to reflect the true values.

The theoretical mapping between the true distance and the measurement provided by a sensor was shown shown in equation 2.1:

$$r = modulus(r_{true}, R_{max})$$

Although, as mentioned, this is not quite true for the Fotonic B70, and it is therefore the measurement correction described in section 4.2.2 is applied.

Since the distance to the ground is known (from calibration) for each pixel in the image, assuming that the ground is flat, it is possible to remap each distance measurement so that it reflects a point in the vehicle space above or on the ground but not underneath the ground plane since this is not plausible under the reigning assumption. Of course, for each pixel, this re-mapping produces a range, with lower end at distance 0, in which measurements are always mapped to where its distance is erroneously unwrapped. This area is sketched in Figure 4.3. The transfer function is then:

$$f_i(r_i) = G_i^{gt} - 3.3 + r_i \tag{4.5}$$

and is applied to all pixels where $r_i + m_r < G_i^{gt} - 3.3$, where $m_r$ is a positively valued margin in the range of 15-40 cm depending on the accuracy of the measurements. A more noisy scene could benefit from a larger margin, and thus less pixels would be unwrapped falsely from objects with a distance within 3.3 meters from ground at the cost of increasing the area assumed empty.

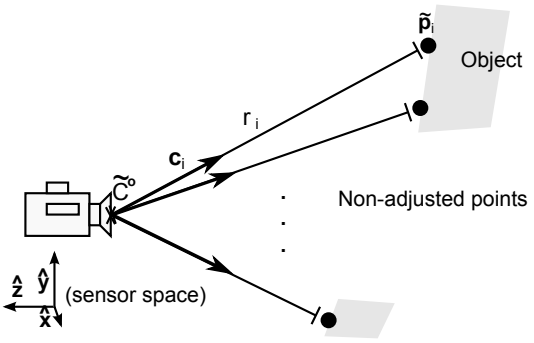



Figure 4.4: A sketch of the camera vectors and their measured distances in the sensor space, i.e. the space relative to the camera orientation.

### 4.3.2 Converting depth image to 3D coordinates

After unwrapping the wrapped distance measurements in the depth map the next step is to transform its pixel values $r_i$ to coordinates in the vehicle space. Firstly, the depth values are transformed into non-adjusted 3D coordinates, $\widetilde{\mathbf{p}}_i$. The camera vectors $\mathbf{c}_i$ are constructed so that this transformation is:

$$\widetilde{\mathbf{p}}_i = \widetilde{C^o} + r_i \mathbf{c}_i$$

where $\widetilde{C^o}$ is the position of the camera in the non-adjusted space, and is, as already mentioned, for simplicity set to $\mathbf{0}$. For a graphic depiction of these relations, see Figure 4.4.

To transfer the non-adjusted coordinates (in the sensor space) to the vehicle space, the transform $T(.)$, equation (4.3), is applied on all $\widetilde{\mathbf{p}}_i$ and $\widetilde{C^o}$ giving $\mathbf{p}_i$ and $C^o$.

### 4.3.3 Classifying pixels

As each pixel in the depth map is a range value to a reflecting point we have chosen to refer to each distance measurement as a pixel, i.e. a ground pixel is a distance measurement identified to be from a point on the ground. We have identified three pixel classes which are useful; confident, ground and object.

#### Confident pixels

Confident pixels are pixels that have an active brightness level above a certain threshold which indicates that the basis for the distance measurement is strong and can thus be reliable. The threshold is found empirically by measuring black objects at the far end of the field of view. On the binary image formed by confident pixels a four-neighbour erode operation is formed to remove potential noise; each confident pixel which doesn't have four confident neighbours is made uncertain.

#### Ground pixels

As mentioned, one assumption we make when classifying pixels is that the ground is flat. In practice, what we mean by flat is that the measured height ($z$-value in the vehicle space) should not exceed a minimum value. A good value range, that allows for noise while not classifying small objects as ground, has been empirically found to be about 20 cm above the estimated ground plane.

This is probably the simplest approach and has an obvious negative consequence. In sharp transitions from flat ground to hills or on the verge of steep declines, which can for example be found on small bridges or large speed bumps, the ground measurements can exceed the height limit and thus result in false or missed detections. Also, tilting of the truck, as a result of hard breaking for example, could produce similar issues. The longitudinal range of the camera will affect how tolerant the system is to the road deviating from flat ground.

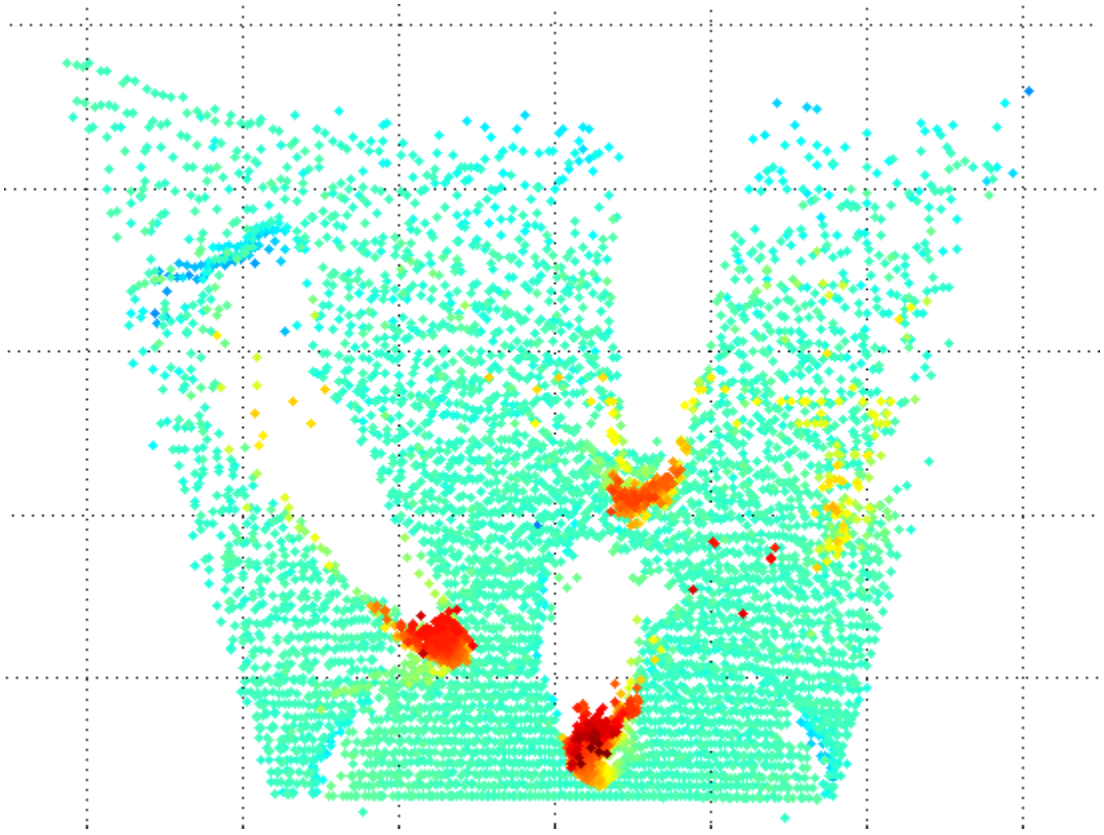


Figure 4.5: The same image as the one on the front page. It illustrates well the data with its "occupied" and "unknown" (shadowed from the TOFS point of view) areas.

An alternative approach to finding ground pixels could be to examine the gradient of the measured height and applying thresholds to the gradient. A low (absolute) value of the gradient would indicate a smooth plane surface, and if at the same time the height is within reasonable limits (taking into account steep hills etc.) the conclusion can be drawn that it is a ground pixel. The latter approach is however computationally more expensive.

#### Object pixels

Pixels that are not classified as ground but are confident are regarded as object pixels. As with the confident pixels, an erode operation is performed on the object pixels, this time with an eight neighbour.

### 4.3.4 Occupancy and unknown grid

As mentioned, the easy, fast, and robust, we argue, way we have chosen to process the adjusted 3D coordinates is to associate them to the cells of ground-parallel grids referred to as an *occupancy grid* and *unknown grid*.

The occupancy grid, the principal grid, denoted $\Omega_{i,j}$, is basically a thresholded two-dimensional histogram of the 3D space points, that originate from an object pixel, projected down on the $xy$-plane. This means that each cell is associated with the number of projected coordinates within each cell, and where the count is in turn translated to a binary value: 1 for a count above $\theta_\Omega$, 0 otherwise. Since the count of points falling inside a cell is roughly proportional to the cell's area and roughly inversely proportional to its horizontal distance from the camera $\Delta_{cc}$, which is hinted at in the image seen in Figure 4.5, the threshold is defined as: $\theta_\Omega = \frac{c_\Omega w_{cell}^2}{\Delta_{cc}}$.

The other grid expresses where there is lack of measurement data and is thus referred to as the unknown

grid, denoted $\Upsilon_{i,j}$, formed in a similar way but based on the ground AND object pixels since they are both considered confident measurements. Similarly, it is subjected to the threshold $\theta_\Upsilon = \frac{c_\Upsilon w_{cell}^2}{\Delta_{cc}}$ but inverted so that 1 represents lacking information. Again, the picture on the front page shows this phenomena; the unknown areas are the white ones without measurements.

# Chapter 5

# Tracking Algorithms

To get better estimates of the states of objects captured and represented by noisy measurements, tracking algorithms can be employed. They incorporate historical information about the state of the system to filter the measurements over time, yielding estimates of the states being measured, e.g. number of targets, target positions etc.. This allows for a more robust prediction of, for example, a targets future position.

## 5.1 Algorithms based on Kalman filtering

We have explored two types of Kalman filtering approaches which are from now on referred to as *box-based* and *cell-based*. The fundamental difference between them is their target size dimensionality; in the box-based each target is a box with a length, width and height, while the cell-based merely tracks the height and number of occupancy cells a target occupies. This difference in turn determines further algorithmic differences. The algorithmic structures of the two variants are the same and it is therefore natural to describe them in parallel.

### Overview of the Tracking system

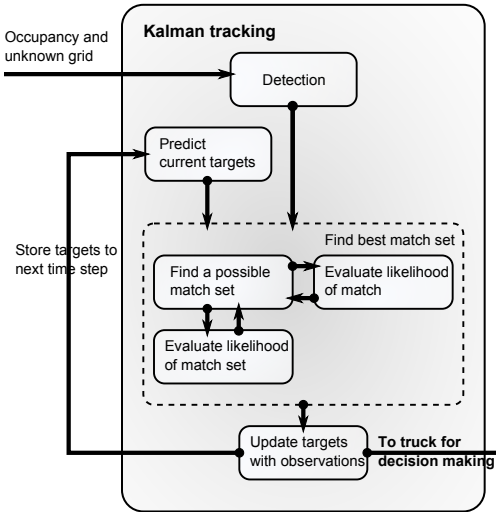



Figure 5.1: A schematic over the Kalman tracking process.

For each time step, the tracking process starts with identifying what we call *detections* from the occupancy grid given from the data processing. They are the basis for the tracking, each originating from of one or more objects which are to be tracked. The tracking algorithms are designed to try to distinguish these targets

and place them into the areas which these objects occupy.

As a second step, each one of the resulting targets from the previous time step, defined by their state vectors consisting of position, velocity and size, is then either associated with a detection from the current instance, deemed to exist in an unknown area or have disappeared from the FOV altogether. These hypotheses need to be evaluated for each target to find the best association, i.e. the one with the highest likelihood. By finding the set of associations with largest total likelihood, the detections can then be assigned to the tracks estimated in the previous iteration. The search for the best set of associations is done with the help of a so-called *branch-and-bound* algorithm to reduce the potentially huge search space. The Kalman filter is a central part of this process, as it calculates the probability distributions for the target states, and from this we estimate the probabilities which the likelihoods are approximated by.

### 5.1.1 Introduction to the Kalman filter

The Kalman filter is an algorithm that uses a time series of measurements to recursively estimate unknown variables. Given a sequence of noisy measurements, the filtered output is more exact than if only the most recent measurement is used. In fact, it produces a minimum mean squared error estimate of the state variables, describing the state of a system, for a linear dynamical system with additive Gaussian noise. Furthermore, it is the best - in the mean squared error sense - linear estimator for a linear system with any additive noise regardless of the distribution of the noise.

The algorithm has two steps for each recursion; prediction of state and an updating of the predicted state with new information. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties, using the result from the last recursion, but not the most recent observation. In the update step, the observation is taken into consideration. This is done by averaging the predicted state variables and observed state variables with weights that depend on the certainty of the measurements and the state variable prediction.

Due to the algorithm's Markov assumption and linearity, combined with the Gaussian noise assumption, the mean and error covariance of the current state is all the information that is needed to describe the state, i.e., are sufficient statistics. Furthermore, the filter requires only the the current observations and the previously calculated state variable probability densities to make optimal estimation; no information from further back in time is required.[10, p.199-203]

The Kalman filter has plenty of applications in technology e.g. in navigation, control and guidance for vehicles. Furthermore, it is widely applied in time series analysis used in fields such as signal processing and econometrics, to mention a few.

#### The process and observation models

For time step $k$, the state vector $\mathbf{x}_k \in \mathcal{R}^n$, which can consist of, for example, the position and velocity values of the object, is assumed to depend on $\mathbf{x}_{k-1}$ (this is the Markov assumption) with the following linear relation:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{w}_k \tag{5.1}$$

where $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ is the so called *process noise*, $\mathbf{F}$ is the $n \times n$ *state transition* matrix. The above equation is often referred to as the *process model*.

The observation, $\mathbf{z}_k \in \mathcal{R}^m$, is assumed to be generated according to the model:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{5.2}$$

where, $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$ is the observation noise, or measurement noise. [8, p. 2]. $\mathbf{H}$, an $m \times n$ matrix, is the *observation model*. The index $k$ on both $\mathbf{w}_k$ and $\mathbf{v}_k$ indicate that their magnitudes are allowed to vary between recursion steps.

#### The filter

Although derivations of the filter equations are skipped here, the curious reader can find them in, for instance, [7, p.433-436]. Instead we have determined that it will suffice to present the resulting equations, which are used for application.

In the prediction step the resulting variables are $\widehat{\mathbf{x}}_{k|k-1}$ and $\mathbf{P}_{k|k-1}$ which are the predictions of the state variables and error covariance matrix respectively, at time step $k$, given observations up to and including the previous time step:

$$\begin{aligned}\widehat{\mathbf{x}}_{k|k-1} &= \mathbf{F}\widehat{\mathbf{x}}_{k-1|k-1} \\ \mathbf{P}_{k|k-1} &= \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}_k\end{aligned}$$

The update step produces $\widehat{\mathbf{x}}_{k|k}$, which is the posteriori state estimate and $\mathbf{P}_{k|k}$, the posteriori error covariance matrix, also both at time step $k$, given observations up to and including the $k$th measurement:

$$\begin{aligned}\widehat{\mathbf{x}}_{k|k} &= \widehat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\big(\mathbf{z}_k - \mathbf{H}\widehat{\mathbf{x}}_{k|k-1}\big) \\ \mathbf{P}_{k|k} &= \big(\mathbf{I}_n - \mathbf{K}_k\mathbf{H}\big)\mathbf{P}_{k|k-1}\end{aligned}$$

which depends on the following relations:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1} \\ \mathbf{S}_k &= \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k.\end{aligned}$$

$\mathbf{K}_k \in [0,1]$ is known as the *Kalman gain* and is the term which weighs the influence of the observation against the prediction. The *innovation covariance* matrix, $\mathbf{S}_k$, is the estimated covariance of the measurement. If one studies the equations, one can see that, for example, for more uncertain observations (elements of $\mathbf{R}_k$ increases) and more certain prediction (elements of $\mathbf{P}_{k|k-1}$ decreases), the Kalman gain decreases and this in turn makes the posteriori approach the priori, as is expected.

### 5.1.2 Detection

As mentioned, both Kalman-based tracking algorithms are based on so called *detections*. A detection, denoted $D_i^k$, is a group of occupied cells in the occupancy grid which are connected (in a neighbourhood of 4 or 8 cells depending on the grid resolution. There are differences in how the detections are described for the two different tracking variants, but there are a couple of things they have in common.

Each detection $D_i^k$ consists of a number of cells with center coordinates $\{C_j^i \in R^2;\ j \in \{1,2,...,k\}\}$, resulting in a measurement:

$$\mathbf{z}_k^i = z_{fun}(D_i^k), \tag{5.3}$$

used as input to the tracking filters. Detections consisting of fewer cells than a certain threshold are considered false and are thus removed. The value of this parameter depends on the size of the cells and how small objects one would like to be able to detect. The height associated to a detection is set as the maximum $z$-value of the 3D coordinates within the detection.

Now to the particularities and differences between the detections of the two tracking variants. In the box-based approach the size of the detection is represented as a rotated box with a height above ground and *width-depth*-expansion in two orthogonal principal axes which are based on center coordinates of the cluster cells. A center point of the detection is calculated by forming a mean value of the center points $x, y$-values. Linear regression is used to estimate one principal axis from which the second then is found. On these, the cell center points are projected and the resulting maximum and minimum value in each direction from the detection center is regarded as the width and depth dimensions of the box. The width and its corresponding vector is always chosen to be the largest box dimension, and depth the smaller one. Since the points are in the center of the cells a detection with one-cell width or length translates to an object with zero-valued expansion in that dimension, and to compensate for this, a cell width is lastly added to each dimension. A graphic depiction of the above mentioned calculations can be found in Figure 5.2.

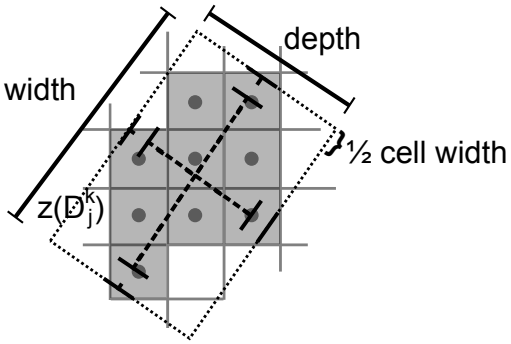



Figure 5.2: An illustration of the box-based measurement function $z_{fun}(.)$ measures the width and depth of a detection.

Detections for the cell-based tracking only include the height, calculated as for the box-based approach, and the number of cells constituting the object.

### 5.1.3 Target states

Each tracked target is described by two vectors where one contains its position and velocity, and the other its size. These are called *state vectors*, and mirror the way the detections are formed for each of the two tracking variants. They are explicitly defined as:

$$\mathbf{x}_{pos} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

and

$$\mathbf{x}_{size} = \begin{bmatrix} w \\ d \\ h \end{bmatrix}$$

for the box-based detections, where $x, y$ are the lateral and longitudinal coordinates of the center of a target, and $\dot{x}, \dot{y}$ are the velocities in these dimensions. $w$ is the width of the object, $d$ its depth and $h$ its height. In an attempt to increase robustness, the cell-based detection method reduces the width and depth to a single variable, representing the number of cells:

$$\mathbf{x}_{size} = \begin{bmatrix} n_c \\ h \end{bmatrix}$$

Note that the measured number of cells always is a discrete number, while the Kalman filter give a non-discrete value. This is not of importance; as described later, the state vector is really only used by the multivariate normal distribution function, (5.5), which obviously accepts non-discrete values.

### 5.1.4 Kalman filtering matrices

As mentioned in 3, the motion process chosen for this application assumes that the target acceleration between samples is described by a normally distributed random process, also known as a *constant velocity model*[10, p. 10-11], while the process model of the box-size changes the box size directly with noise. The matrices for the Kalman filter are thus:

$$\mathbf{F}_{pos} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now to the noise, which is modelled:

$$\mathbf{w}_k = \mathbf{B}\mathbf{u}_{k-1}$$

where $\mathbf{u}_{k-1} \in \mathbb{R}^2 \sim \mathcal{N}(0, \mathbf{Q}_\sigma)$, which is an isotropic (invariant to rotation) distribution because of its diagonal covariance matrix:

$$\mathbf{Q}_\sigma = \begin{bmatrix} \sigma_{pos}^2 & 0 \\ 0 & \sigma_{pos}^2 \end{bmatrix}$$

Because of the discrete time steps, the noise affecting the velocity ought to be proportional to $\Delta t$. The change of velocity also affects the position between the time steps, which is the integrated velocity change with respect to time in the interval (i.e. resulting in $\frac{\Delta t^2}{2}$). These aspects are expressed in the matrix $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$$

This results in the final noise covariance:

$$\begin{aligned} \mathbf{Q}_{pos} &= \mathrm{cov}(\mathbf{w}_k) \\ &= \mathbf{B}\mathrm{cov}(\mathbf{u}_{k-1})\mathbf{B}^T \\ &= \mathbf{B}\mathbf{Q}_\sigma\mathbf{B}^T \\ &= \sigma_{pos}^2 \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \end{aligned}$$

The one-to-one relationship between the measurements and the process variables and the lack of velocity information in the measurements give the observation model

$$\mathbf{H}_{pos} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

and measurement noise

$$\mathbf{R}_{pos} = \mathrm{diag}\begin{bmatrix} e_x & e_y \end{bmatrix}$$

In a similar notation for the size process models:

$$\mathbf{F}_{size} = \mathbf{I}_3$$

$$\mathbf{Q}_{size} = \mathrm{diag}\begin{bmatrix} \sigma_w^2 & \sigma_d^2 & \sigma_h^2 \end{bmatrix}$$

$$\mathbf{H}_{size} = \mathbf{I}_3$$

$$\mathbf{R}_{size} = \mathrm{diag}\begin{bmatrix} e_w & e_d & e_h \end{bmatrix}$$

or

$$\mathbf{F}_{size} = \mathbf{I}_2$$

$$\mathbf{Q}_{size} = \mathrm{diag}\begin{bmatrix} \sigma_{n_c}^2 & \sigma_h^2 \end{bmatrix}$$

$$\mathbf{H}_{size} = \mathbf{I}_2$$

$$\mathbf{R}_{size} = \mathrm{diag}\begin{bmatrix} e_{n_c}^2 & e_h^2 \end{bmatrix}$$

for the box-based detection and the cell-based respectively.

### 5.1.5 Target-detection association

For each time step $k$ the set of tracks $\{T_i^{k-1} : i = 1, 2, ..., t\}$, i.e. the targets from the previous time step, and $d$ detections $\{D_i^k : i = 1, 2, ..., d\}$ need to be associated to each other in some arrangement. Specifically, there are three hypotheses for each target which need to be explored. Either of these alternatives, $\{A_j : j = 1, 2, 3\}$, can be true for a target. A target can either be:

1. determined to be non-existent or lost outside the sensor area, from now on denoted $E_0$,
2. associated with a detection (partially or fully), denoted $D_i^k$, or
3. totally enclosed in an unknown area, found in the unknown grid, from now on denoted $E_1$, such that it cannot be observed.

Each of these associations, $M_i^k = \{T_i^k, A_j\}$, has a likelihood $\ell(M_i^k \mid D_{1:d}^k, T_{1:t}^k, M^{\chi}{}_i^k)$, given the other associations made for the current time step

$$M^{\chi}{}_i^k = \{M_j^k : j \in \{1, ..., t\} \setminus i\}.$$

How these likelihoods are derived is explained in 5.1.5.2. The goal of the association algorithm is to find the set of associations,

$$M_{1:t}^k = \{M_j^k : j \in \{1, ..., t\}\},$$

that maximizes the combined likelihood for all the matchings. As described in 5.1.5.2, the likelihoods are approximated with probabilities. One important characteristic of this approximation is that, given an association set, the likelihood of two associations with different detections and targets (this is also true for $E_0$, $E_1$) are independent. That is

$$\ell(M_a^k \mid D_{1:d}^k, T_{1:t}^k, M^{\chi}{}_a^k) \perp\!\!\!\perp \ell(M_o^k \mid D_{1:d}^k, T_{1:t}^k, M^{\chi}{}_o^k)$$

where $M_a^k = \{T_a^k, D_b^k\}$ , $M_o^k = \{T_o^k, D_p^k\}$ and where $a \neq o, b \neq p$. Therefore one could describe the total likelihood as a multiplication of the likelihood for a set of targets being associated for each detection. This is based on the fact that for the two independent events $A$ and $B$ the probability that both occur is $P(A \text{ and } B) = P(A)P(B)$. However, the targets that *share* a detection are not independent. Although, given the maximum-likelihood algorithm for sharing a detection, which will be described later, a reasonable approximation is to calculate their combined likelihood, again, as a multiplication. There is also the possibility for detections to have no associations which are thus considered to be new targets. Since there is one target created for each new detection, these likelihoods are also independent. Thus the final likelihood becomes:

$$\ell(M_{1:t}^k) = (\ell_{new})^s \prod_{i=1}^{t} \ell(M_i^k \mid D_{1:d}^k, T_{1:t}^k, M^{\chi}{}_i^k) \tag{5.4}$$

where $s$ is the number of detections not having targets associated with them; or simply, the number of new targets. The constant $\ell_{new}$ is the likelihood that is chosen for a new target.

Since there are $d$ detections, the three possible alternative matchings are $d+2$ in quantity. As we assume that any number of targets can share a detection, it means that the total number of possible matchings is $(d + 2)^t$, and checking them all can easily become infeasible. To reduce this number we use a so called branch-and-bound algorithm which only pursues a branch, of the combinatorial search tree, that has the possibility of producing a higher total likelihood than all others previously found.

#### 5.1.5.1 Branch-and-bound algorithm

If one represents all of the association solutions as a tree where each node at depth $i$ has $d + 2$ children nodes representing the possible matchings for target $T_{i+1}^k$, the best association set becomes the branch with the largest combined likelihood (equation (5.4)). Naturally, the solution can be found by simply searching through the whole tree, but the problem with this is that all $(d + 2)^t$ solutions would have to be explored. Figure 5.3 illustrates this tree with an example of an association set.

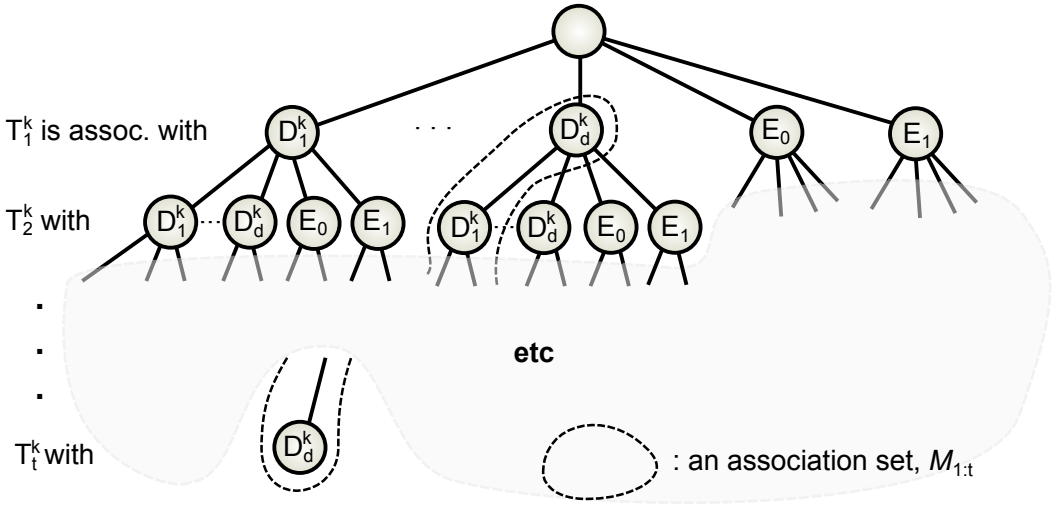



Figure 5.3: This figure depicts the tree which the branch-and-bound algorithm searches through. Each level $i$ consists of the detections that the $i$th target can be associated with. Note that many target can be associated with a detection, and thus each non-leaf node has the same number of children. An association set thus becomes a path of the tree, from the root to a leaf-node.

To eliminate the need to search the full depth of every branch, at each node we evaluate whether any of the branches in the sub-tree below has the possibility of producing a likelihood higher than the highest yet found. The evaluation is simple: each association likelihood lies in $[0, 1]$ (as it is either derived from an estimated probability, or chosen to be so for $E_0$ and $E_1$), which is the predefined range used herein, and the fact that the combined likelihood can be expressed as a product of likelihoods it follows that the combined likelihood decreases towards zero for each association beginning from 1 at the root. The likelihood approximation takes only values between 0 and 1, which follows from the approximation expressed in equation 5.7. Therefore, for each node it suffices to check whether the current likelihood is already below the best yet found and in that case skip the current branch.

**Making approximated likelihoods which are independent from each other**

In the case of multiple targets share a detection, all of the targets need to be associated to the detection before their combined likelihood can be calculated. This poses a problem for the branch-and-bound algorithm since every proceeding target in the combinatorial tree branch can potentially be a part of every detection. In other words, every target needs be associated before the likelihoods can be calculated, and thus a whole branch needs to be traversed. To avoid this, for every node in the combinatorial tree we make a "best case" likelihood approximation for each association which is guaranteed to have a larger likelihood than any actual association combination (i.e. $\ell(\text{best-case-}M_i^k) < \ell(M_i^k \mid D_{1:d}^k, T_{1:t}^k, M^{\chi}{}_i^k)$ ), while also being calculated independently from the other associations ($M^{\chi}{}_i^k$). This ensures that no branches of the combinatorial tree are skipped erroneously. How these "best-case" likelihoods are calculated is described in *Some targets yet unknown: Box-based* and *Some targets yet unknown: Cell-based*, respectively, in section 5.1.6.2 and 5.1.6.3. The actual likelihoods ($\ell(M_i^k \mid D_{1:d}^k, T_{1:t}^k, M^{\chi}{}_i^k$, for $i = 1, ..., t$) are then only calculated when all targets are associated - that is, when a leaf node is reached in the combinatorial tree. It is the actual likelihoods which are compared and are the basis of the final association set choice.

**Complexity**

Potentially, this branch-and-bound approach results in a significantly shorter search: in the best case only the nodes at the top layer and the first branch are traversed, giving the algorithm a best-case complexity of $\Omega(c(d + 2 + t))$ (assuming $O(c)$ for each node). In the worst case, however, if each consecutive branch has

larger likelihood than the previous and the node likelihoods are arranged in an unfortunately, the whole tree will be searched - which means a complexity of $O(ct(d+2)^t)$.

#### 5.1.5.2 The associative likelihoods

To determine how likely a certain association (between a target and a detection) is, the current state, at that particular instance, of the target is required. Naturally, this is not available at the time of detection. Therefore, the state is instead estimated by the prediction of the Kalman filter, and from the estimate, $\widehat{T}_i^k$, the association likelihoods can be calculated.

Since the state variables $\mathbf{x_k}$ and their covariance matrices $\mathbf{P}_k$ (for position and size) define the distributions of the target state, the equation used to evaluate the probability of a measurement is a multivariate normal distribution integrated around a point, combining the position and size state variables:

$$p(\mathbf{x} \mid \overline{\mathbf{x}}, \mathbf{P}) = \frac{1}{\sqrt{2\pi \left|\mathbf{P}\right|^n}} e^{-\frac{1}{2}(\mathbf{x}-\overline{\mathbf{x}})^T \mathbf{P}^{-1}(\mathbf{x}-\overline{\mathbf{x}})} \tag{5.5}$$

$$P(\mathbf{x} \in \Omega_{\overline{\mathbf{x}}} \mid \mathbf{P}) = \int_{\mathbf{y} \in \Omega_{\overline{\mathbf{x}}}} p(\mathbf{y} \mid \overline{\mathbf{x}}, \mathbf{P}) d\mathbf{y}, \text{ where } \Omega_{\overline{\mathbf{x}}} = \{\mathbf{y} : |\mathbf{y} - \overline{\mathbf{x}}| < \frac{1}{2}\epsilon\} \tag{5.6}$$

where $n$ is the number of elements in $\mathbf{x}$ (and $\overline{\mathbf{x}}$). $\epsilon$ is the size of the interval from which the probability is based. As long as $\epsilon \ll |P|$ holds, the exact value is not of importance; it only scales the relative likelihood of detection-target association compared to the other two alternatives, i.e. $E_0$ and $E_1$, which described in 5.1.5. We have chosen $\epsilon = 1$ mm. The approximated likelihood is thus:

$$\ell(M_j^k \mid D_{1:d}^k, T_{1:t}^k) = P(z_{fun}(D_k^j) \in \Omega_{\mathbf{x}_{k|k-1}^i} \mid \mathbf{P}_{k|k-1}^i) \tag{5.7}$$

where $M_j^k = \{T_j^k, D_i^k\}$ and $\mathbf{x}_{k|k-1}^i = [\ (\mathbf{x}_{pos,k|k-1}^i)^T \quad (\mathbf{x}_{size,k|k-1}^i)^T\ ]^T$ and $\mathbf{P}_{k|k-1}^i = \begin{bmatrix} \mathbf{P}_{pos,k|k-1}^i & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{size,k|k-1}^i \end{bmatrix}$, and $z_{fun}(.)$ is the measurement function introduced in 5.1.2.

$\ell(M_j^k \mid D_{1:d}^k, T_{1:t}^k)$ is the result of an integration of a normal probability density function , and therefore it falls within $[0, 1]$ (although, due to the small $\epsilon$ and high dimensionality of the integrated space it is significantly closer to 0 than 1). This fact makes the branch-and-bound possible.

One omission that the likelihood approximation makes is that the measurement noise of the current time step ($\mathbf{R}_k$ in the Kalman equations) is ignored. The measurement noise for the previous time step is however included in the covariance matrix $P_{k|k-1}^i$, as was described in 5.1.1.

#### Target is non-existent or outside field of view ($M_i^k = \{T_j^k, E_0\}$)

Beside some minor differences concerning parameter values, this association case is the same for both tracking variants.

Even when using the accurate time-of-flight sensor and removing small detections, the possibility for a false detections remains. This is often due to the wrapping problem. It is rarely the case that a false detection exists for a prolonged time; most commonly, it only appears for one or two frames. Therefore we have chosen to let the likelihood for a target to be associated with $E_0$ decrease to some extent with the lifetime of a track. There is also the possibility that an object has moved outside the FOV. The likelihood of two cases are combined into:

$$\ell(M_i^k \mid T_{1:t}^k, D_{1:d}^k) = \max(1, \tau^{max} + 1 - \tau_j) \frac{\ell_{E_0}^{max}}{\tau^{max}} \tag{5.8}$$

where $M_i^k = \{T_i^k, E_0\}$ and $\tau_j$ is the time the target $j$ has existed. $\ell_{E_0}^{max}$ depends on the distance to the sensor area borders, making it more likely for it to disappear when it is close to the border:

$$\ell_{e_o}^{max} = \begin{cases} \ell_{E_0}^{border} & \text{, if } dist(T_k^i, border) <\approx 40cm \\ \ell_{E_0}^{inside} & \text{, otherwise} \end{cases} \tag{5.9}$$

Where $\ell_{E_0}^{border} > \ell_{E_0}^{inside}$ and are determined empirically. Although not exhaustively evaluated, the parameter values in the order of $\ell_{E_0}^{inside} = e^{-55}$, and $\ell_{E_0}^{border} = e^{-40}$, for the box-based tracking seem to give relatively good results.

**Target is matched with a detection ($M_i^k = \{T_j^k, D_l^k\}$)**

There are three ways a target can be associated with a detection; it can be the sole target, it can be among multiple where all others are known, or it could be among many where not all are known due to an incomplete association set. The association set is only complete when a leaf node is reached in the combinatorial tree (Figure 5.3), mentioned in 5.1.5.1. For the first two cases, the likelihood of the associations can be calculated, since all targets, with their state vectors, are known. But when there are targets with undefined detection associations, a likelihood estimation must be made. This needs to be done with some care because if the estimate is too low, the total likelihood of the association set can become so low that the branch of the tree is skipped, thus missing a possible solution erroneously. Conversely, a too high estimation results in too many sub-trees pursued and thus slowing the algorithm. In other words, an as high as possible best-case likelihood estimation for a multiple target-detection association (which is not too demanding computing-wise) is wanted. Our solutions, for both tracking variants, to these three cases are described in the three following sections.

**Target is matched with a detection: one to one**
For both tracking variants, when a single target is proposed to be matched with a detection, the complete area of the detection is assumed to be a part of the target. It is possible that part of the object is (or has been in the previous time step) occluded, making the object seem smaller (or larger) and thus lowering the likelihood of the association. Figure 5.6 shows such an example. Therefore an appropriate unknown area, if available, is included to the detection. How this is done is explained in 5.1.6.1.

Given the detection, which is expanded into the unknown area, the likelihood becomes the probability similarly as previously, expressed in equation (5.7).

**Target is matched with a detection: Multiple targets sharing a detection**
The occupancy grid approach allows for multiple objects to seem as one by moving together, and thus multiple targets need to be able to share a detection. As mentioned earlier, this poses a problem for the branch-and-bound search since it needs to calculate $\ell(M_i \mid M^{\chi_i^k}, D_{1:d}^k, T_{1:t}^k)$ for each node $i$ but $M^{\chi_i^k}$ is only determined when all other targets are matched up, which is only when node $i$ is a leaf node. This issue can be dealt with by estimating a best case association where the decrease in total likelihood is certain to be smaller than any $M^{\chi_i^k}$ compatible with the current branch. Our approach is based on a couple of assumptions, making it simple and fast. The specific procedure of forming measurements with these assumptions is found in 5.1.6.2.

When a leaf node is reached, $M_j^k$, and thus also $M^{\chi_j^k}$, are known for all $T_j^k$. Therefore, better likelihood estimations can be calculated. This entails breaking the detections that are shared by multiple targets into an equal number of pieces, each piece then being matched with one of the targets. How the splitting is performed is also described in 5.1.6.2. Since each detection piece then is matched with its corresponding target the same likelihood calculation as in 5.1.6.1, can be made for each association.

**Target is completely in an unknown area ($M_i^k = \{T_j^k, E_1\}$)**

There is also the possibility that the target is entirely occluded and therefore exists in an unknown area, a state denoted $E_1$. The two tracking variants handle this case differently and are described separately below. For both methods, when an unknown area is occupied, a detection and its corresponding state measurements are formed with the standard procedure, found in 5.1.2, from which in turn a association-likelihood is calculated, also in the manner already mentioned.

**Occupying unknown cells: Box-based**
Relying heavily on the Kalman filter and its prediction, we simply assume that the unknown cells with

center coordinates inside the predicted target box forms the basis of the new detection. As mentioned, the unknown cells are the 1-valued cells in the unknown grid, see section 4.3.4. For a graphic depiction, see Figure 5.4.

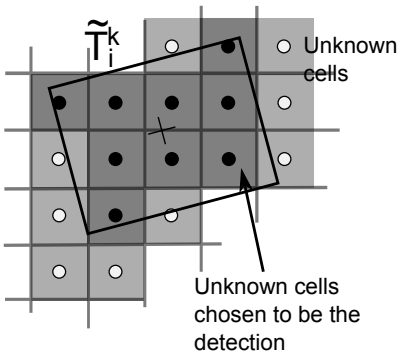



Figure 5.4: A sketch of an example scenario where predicted box-based target occupies unknown cells only.

**Occupying unknown cells: Cell-based**
Compared to the box-based variant, the cell-based occupation of unknown cells is somewhat more complicated. It is an iterative process beginning with occupying the closest unknown cell $C_{min}$ to the predicted target position $\widetilde{T}_i^k$. The unknown area subsequently expands repeatedly by including the adjacent unknown cell closest to $\widetilde{T}_i^k$ - one for each iteration. The occupied cells are evaluated each iteration by calculating their likelihood, using equation 5.7, and the recurrence is ended when a new largest likelihood, among those in the recurrence, has not been reached in $\kappa$ iterations. Figure 5.5 illustrates an example occupation with the first two iterations. Note that this iterative process is performed to completion for every time step, $k$.

Determining a good value of $\kappa$ is associated with some considerations; for example, increasing $\kappa$ increases the chance of larger association likelihoods but on the other hand also increases the number of iterations, thus making the algorithm slower. Also, $\kappa$ is coupled with the cell size ($w_{cell}$); the smaller cell size the larger $\kappa$ ought to be if one wants the algorithm to cover a similar area. $\kappa$ is thus preferably proportional to the cell size squared.

### 5.1.6 How the detections are formed for the different cases

As recounted in the previous section, for every detection there are two types of hypotheses; it is either the result of a single object or a collection of objects that are close to each other. Here we explain how the detections are divided between targets depending on which targets they are associated with.

#### 5.1.6.1 Single target occupying a detection

When a target alone occupies a detection, the process is, for both tracking variants, significantly simpler than in the multiple-target case. It is assumed that the whole detection is part of the target and we let the Kalman filter consider the measurement uncertainties. For this case, it is possible for previously visible parts of an object to become occluded and make it seem smaller than before, which in turn lowers the target-detection association likelihood. A scenario of this sort is depicted in Figure 5.6. In an attempt to reduce this risk, some of the unknown cells which are most likely to be occupied by the targeted object are included. The difference between the box-based and cell-based tracking algorithms in how a single target occupies a detection is how these unknown cells are chosen to be included with the detection.

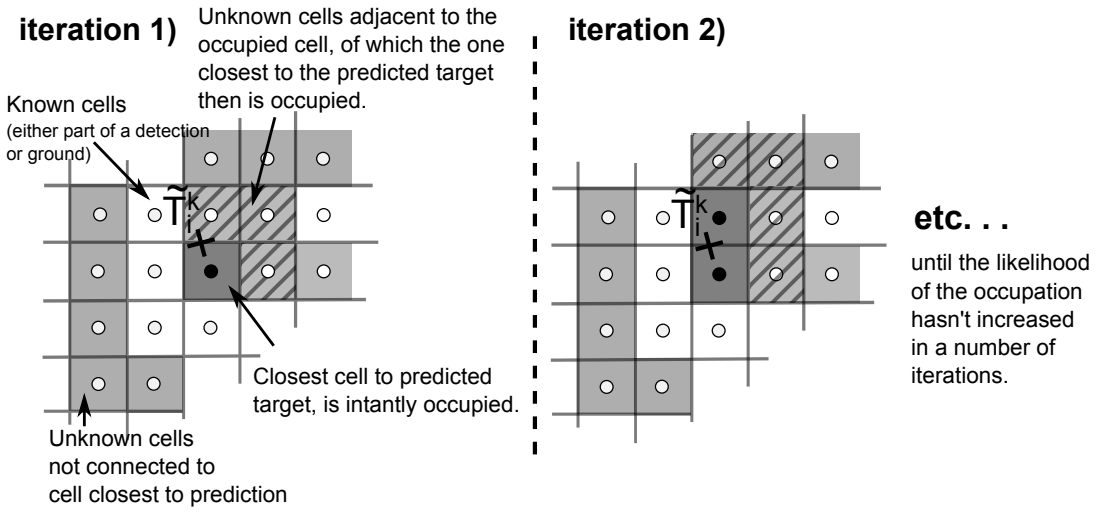



Figure 5.5: Two iterations of the cell-based algorithm occupying only unknown cells.

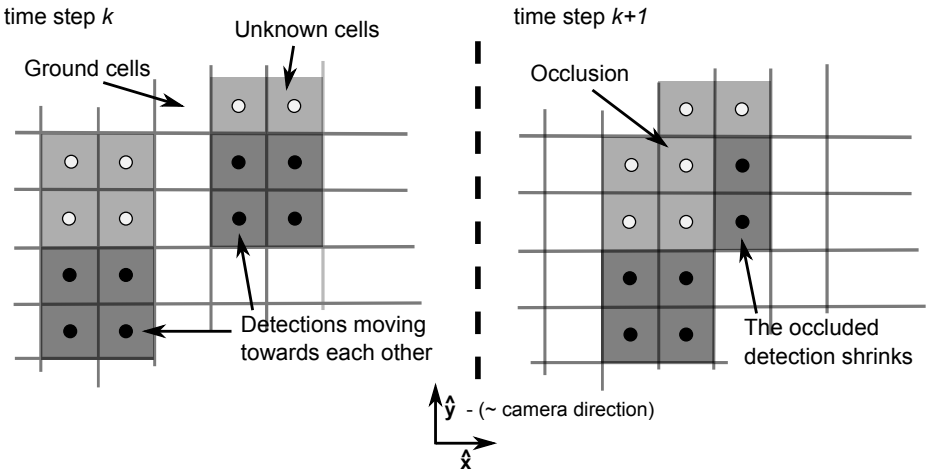



Figure 5.6: An example scenario where an object occludes another.

**Single target occupying a detection: Box-based**

As mentioned, all the cells of the concerned detection are first assumed to be part of the target. The unknown cells included are chosen by a comparison with the predicted targets bounding box; the cells with center coordinates within the bounding box of $\widetilde{T}_k^i$ are included. Figure 5.7 depicts this procedure. When included, the unknown cells are treated analogously as a detection cell, and therefore affect the size and position of the detection similarly. This is described in 5.1.2.

**Cell-based tracking - a single target occupying a detection**
While the occupation of the detection cells here remains equal to the box-based, the possible occupation of a surrounding unknown area is handled similarly as the procedure of which this tracking variant occupies cells in an unknown area, see 5.1.5.2. The only difference is that the initial cells occupied are the detection cells, and thus it is the unknown cells adjacent to these which possibly also are occupied by the target.

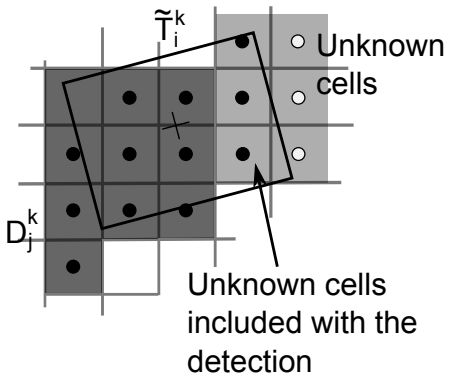



Figure 5.7: Sketch of how unknown cells are included to a detection using the box-based tracking.

#### 5.1.6.2 Multiple targets occupying a detection: Box-based

For both tracking variants, the fundamental assumption here is that all of the cells of the detection are a part of a target. Also, to not make the target-detection association overlap with the unknown area-association alternative, each target has to occupy at least one detection cell. Having an estimated probability distribution for both position and size of a target, a likelihood for each target occupying a cell can be formed. From these likelihoods a maximum-likelihood partition of the detection cells can be made; each cell of the detection is assigned to be a part of the target which has the largest likelihood of occupying it. These points hold true for both the box-based and cell-based, but their differences follow in the two sections below.

**Some targets yet unknown: Box-based**
On the question of how to evaluate a particular association when there are more associations still to be made in the association set, we have a solution that is simple and fast. Remember, the aim is to calculate a "best-case" association likelihood for the target and detection. First off, the likelihood increases significantly by individually treating each target which potentially shares the detection, thus ignoring what cells the other targets are considered to occupy. To increase the likelihood further, the occupation is assumed to give the same box size as the predicted target has, as this is the size which gives the largest likelihood. Remember that the likelihood increases with positions closer to the prediction and that the target still needs to occupy at least one cell. Adhering to the above mentioned, it follows that the placement of each target with the largest likelihood is where it occupies at least one cell of the detection and has its center on the line between the target $\widetilde{T}_i^k$ and the closest cell coordinate to it (see Figure 5.8). Since the process disregards the rotation of targets, the target is also assumed to be rotated so as to maximize the likelihood (i.e. its center as close to the center of $\widetilde{T}_i^k$ as possible).

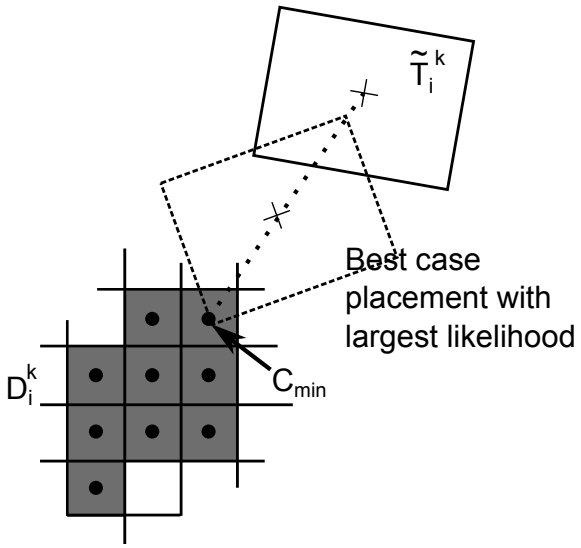



Figure 5.8: A sketch of the largest probability placement of the target while sharing a cell of the detection.

Mathematically, for each target, the best case observation has the state variables:

$$\mathbf{z}_{pos}^{bc} = \left(\frac{C_{min} - \mathbf{x}_{pos}}{\|\mathbf{z}_{pos} - \mathbf{x}_{pos}\|_2}\right)\left(\frac{\|\mathbf{x}_{pos}\|_2}{2}\right) + \mathbf{x}_{pos}$$
$$\mathbf{z}_{size}^{bc} = \mathbf{x}_{size}$$

where $C_{min}$ is the closest cell coordinate of the detection, $\mathbf{x}_{pos}$ is the position of the target, $\mathbf{x}_{size}$ is the target size in three dimensions, and, of course, $\mathbf{z}^{bc}$ are the corresponding variables for the best case observation.

**All other targets are known: Box-based**
When all targets assigned to share a detection are known, which happens when the branch-and-bound algorithm has reached a leaf-node, the cells of the detection are split up between the targets. An example of this split can be seen in Figure 5.9. Each detection cell is associated with a likelihood of occupation for every target. This likelihood is derived from the target state vectors by transforming them to a $x, y$ position probability distribution for the target box corners, which in turn forms the probability of a point being inside the box. The specifics of this process will now be described.

To simplify the calculations, the first step is to make the coordinate system relative to the target box; moving the origin to the target box position and making the target principal axes the coordinate system axes. This is done by translating the origin and projecting it on the new orthonormal base vectors $\widehat{\mathbf{e}}_w$, $\widehat{\mathbf{e}}_d$ that are the principal axes of the target box. This means transforms a point $\mathbf{p}$ in the original coordinate system to $\mathbf{p}^*$, in the reoriented system:

$$\mathbf{R} = [\ \widehat{\mathbf{e}}_w \quad \widehat{\mathbf{e}}_d\ ]$$
$$T^*(\mathbf{r}) = \mathbf{R}\big(\mathbf{r} - [\ x \quad y\ ]^T\big) \tag{5.10}$$
$$\mathbf{p}^* = T^*(\mathbf{p}) \tag{5.11}$$

where $x, y$ are the targets position coordinate values. See Figure 5.10, which illustrates these calculations.

The covariance matrix $\mathbf{P}_{pos}$ also needs to be adjusted to the new coordinate system. This is done by projecting the underlying random variable, $X_{x,y}$, onto the new axes. Since it is already translated, the

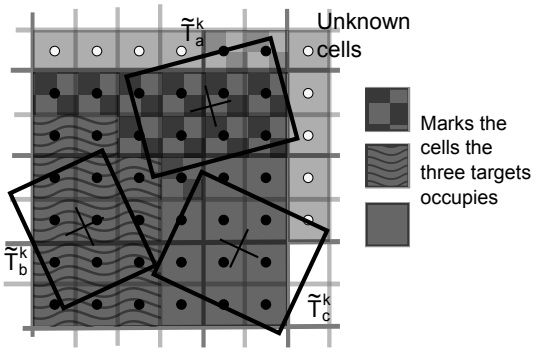



Figure 5.9: An example of a detection being split up between three targets.

projection is simply $\mathbf{R}X_{x,y}$, from which the new covariance matrix is derived:

$$\begin{aligned}
&\mathrm{cov}(\mathbf{A}B) = \mathrm{E}\big[(\mathbf{A}B - \mathrm{E}[\mathbf{A}B])(\mathbf{A}B - \mathrm{E}[\mathbf{A}B])^T\big] \\
&= \mathrm{E}[\mathbf{A}B(\mathbf{A}B)^T] - \mathrm{E}[\mathbf{A}B]\mathrm{E}[(\mathbf{A}B)]^T \\
&= \mathrm{E}[\mathbf{A}]\mathrm{E}[BB^T]\mathrm{E}[\mathbf{A}^T] - \mathrm{E}[\mathbf{A}]\mathrm{E}[B]\mathrm{E}[B^T]\mathrm{E}[\mathbf{A}^T] \\
&= \mathrm{E}[\mathbf{A}]\big(\mathrm{E}[BB^T] - \mathrm{E}[B]\mathrm{E}[B^T]\big)\mathrm{E}[\mathbf{A}^T] \\
&= \mathbf{A}\big(\mathrm{E}[BB^T] - \mathrm{E}[B]\mathrm{E}[B^T]\big)\mathbf{A}^T \\
&= \mathbf{A}\mathrm{cov}(B)\mathbf{A}^T \\
&\Rightarrow \big[\mathrm{cov}(X_{x,y}) = \mathbf{P}_{pos,1:2}\big] : \mathrm{cov}(\mathbf{R}X_{x,y}\mathbf{R}) = \mathbf{R}\mathbf{P}_{pos,1:2}\mathbf{R}^T
\end{aligned}$$

So, the rotated covariance matrix is thus defined as:

$$\mathbf{P}^*_{pos} = \mathbf{R}\mathbf{P}_{pos,1:2}\mathbf{R}^T \tag{5.12}$$

In this new coordinate system, the upper right corner of a target box, seen from above, has the coordinate position distribution given by the random variable $Z^{\urcorner}$:

$$Z^{\urcorner} = X_{x,y} + \frac{1}{2}X_{w,d}$$

or

$$\begin{aligned}
Z^{\urcorner} &= \mathbf{H}X \\
\mathbf{H} &= \begin{bmatrix} 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & \frac{1}{2} \end{bmatrix} \\
X &= \begin{bmatrix} X_{x,y} & X_{w,d} \end{bmatrix}^T
\end{aligned}$$

where $X_{x,y}$ is the target box $x, y$-position with mean in the origin:

$$X_{x,y} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}^*_{pos})$$

and $X_{w,d}$ is the target box width and depth that also has a covariance matrix (which is already appropriately oriented).

Combining their covariance matrices gives $\mathbf{P}^*$:

$$\mathbf{P}^* = \mathrm{diag}\begin{bmatrix} \mathbf{P}^*_{x,y} & \mathbf{P}_{w,d} \end{bmatrix}$$

which is the covariance matrix of $X$, but for $\mathbf{H}X$:

$$\mathbf{P}_{\mathbf{Z}^{\urcorner}} = \mathrm{cov}(\mathbf{H}X) = \mathbf{H}\mathrm{cov}(X)\mathbf{H}^T = \mathbf{H}\mathbf{P}^*\mathbf{H}^T$$

and, of course:

$$\mathrm{E}[Z^{\urcorner}] = \mathrm{E}[\mathbf{H}X] = \mathrm{E}[X_{x,y} + \frac{1}{2}X_{w,d}]$$
$$\mathrm{E}[X_{x,y}] + \mathrm{E}[\frac{1}{2}X_{w,d}] = \frac{1}{2}\mathrm{E}[X_{w,d}] = \frac{1}{2}[\ w \quad d\ ]^T$$

These equations give that the probability density of a corner having the position $\mathbf{p}$:

$$p(Z^{\urcorner}) = p(\mathbf{p}^* \mid [\ \tfrac{w}{2} \quad \tfrac{d}{2}\ ]^T, \mathbf{P}_{Z^{\urcorner}})$$

where $p(.)$ is the multivariate normal distribution, defined in equation (5.5). From this probability distribution, the probability distribution of points being to the left and below the corner (i.e. lower $x, y$ values) becomes the integral of the distribution from each point to $\infty$ :

$$P(p_1^* < Z_1^{\urcorner} \wedge p_2^* < Z_2^{\urcorner}) = \int_{p_1^*}^{\infty}\int_{p_2^*}^{\infty} p(\mathbf{x} \mid [\ \tfrac{w}{2} \quad \tfrac{d}{2}\ ]^T, \mathbf{P}_{Z^{\urcorner}})dx_1dx_2 \tag{5.13}$$

Since the target box is rectangular and all of the corner points probability distributions have the same covariance matrices, one can treat every $\mathbf{p}$ as if it was in the upper-right quadrant of the target. Also, because it is given that the left bottom and upper corners are to the left (lower $x$) of the target center, the other corners can be ignored. Once again, Figure 5.10 depicts this graphically, The probability of a point $\mathbf{p}$ being inside the target $T$ is thus derived from equation (5.13):

$$P(\mathbf{p} \in Box(T)) = P(|T^*(\mathbf{p})_1| < Z_1^{\urcorner} \wedge |T^*(\mathbf{p})_2| < Z_2^{\urcorner}) \tag{5.14}$$

where $Box(T)$ is the rectangular 2D space the target $T$ is assumed to take up. $T^*(\mathbf{p})_1$,$T^*(\mathbf{p})_2$ is the first and second element, of the translated point, respectively.

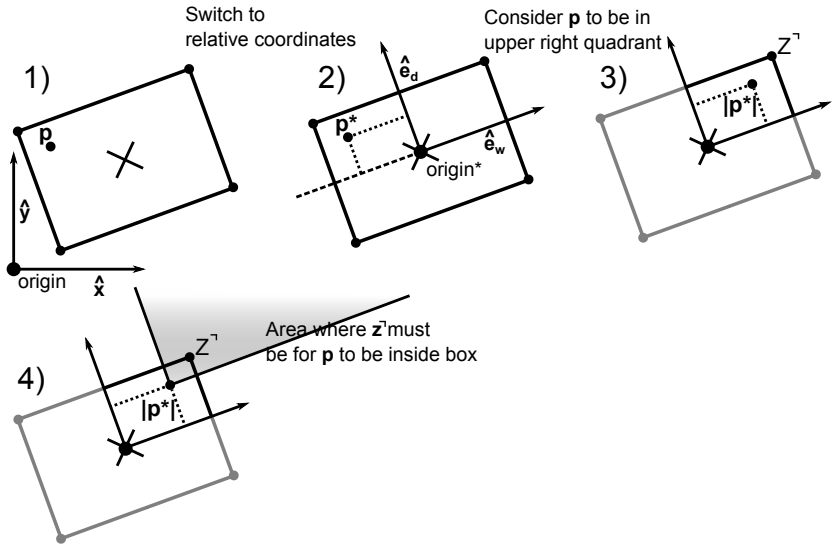



Figure 5.10: The four mathematical steps that precede how a $x, y$-point is calculated to be inside a target box.

Now the likelihood of a cell $C_j^i$ being occupied by target $T_l^k$ is simply equation (5.14) applied to the cell:

$$\ell_{occ}^l(C_j^i) = P(C_j^i \in Box(T_l^k)) \tag{5.15}$$

#### 5.1.6.3 Multiple targets occupying a detection: Cell-based

**Some targets are unknown: Cell-based**
As in the box-based variant, when not every target associated to a detection is known, a best case occupation with an as large as possible likelihood is needed for each target. To get these, the procedure treats each target independently; it is possible for many targets to occupy the same cells, and they do not depend on each other in any way. The procedure is essentially the same as the cell-based unknown area occupation (see 5.1.5.2), but with two exceptions: it begins by occupying the closest detection cell and subsequently occupies both unknown and detection cells.

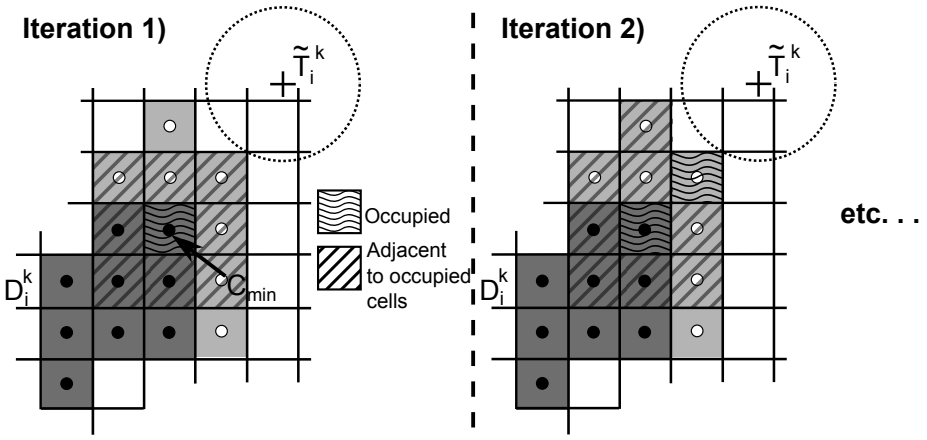



Figure 5.11: An example of an estimated best case occupation of a target.

**All other targets known: Cell-based**
When all targets associated to a detection are known, which happens only when the branch-and-bound algorithm has reached a leaf node, the occupation follows the same underlying structure as for the box-based. The detection is first split up and completely occupied by the targets, without allowing overlaps between them. Lastly, the surrounding unknown area is occupied by the target.

The detection cells are split between targets in a manner similar to the box-based; each cell is associated with a likelihood for each target and is assigned to the one with the greatest likelihood. These likelihoods are calculated in a procedure where the detection cells are temporarily occupied one by one in the order of their distance to the predicted target $\widetilde{T}_i^k$, and a target-detection association likelihood is then calculated for every iteration. This order of occupation implies that, for each target, every possible set of occupied cells will include the closest cell, and every set except one will include the second closest cell, and so forth. This means that the likelihood of occupation for each cell is the combined likelihoods of the sets of occupied cells that contains it. Figure 5.12 depicts the cell occupation likelihood calculation. The iteration is done until all detection cells have a likelihood for each target, and then the target with the largest likelihood for each cell gets to occupy it.

The occupied cells are then expanded by occupying unknown cells precisely as described in 5.1.6.1. Finally, the occupation likelihood is similarly calculated as when a detection occupies only unknown cells and was described in 5.1.5.2.

## 5.2 Tracking - Particle based

As a reference for the aforementioned tracking methods a particle tracking algorithm inspired by [5] was implemented.

The traditional particle filter, also known as sequential Monte Carlo method, has a few similarities to the Kalman filter. The key similarity is that it provides an estimate of the Bayesian model of the current targets

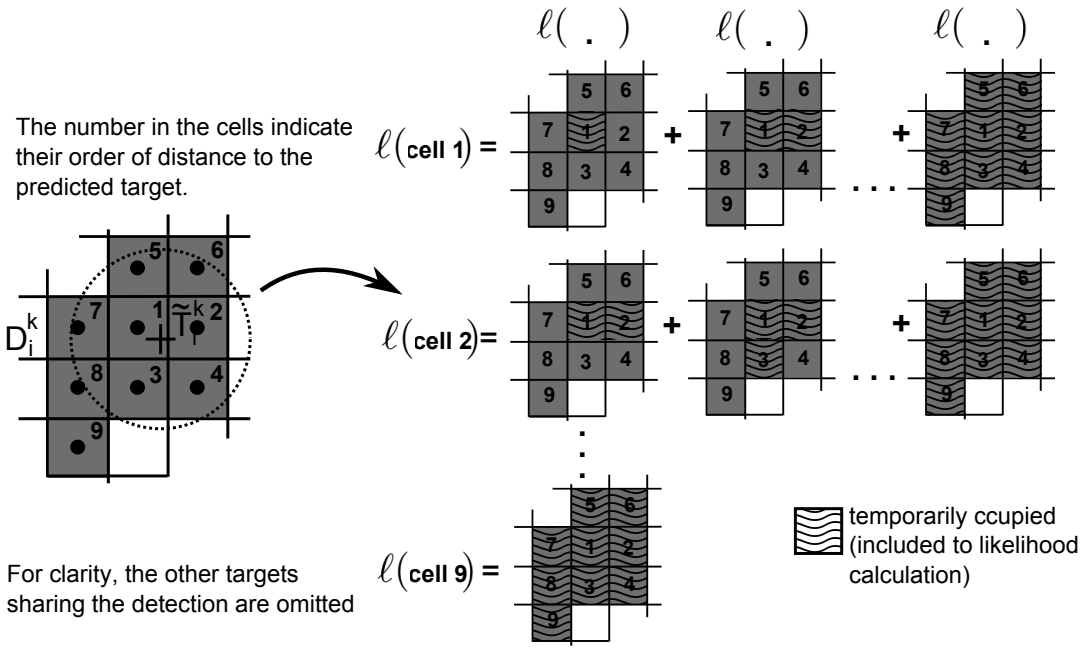



Figure 5.12: For a given target, this shows how the likelihood for each cell is calculated, when a detection is split up between many targets.

state based on its historical development. It sequentially simulates the model with a number of samples called particles, more particles usually meaning more accurate filtering, going towards optimal Bayesian estimates. Thus by choosing the number of particles to be sufficiently large the particle filter ought to give a good reference for comparison. In our case, each particles consists of an $x, y$-position, velocity and an age variable (more about this later).

Just like the Kalman filter the particle filter requires *a priori* information on the underlying models of the system: the process model, as found in equation 5.1 ($\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{w}_k$) and the observation model, equation 5.2 ($\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$). However, an important difference on this point is that the classical particle filter does not require the models to be linear to give good results. Furthermore, the Kalman filter supposes that the uncertainties, $\mathbf{w}_k$ and $\mathbf{v}_k$ are normally distributed and the resulting state estimate distributions will therefore also be Gaussian, whereas the particle filter has no such preferences - it can handle any noise distributions in the models and provide state estimates with arbitrary distributions.

The idea is to, from a distribution of the targets initial state, $\mathbf{x}_0$ (which is also needed beforehand) draw samples and let them evolve according to the process model, and subsequently draw new samples from that particle set to form the particle set for the next time step. Each particle is associated with a specific weight and the measurements are then used to update the weights. The purpose of the weights is to enable the calculation of the weighted average of the particles which will be the filtered estimate of the targets state, and they also define the probability that the particle survives to the next time step. This leads us to the introduction of the implemented particle tracking method, which does not use weights, described in the following sections.

## Overview of *Danescu et al* [5]

The particle filter we used is designed for the occupancy grid and can handle multiple targets with one set of particles. As already touched upon, it differs from the convention in some respects. Most importantly, the particles do not have varying weights, and each one poses a hypothesis for a cell's velocity and occupation. The resulting occupancy grid is formed from the particles only, and not from the measurements directly, which one might say deviates from the particle filter in its standard form, described in [6], for example.

For each time step, the age variable is increased by one for each particle. Only the particles which are above a certain age are active, and when the number of those particles in a cell exceeds a certain threshold (half of the maximum allowed number of particles per cell), the cell is considered occupied. Furthermore, a mean velocity of all particles for each occupied cell is used for clustering. The cells that have a velocity in both $x$ and $y$ dimension (regardless of direction) smaller than twice the cells standard deviation are considered to be static. This ensures that a newly seen object occupies cells before its particles have been able to conform into more similar velocities from the initial uniform random distribution. Those neighbouring cells with similar averaged velocity and directions are then labelled as part of the same target.

### Our additions

To keep the frame rate relatively unaffected by the amount of sensor data, we introduce a cap on the number of particles, denoted $n_{global}$. Using the same notation as in [5] the maximum number of particles in a cell is adjusted to the number of occupied cells, $|occupied\_cells|$:

$$n_c = min(n_c^{init}, \frac{n_{global}}{|occupied\_cells|})$$

## 5.2.1 Target state forming

Target states are formed from the labelled cells' coordinates and their contained particles. While the position and size are determined by the cells as before with, described in 5.1.2, the velocities are set as the mean of contained particles with a valid age. The resulting state variables are similar to the box-based ones:

$$\mathbf{z} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ w \\ d \\ h \end{bmatrix}$$

## 5.2.2 Target association

Since there are no detections which need to be split up between different targets, and the filter handles occlusion of objects by effectively letting particles survive longer in unknown areas compared to areas lacking measurements, the target association becomes much simpler than previously. This means that a local search suffices; finding the best association for each target individually will have a large probability to be the correct one. Of course, a target can appear or disappear from one step to the next. This calls for a restriction limit where improbable matchings are impossible and targets disappear and are now created instead.

The obvious choice of algorithm for this type of association is the nearest neighbour with a maximum distance for a association. The algorithm simply calculates the distance between each target and all of the previous targets using equation (5.16). These distances are then sorted in ascending order and each target are then paired up in that order until a distance which is larger than a constant, $\text{dist}_{max}$, is reached. Then the remaining unmatched targets are considered to be new, and the unmatched targets are considered to have disappeared. A benefit of the ordered pairing is that it makes the associations invariant to the order of the current and previous targets.

Compared to the exponential complexity of the branch-and-bound, the most complex part of the nearest neighbour algorithm, calculating all of the distances, is $m \cdot n$. $m$ and $n$ being the number of previous targets and targets respectively.

### Calculating the distances

The distance function simply is:

$$\text{dist}_w(\mathbf{z}_i^k, \mathbf{z}_j^k) = \|\text{diag}[\mathbf{w}](\mathbf{z}_i^k - \mathbf{z}_j^k)\| \tag{5.16}$$

where diag[$\mathbf{w}$] is a weight matrix which does three things: i) gives the velocity elements the same unit as the position elements, ii) downplays the significance of the $w, d$-size difference relative to the position and size variables, and iii) wholly disregards the height of the object:

$$\mathbf{w} = \begin{bmatrix} 1 & 1 & \Delta t & \Delta t & 0.5 & 0.5 & 0 \end{bmatrix}^T$$

Finally, the maximum distance is:

$$\text{dist}_{max} = \|\text{diag}[\mathbf{w}]\mathbf{d}_{max}\|$$

where $\mathbf{d}_{max} = \begin{bmatrix} 0.8 & 0.7 & 0.7 & 0.2 & 0.2 & 0 \end{bmatrix}^T$.

# Chapter 6

# Results

The evaluation results presented here are products of both the detection algorithm and the detection plus tracking applications as a whole. The data used in the evaluation has been gathered with the TOF. Unfortunately the stereo camera built did not provide data of sufficient quality to be used for the evaluation of the algorithms.

## 6.1 Evaluation

The data used for evaluation consists of three sets. One was recorded outdoors with the TOFS mounted on a truck, while the second was captured indoors with noise added to compensate for the minimum background IR-light. The third was recorded solely for the purpose of evaluating the tracking algorithms' abilities to separate detections into the correct number of targets, and consists of 1 person walking close to a larger obstacle. In effect, these sequences always contain two objects fully being in the FOV. This avoids the detection problems that are produces at the edges of the FOV. The sequences of this dataset are also recorded indoors with similar noise added as previously.

Table 6.1 and Figure 6.1 contains some of the characteristics of the three set-ups. Sample frames from all of them are also shown in Figure 6.2, 6.3 and 6.4.

| **Notation** | **Environment** | **Modification** | **Avg. number of objects per frame** | **Length** |
|---|---|---|---|---|
| INDOORS | Indoors, 4 pedestrians | Noise added | 2.9 | 93.3 s |
| TRUCK | Outdoors, mounted on truck, 3 ped. | None | 2.1 | 186.7 s |
| 2OBJ | Indoors, 1 ped. 1 obstacle | Noise added | 2 | 14.6 s |

Table 6.1: Characteristics of the two data sequences used in the evaluation

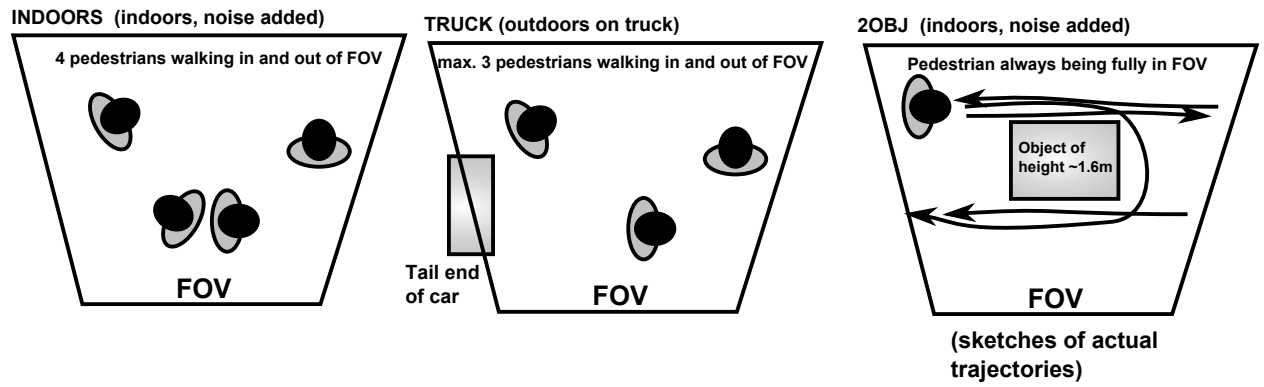



Figure 6.1: Sketches of the set-ups of the three data sequences used in the algorithm evaluations.

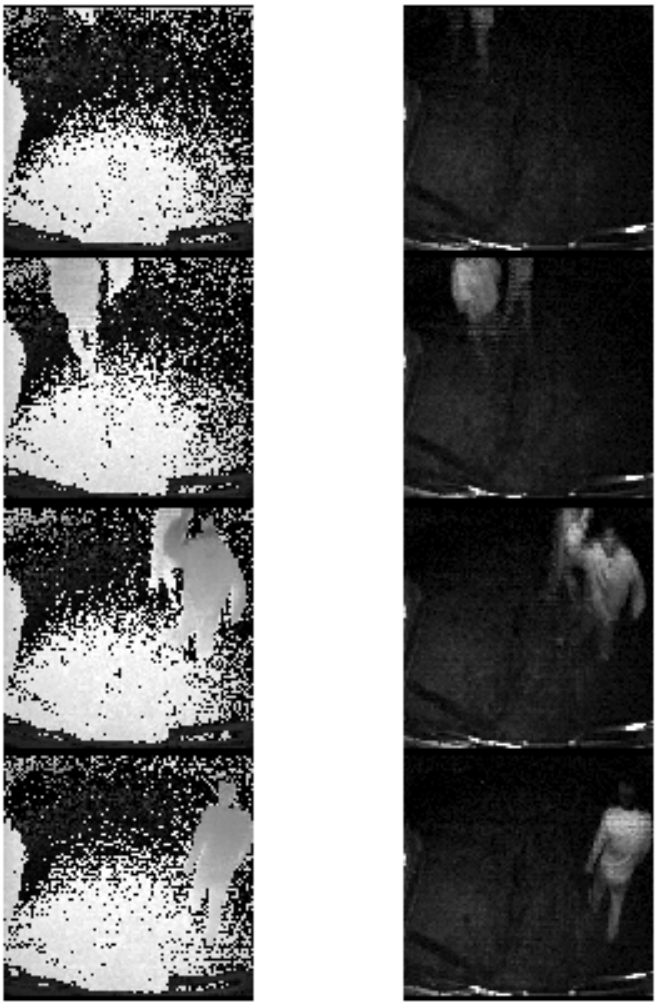


Figure 6.2: Frames from the outdoors evaluation sequence. Left: depth images, right: brightness images.

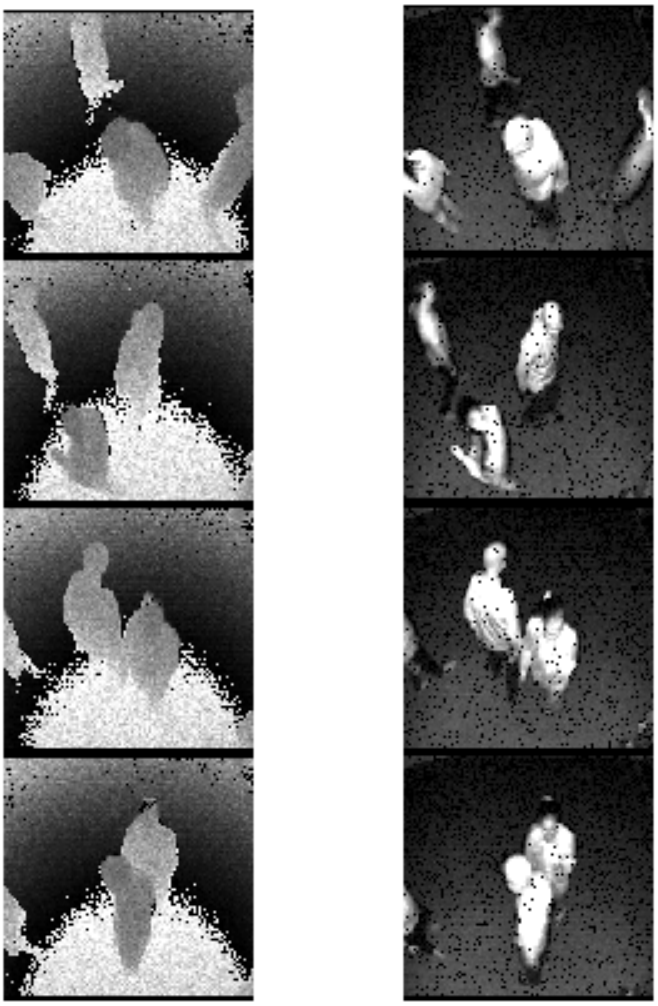


Figure 6.3: Frames from the indoors evaluation sequence with added noise. Left: depth images, right: brightness images.

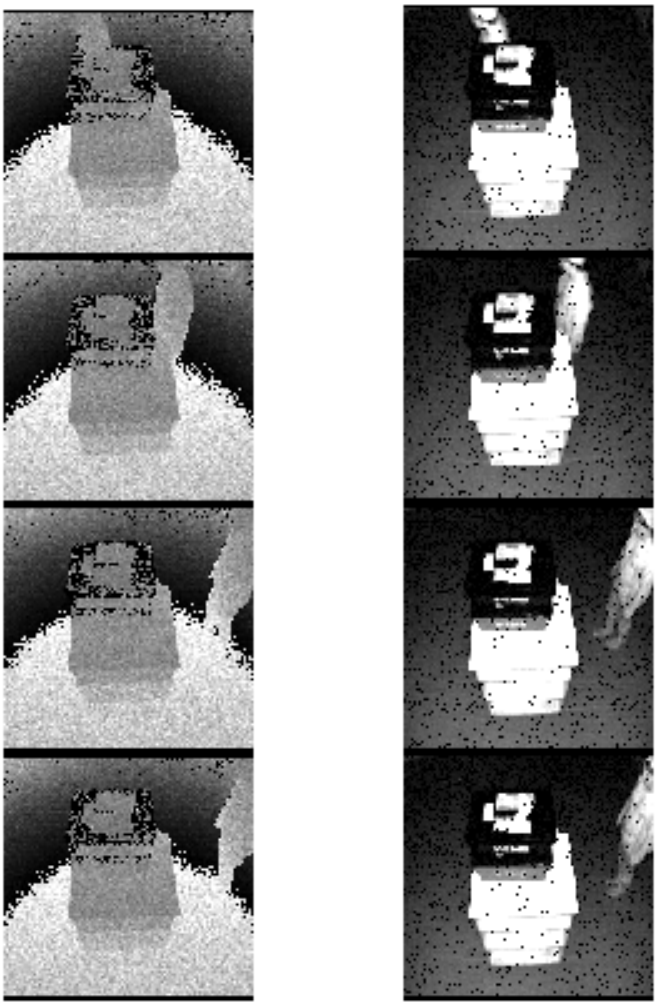


Figure 6.4: Frames from the 2OBJ evaluation sequence with added noise. Left: depth images, right: brightness images.

### 6.1.1 Detection evaluation

The detection evaluation comprises of running the data processing on the outdoor sequences and manually noting the position of an object whenever it is visible in the sequence while not producing a detection. Of course, since an object is not detected the position is merely an estimate. The occupancy grid cells which are part of a detection are also stored, and combining the two we get a picture of the area where robust pedestrian detection can be made.

#### Results

Figure 6.5 shows the areas where pedestrians have were, and were not, detected. It is worth mentioning that pedestrians were walking all over the field of view, so a white cell really indicates detection failure. The

lack of crosses further away from the camera can be explained by that objects at those distances could not even be detected by eye from the recordings, although this judgement probably varies somewhat between different observers.

While the area with successful detection is smaller than anticipated, it seems less imperfect when the data is analyzed further. It is primarily the confluence of five factors that shrinks the detectable area. First off, the further away from the camera, i.e. the further up and to the sides of the images captured, the more ground area is captured for a given image area. The area which is large on the occupancy grid is therefore a smaller part of the image. Secondly, the further away an object is, the more it is cut off height-wise due to the narrowing maximum height the FOV can contain. Simply due to this, an object consists of fewer pixels than it would otherwise. Of course, the object also shrinks, in terms of pixels, due to its distance. While we have tried to take the latter into account by having a diminishing occupancy grid threshold (mentioned in 4.3.4), there is a limit to how small the threshold can be without false detections are spawned from - at those ranges - more unreliable measurements. Lastly, as established in 2.2.4, the measurement reliability decreases roughly linearly with distance. Therefore, the measurements projected onto the ground plane are more sparse and thus less likely to produce occupied cells in the occupation grid. These reasons indicate that significantly improving the detection rate at the top is not as simple as it might seem. That is, without introducing false detections.

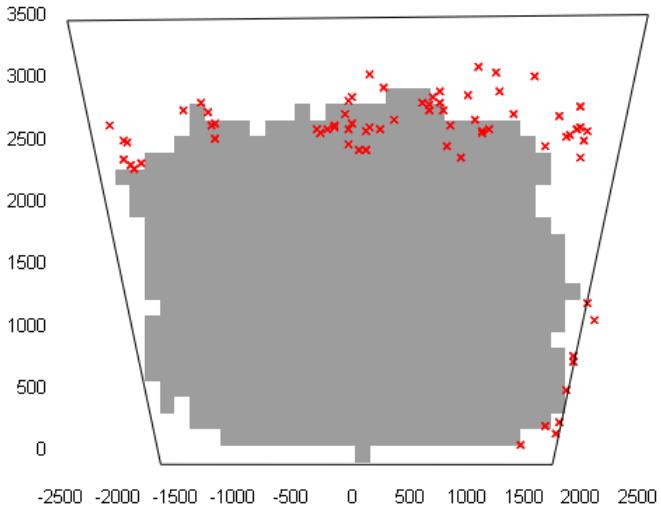



Figure 6.5: The grid cells which were part of a detection (grey squares), and the missed objects (red crosses) from the captured outdoor sequences (data sequence TRUCK).

| **Number of instances of ...** | |
|---|---|
| ... missed detections | 292 |
| ... found detections | 11776 |
| Fraction | 2.48% |

Table 6.2: Missing detection rates.

### 6.1.2 Tracking evaluation

#### 6.1.2.1 Tracking evaluation, target position

Another feature of the tracking is that it, of course, gives an estimation of the object positions. To evaluate the ability of the system's ability in general, and the tracking algorithm in particular, we have recorded two sequences where a pedestrians walks in a straight laterally at two different distances. The real-world forward-distances, 0.85 and 1.85 meters, are then compared with the position returned from the three different tracking algorithms.

**Results**

Figure 6.6 shows all the results in one plot.

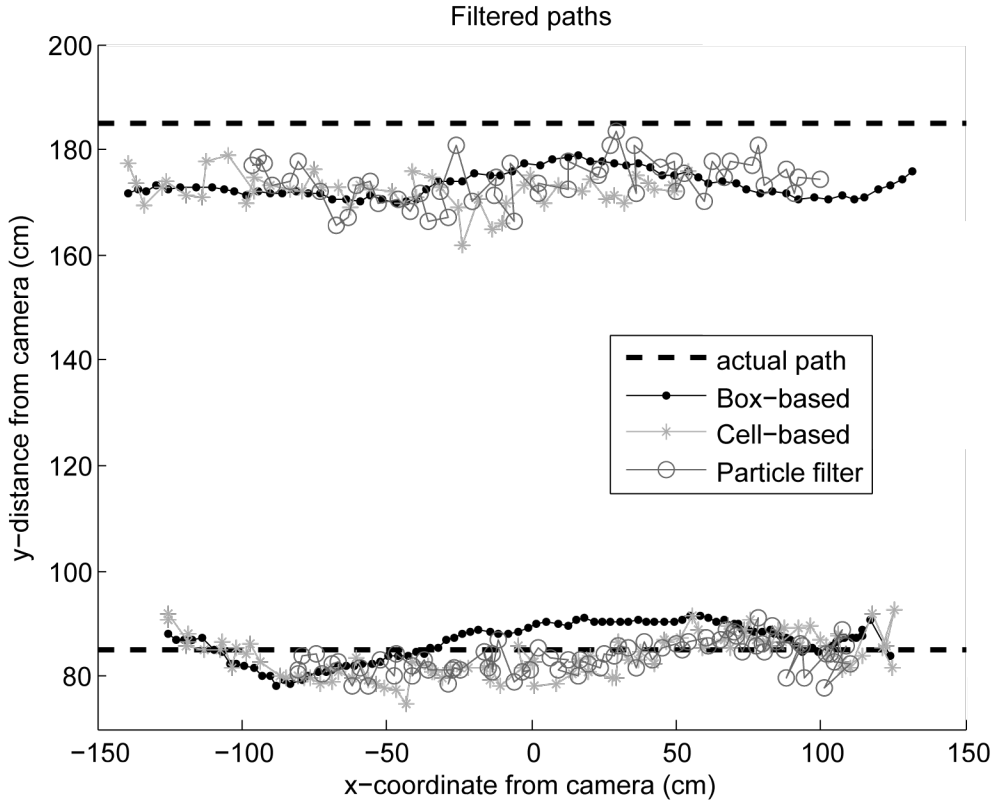



Figure 6.6: The estimated position of a pedestrian walking horizontally at 0.85 and 1.85 meters distance, estimated by the three tracking algorithms.

The $y$-bias for the path at 1.85m, for all tracking algorithms, is explained by that the back-side of an object is in shadow from the front side and thus the object's size in that dimension is smaller and seems closer to the camera. This in contrast with the path at 1 meter where the camera is practically right above the pedestrian and therefore the portion of the pedestrian in shadow is significantly smaller.

Although not quite a straight line, the box-based algorithm produces a relatively smooth path compared to the other two. It is worth to mention that the occupancy grid cell width was set to 15cm for both the Box-based and Cell-based algorithms. Therefore, it is a mean of discrete 15cm-spaced points which are filtered, while the same distance is 11cm for the particle filter. For the particle filter, the jaggedness becomes understandable if one recalls that the target position is based on the (non-filtered) mean position of the particles in a cluster of cells. However, the filtered Cell-based estimated position ought, ideally, to be smoother. It is a tell-tale sign of the difference in effort we put into the different algorithms; it probably suffices with some further tinkering to get a smooth tracking path for both algorithms.

#### 6.1.2.2 Measurement and tracking evaluation, target size

Arguably the simplest form of object classification based on 3D data is basing it on the target size dimensions. Therefore, to answer the question whether the sensor can perform object classification, we perform a brief verification of the algorithms' abilities to estimate target sizes. The estimated size in the three dimensions (width, depth and height) by the three tracking algorithms can be seen in Figure 6.7, 6.8 and 6.9.

**Tracking evaluation, target size: Box-based**

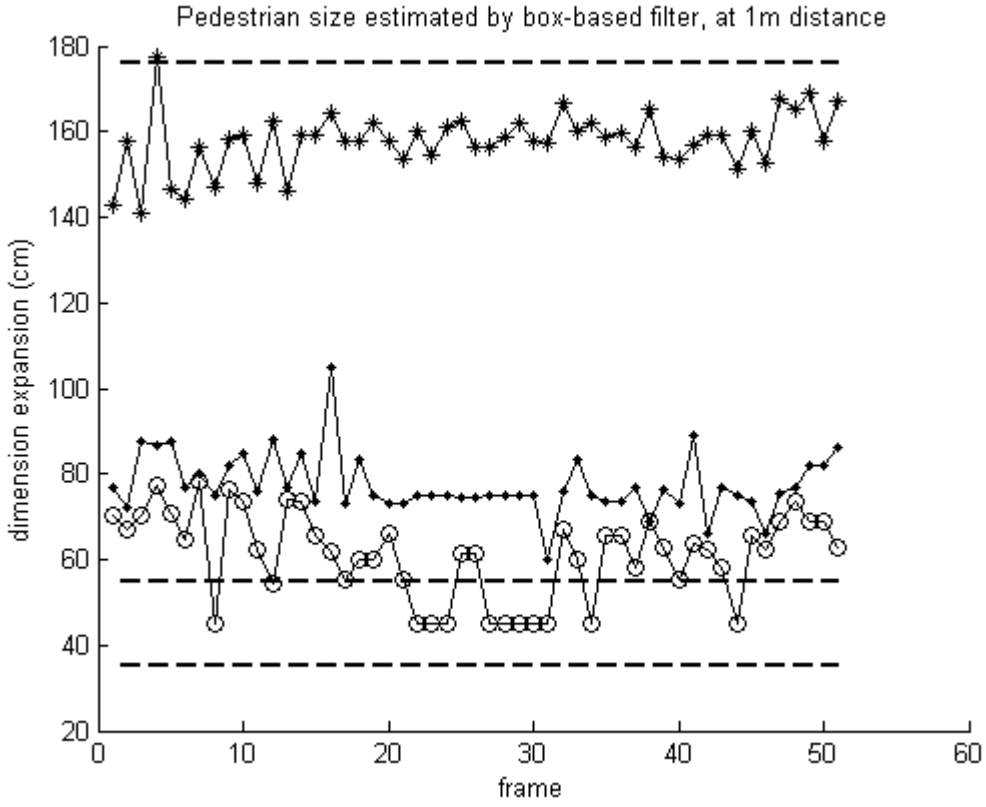



Figure 6.7: The estimated size (height, width and depth) of a pedestrian walking horizontally at 1 meters distance, estimated by the box-based algorithm.

**Tracking evaluation, target size: Cell-based**

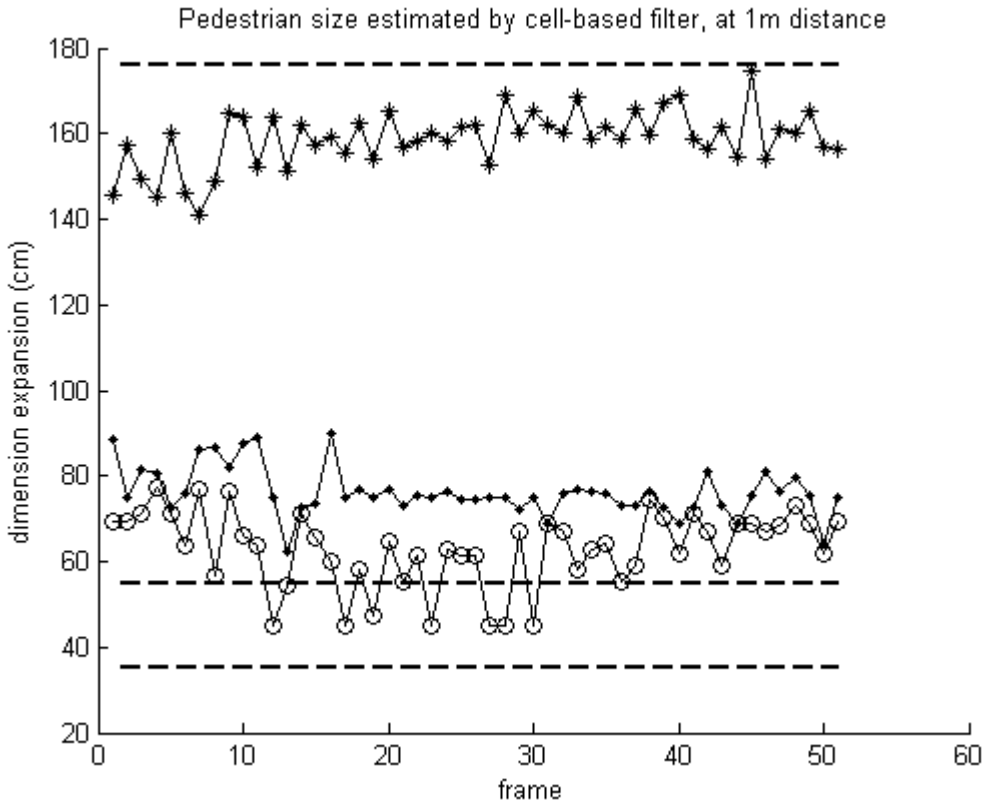



Figure 6.8: The estimated size (height, width and depth) of a pedestrian walking horizontally at 1 meters distance, estimated by the cell-based algorithm.

**Tracking evaluation, target size: Particle filter**

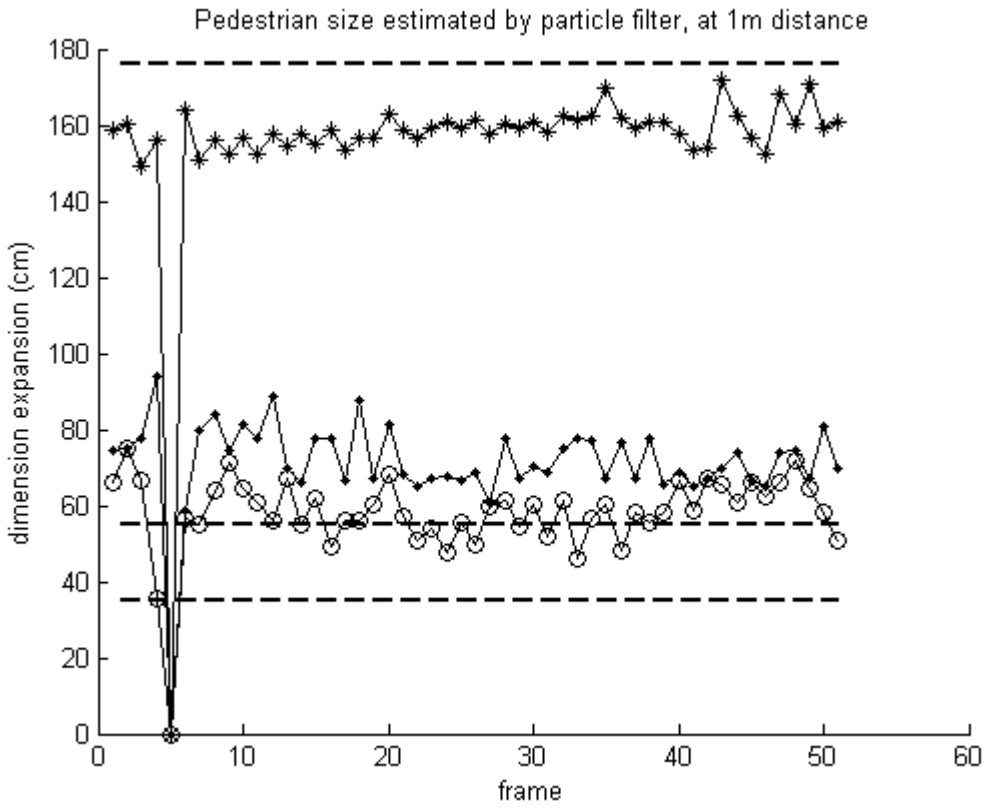



Figure 6.9: The estimated size (height, width and depth) of a pedestrian walking horizontally at 1 meters distance, estimated by the particle filter.

#### 6.1.2.3 Tracking evaluation, number of targets

Here we evaluate the tracking algorithms' abilities to identify the correct number of targets. The evaluation is based on three numbers: the true number of objects (e.g. pedestrians), $N^k_{objects}$, as a reference, the number of targets found, $N^k_{tracks}$, and lastly the number of detections, $N^k_{det}$, all for each time step, $k$, in a sequence. Recall that a single detection can contain several objects. From these numbers, three different error values are calculated, the first one being the average deviation between the number of objects in the FOV and targets actually found:

$$|\overline{e}| = \frac{1}{K}\sum_{k=1}^{K}|N^k_{tracks} - N^k_{objects}| \tag{6.1}$$

The other two error values are taken from different frame subsets. The first frame subset consists simply of all of the frames, and is denoted $ALL$. It is intended to give an evaluation more of the system as a whole (but not it's ability to distinguish clustered objects). The second set, denoted $DIFF$, consists only of the frames where the number of detections deviates from the true number of objects is intended to show the detection-splitting ability of the system. Naturally, the respective error value for each of these frame sets also reflect the purpose of the sets. These are denoted $|\overline{e}_q|$ and are defined as:

$$\text{based on } ALL: \quad |\overline{e}_q| = \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\frac{|N^k_{tracks}-N^k_{objects}|}{N^k_{objects}} \tag{6.2}$$

$$\text{based on } DIFF: \quad |\overline{e}_q| = \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\frac{|N^k_{tracks}-N^k_{objects}|}{|N^k_{det}-N^k_{objects}|} \tag{6.3}$$

#### 6.1.2.4 Tracking evaluation, number of targets: Box-based

Table 6.3 contains the error rates for the three data sequences and the different sets of frames. Figure 6.10 shows the distribution of $N^k_{tracks} - N^k_{objects}$ for all $k$.

| **Data sequence** | **Frame set** | **Avg. absolute value of error**, $\|\overline{e}\|$ | **Avg. error fraction**, $\|\overline{e}_q\|$ |
|---|---|---|---|
| INDOORS | $ALL$ | 0.629 | 0.222 |
| | $DIFF$ | 0.830 | 0.586 |
| TRUCK | $ALL$ | 0.203 | 0.105 |
| | $DIFF$ | 0.672 | 0.656 |
| 2OBJ | $ALL$ | 0 | 0 |
| | $DIFF$ | 0 | 0 |

Table 6.3: The results from the Box-based tracking evaluation

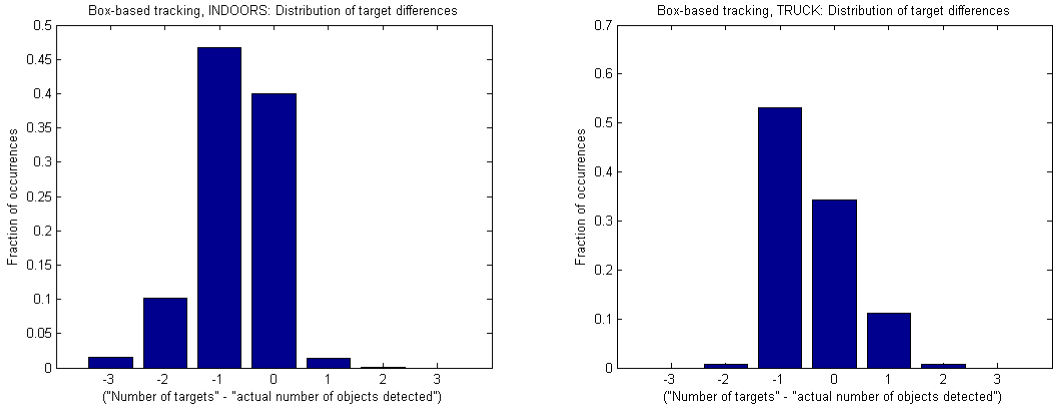



Figure 6.10: The distribution of differences of targets found and actual number of objects detected for all frames in the sequences.

The error rates are both promising and discouraging. The contrast between the flawless separation in the 2OBJ data sequence and the majority of failed separated targets in the other two sequences is eye-catching. The explanation for this significant difference in performance is not that 2OBJ is recorded indoors, since INDOORS is also that, but is probably found in the behaviour and quantity of the pedestrians.

In the 2OBJ sequence, the pedestrian never leaves the FOV -or even stands close to an edge- and thus the tracking algorithm never has to choose between: i) that a (vanished) target really has left the FOV, ii) associating it with a nearby detection or iii) associate the target with the unknown area behind the nearby detection. A first step to remedy this, to the extent it is possible with the algorithm, is to run it on more data sequences and adjust the likelihood parameters.

Although not often, it also happens that pedestrians are occlude each other and thus the algorithm has to do the same decision as above, and sometimes fails.

#### 6.1.2.5 Tracking evaluation: Cell-based

Table 6.4 contains the error rates for the three data sequences and the different sets of frames. Figure 6.11 shows the distribution of $N^k_{tracks} - N^k_{objects}$ for all $k$.

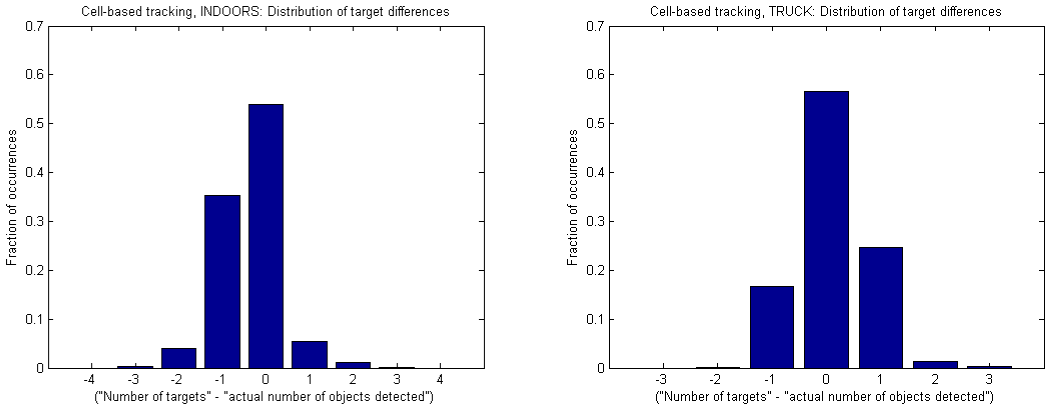



Figure 6.11: The distribution of differences of targets found and actual number of objects detected for all frames in the sequences.

| **Data sequence** | **Frame set** | **Avg. absolute value of error**, $\lvert\overline{e}\rvert$ | **Avg. error fraction**, $\lvert\overline{e}_q\rvert$ |
|---|---|---|---|
| INDOORS | $ALL$ | 0.435 | 0.154 |
| | $DIFF$ | 0.560 | 0.425 |
| TRUCK | $ALL$ | 0.226 | 0.112 |
| | $DIFF$ | 0.456 | 0.440 |
| 2OBJ | $ALL$ | 0.007 | 0.003 |
| | $DIFF$ | 0.143 | 0.147 |

Table 6.4: The results from the Cell-based tracking evaluation

What is to note here is that the Cell-based algorithm performs significantly better than the box-based in the more realistic data (TRUCK, INDOORS), while not in 2OBJ. The issues mentioned for the Box-based also hold for this algorithm, as it shares the data association process.

#### 6.1.2.6 Tracking evaluation: Particle filter

Table 6.5 contains the error rates for the three data sequences and the different sets of frames. Figure 6.12 shows the distribution of $N^k_{tracks} - N^k_{objects}$ for all $k$.

| **Data sequence** | **Frame set** | **Avg. absolute value of error**, $\lvert\overline{e}\rvert$ | **Avg. error fraction**, $\lvert\overline{e}_q\rvert$ |
|---|---|---|---|
| INDOORS | $ALL$ | 0.71 | 0.737 |
| | $DIFF$ | 0.841 | 0.73 |
| TRUCK | $ALL$ | 0.356 | 0.149 |
| | $DIFF$ | 0.722 | 0.693 |
| 2OBJ | $ALL$ | 0.547 | 0.274 |
| | $DIFF$ | 0.445 | 0.445 |

Table 6.5: The results from the particle filter tracking evaluation with regards to the number of targets.

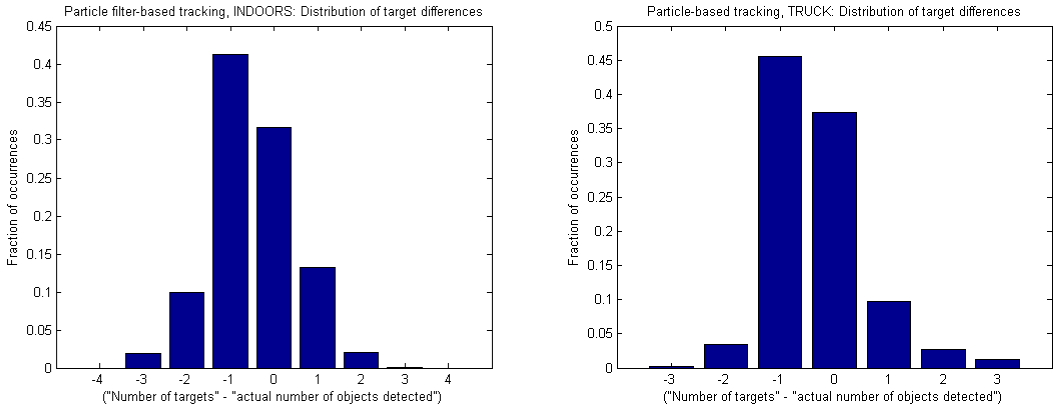



Figure 6.12: The distribution of differences of targets found and actual number of objects detected for all frames in the sequences.

As seen in 6.12, there is a relatively large fraction of frames where the number of targets exceeds the number of detections. This is largely due to that targets are often spontaneously split into smaller ones on their fringes, with no apparent reasons given by the input depth data. We experimented with remedying this by letting the velocities of particles of the same target homogenize slightly for each time step, but with no discernible improvement.

### 6.1.3 Speed evaluation

To test whether the algorithms can realistically run in real-time ($\approx$ 25 frames per seconds or more), they are timed on the data sequences (with 3-4 pedestrians) on a 5 year-old, then-high-end, 2.4GHZ desktop computer. The measured frame rates can be seen in 6.6.

| **Algorithm** | **Data sequence** | **Frame rate (frames / sec)** |
|---|---|---|
| Box-based | INDOORS | 14.4 |
| | TRUCK | 21.0 |
| Cell-based | INDOORS | 5.8 |
| | TRUCK | 11.2 |
| Particle | INDOORS | 17.1 |
| | TRUCK | 18.25 |

Table 6.6: The results from the overall frame rate evaluation. For the particle filter, the number of maximum particles per cell were set to $N_c = 100$.

First off, one can see that the varied frame rate shows that the computational demand of the tracking relative to the whole system is large.

As the algorithms was implemented in MATLAB script, the benefits of a transfer to another language in terms of speed depends heavily on the code style. We have made some effort to make fast code, most importantly by using built-in functions and matrix operations to as large extent as possible and avoiding loops, but more could be done. With a language transfer and some optimization, the performance of the algorithms, especially the Box and Cell-based, could probably be increased significantly.

The particle filter and box-based performs significantly better than the cell-based in this regard, and we conclude them to be possible to run real-time in e.g. C++ with similar computing power. The major reasons for the difference is speed are probably that we have focused more on optimization on the Box-based and the particle filter code is more suited for MATLAB due to its heavy use of matrix-operations. For the Cell-based algorithm the crucial difference may be that the likelihood of target-detection match is calculated multiple times for each target occupying shared detections. As was described in 5.1.6.2, a target in the

Cell-based algorithm iteratively occupies shared detection which is a loop that is called a lot, and thus ought to be significantly faster in another language.

# Chapter 7

# Conclusions and discussion

## 7.1 Evaluation of sensors

Since neither the Fotonic B70 nor the stereo camera built during the project seemed to have good enough specifications to actually be used in a safety system for production, we have not considered it necessary to evaluate their performance at great depth. However, the B70 is inadequate mainly because of its low performance in sunlight and limited FOV, although the area covered by it could be improved somewhat with higher elevated mounting. The stereo camera provided images lacking the details the stereo matching algorithm needs for finding correlated pixels. But as we have learnt during the project, stereo vision is a mature technology that has proven itself good enough to be implemented by large Japanese and German auto-mobile manufacturers for the task of pedestrian detection [13] [11]. Regarding the TOF technology, it too has reached far enough to be seriously considered for auto-mobile implementations, Audi has for example indicated to use the technology in the near future[12].

### 7.1.1 Field of view

We proposed a placement as high up as possible with the camera facing downward to achieve maximum coverage of the blind spots areas just in front of the truck. Despite this the FOV does not seem satisfactory in both lateral and forward direction, with a reliable forward pedestrian detection distance of $\approx$ 2.4m as shown in Figure 6.5. If one wants effective warnings at velocities of much more than 15km/h (the limit depending on the assumed reaction time for the driver or system) while still covering the immediate area in front of the truck cabin, the effective FOV needs to be extended in the forward direction. Of course, for slower velocities, and certainly for start-inhibition, our conclusion is that the combination of the B70 and the algorithms produced perform satisfactory. As mentioned, good performance assumes that the background IR lighting is low.

The demands on the sensor may differ between trucks and cars, however, so just because the sensor technology is mature enough for car market deployment it may not be quite there yet for trucks. Cars do not have the same problem with close-range frontal blind spots that trucks have, and are not as wide. Therefore trucks may pose higher demands on the FOV of the sensor to be able to detect objects at a sufficient lateral position in front of the truck. Our evaluation of the detection algorithm, boiled down in Figure 6.5, shows that objects are detected at lateral positions of $\approx \pm 1.8 - 2$m. For a 2.5 meter wide truck this means a margin of roughly 0.5 meters on each side at its least. Although we cannot see any theoretical reasons for why TOF cameras would have a more limited FOV than stereo cameras, the widest stereo camera we have found claimed to be manufactured has a FOV of 83°x60° [20] compared to the TOF camera maximum of 70°x53°[15]. This may be due to the fact that the cameras used for stereo vision are basically regular digital RGB/mono cameras which have been in production for decades whereas TOF cameras are relatively new.

### 7.1.2 Depth resolution

The depth resolution of the B70 TOF sensor is the same regardless of range, at least theoretically, its specific value being unknown to us. However, it provides values with millimetre precision which is concluded to be sufficient. What is more interesting is the accuracy which we present in Figure 2.9 and Figure 2.10. They show that the standard deviation of the measurements increase with the distance, as one would expect. In outdoor light the standard deviation of the measurements are of the order 10% of the measured distance which seems to be rather significant, and is likely to be the reason for the object detectability being poor at distances greater than 2.4 meters. This is however something that Fotonic claims to have improved in their latest generation sensors [15].

Resolution of the stereo sensor has not been evaluated quantitatively, but we can conclude that it decreases tangentially, i.e. roughly exponentially, with the distance, by looking at equation 2.2. For example, one of the stereo cameras described in [21] has depth resolutions of 0.31 cm to 48.2 cm for ranges varying from 1.25 meters to 15.92 meters with a horizontal FOV of 83°.

Thus one can safely conclude that both of the sensors can provide sufficiently high depth resolutions for the purpose explored in this thesis.

## 7.2 Tracking

The Kalman-based approaches proposed in this paper, especially the box-based variant, seem to give, adequate performance with regards to the number of targets, size and computational speed. Concerning its ability to correctly identify the number of targets, it provides the most accurate estimate of the three, its accurateness varies from perfect to merely adequate (in our opinion) depending on which data sequence was used. The data sequence which was designed specifically for the the task of number target estimation, and where the objects never moved in the uncertain edge areas of the FOV, was the sequence where it performed flawlessly in this aspect. The particle filter-based tracking algorithm needs, on the other hand, further adjustments since it has a relatively high tendency to both split detections into too many targets and fail to distinguish coalescent ones as shown in Figure 6.12. Furthermore, it lacks a mechanism, in contrast to the other two algorithms, that prevents keeping two adjacent targets, with sufficiently similar velocities, separate.

Evaluated in 6.1.2.2, for all three algorithms, the combined sensor and algorithm size estimation of pedestrians is not entirely accurate with an error of 10-40cm ($\approx 25-100\%$) in the width and depth dimensions, and a more acceptable error of about 20cm (11%) when estimating the height. Notwithstanding these large errors, a basic discrimination between e.g pedestrians, bicycles and cars ought still to be possible due to their significant difference in size. The magnitude of the error percentage would not be the same for larger objects either, such as cars, since we can see no connection between object size and error of size estimation. It is rather coupled with the grid cell size and measurement variation where the former is constant and the latter only increases with distance from the camera.

Regarding the algorithm speed, once more, the most polished Box-based algorithm performs well with a frame rate of 15-20 frames per second when tracking 3-4 objects. Although perhaps not adequate for real-time use in the implementation made in this project, it ought to perform adequately with some optimization and implemented in another language, e.g. C++. The same holds for the particle filter-based algorithm's 17-18 frames/s, with the relatively unpolished Cell-based having a significantly a longer way to go for usable speeds with its 6-11 frames/s.

## 7.3 Classification

As already concluded, successful classification of some of the most important objects (pedestrians, bicycles, cars..) in traffic is achievable with the B70, and thus also future improved models. An estimation of the size of an object as a 3D box is probably sufficient as a basis for discrimination of the mentioned three types of objects due to their distinct size and shape. Although the estimated size of a pedestrian is significantly enlarged in the $x, y$-dimensions, as established in 6.1.2.2, our conclusion is that the algorithms we explore could provide sufficiently accurate size estimations for the discrimination mentioned.

On the question whether or not the depth image needs to be coupled with an intensity image, we conclude that it is not necessary since the high density depth image ought to be sufficient information. We use the active brightness only to determine the validity of the corresponding depth measurements. Regarding classification however, we do think that the brightness information could enable more accurate classification of objects of similar size, since features can be found therein that are not available in 3D information.

## 7.4 General conclusions and thoughts

Regarding the question of the compatibility of the algorithms here proposed with other sensors and a Bayesian tracking framework, we think that there are both great possibilities and some deficiencies. Primarily the simplicity of the two-dimensional occupancy and unknown grids provides the possibility of also including the measurements from any type of ranging sensor. The data fusion could also be done later in the process. The Kalman-based algorithms give estimates of position, velocity and size that are associated with normally distributed probabilities. This should be enough to give the possibility of merging with targets given from other sources. A drawback is the lack of the probability, or some confidence measure, of a target being a true target. At a higher level fusion, this information could be used, for instance, to further reduce the number of false positives by removing uncertain targets that are only detected by a single, or few, sensors.

The theoretical upper bounds of the performance of the system as questioned in the goals is rather hard to grasp. The short answer is that there is no specific intrinsic limit of the technologies to our knowledge, but we have not studied the hardware components of the TOF cameras. The long answer is extremely difficult to give since all three variables are co-dependent; broader FOV means lower accuracy and vice versa, higher accuracy and broader FOV mean lower speed, improving all three aspects at the same time means higher cost. If the scope is limited to the available sensor we can increase the accuracy of the occupancy grid or the speed of the computer and improve the performance somewhat, however the B70's sensitivity towards light is the most limiting factor.

# Bibliography


[1] Hashimoto Y et al. Photo-sensitive-area modulation pixel for 3D real-time CCD imager. No date [cited 2013 Feb 28]. Available from http://www.imagesensors.org/PastWorkshops/2011Workshop/2011Papers/P24_Hashimoto_3D-CCD.pdf

[2] Hashimoto Y, Kurihara F, Tsunesada F, Imai K, Takada Y, Taniguchi K. 3 D real-time CCD imager based on Background-Level-Subtraction scheme. 2007 International Electron Devices meeting Technical Digest pp. 1019-1022.

[3] Devy M, Boizard J-L, Botero Galeano D, Carrillo Lindado H et al (2011). Stereo-vision Algorithm to be Executed at 100Hz on a FPGA-Based Architecture. In Advances in Theory and Applications of Stereo Vision. Editor Dr Asim Bhatti (Ed.), ISBN: 978-953-307-516-7, InTech, DOI: 10.5772/14037. Available from: http://www.intechopen.com/books/advances-in-theory-and-applications-of-stereo-vision/stereovision-algorithm-to-be-executed-at-100hz-on-a-fpga-based-architecture

[4] Volvo 3P. Volvo 3P European Accident Research Safety Report. Version 1. Gothenburg & Lyon ARTs. 2011-09-15.

[5] Danescu R. Oniga F. Nedevschi S. Modeling and Tracking the Driving Environment with a Particle Based Occupancy Grid. IEEE Transactions on Intelligent Transportation Systems. 2011; 12(4): 1331-1342.

[6] Gordon N. Salmond D. Smith A. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings on Radar and Signal Processing. 1993; 140:107-113.

[7] Kay SM Fundamentals of Statistical Signal Processing: Estimation Theory. Vol 1. Prentice Hall. 1993.

[8] http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf. Department of Computer Science-vUniversity of North Carolina at Chapel Hill; 2006 [updated 24 July 2006; cited 9 January 2012].

[9] Sandblom F. Filtering and Modelling for Automotive Safety Systems [unpublished dissertation]. Gothenburg: Chalmers University of Technology;2011

[10] Bar-Shalom Y. Rong Li X. Kirubarajan T. Estimation with Applications to Tracking and Navigation. John Wiley & Sons. 2001.

[11] Automotive News Europe. Continental develops camera-based collision-prevention system for Mercedes [homepage on the Internet]. No date [cited 2013 Feb 28]. Available from http://europe.autonews.com/apps/pbcs.dll/article?AID=/20121026/ANE/121019874/continental-develops-camera-based-collision-prevention-system-for

[12] SpectroNet. Integrated Safety - Challenges, Trends and Influence on Systems [homepage on the Internet]. No date [cited 2013 Feb 26]. Available from http://spectronet.de/portals/visqua/story_docs/vortraege_2010/101118_pmd_muenchen/101118_07_gollewski_audi_ag.pdf

[13] The truth about cars. Subaru Uses Stereo 3D Tech in New EyeSight ADAS [homepage on the Internet]. No date [cited 2013 Feb 22]. Available from http://www.thetruthaboutcars.com/2012/03/subaru-uses-stereo-3d-tech-in-new-eyesight-adas/

[14] SpectroNet. PMD Cameras for Automotive & Outdoor Applications [homepage on the Internet]. No date [cited 2013 Feb 26]. Available from http://spectronet.de/portals/visqua/story_docs/vortraege_2010/101118_pmd_muenchen/101118_08_frey_ifm.pdf

[15] Fotonic. FOTONIC E40/70. ROBUST REAL VISION, SMART CAMERA [homepage on the Internet]. No date [cited 2013 Feb 26]. Available from http://www.fotonic.com/assets/documents/fotonic_E40-70_high.pdf

[16] Morgan et al. BMC Public Health 2010, 10:699. http://www.biomedcentral.com/1471-2458/10/699

[17] Falie, D.; Buzuloiu, V., "Wide range Time of Flight camera for outdoor surveillance," Microwaves, Radar and Remote Sensing Symposium, 2008. MRRS 2008 , pp.79,82, 22-24 Sept. 2008

[18] Sedgwick D. Autoliv exec: Stereo cameras are next big thing in safety. Automotive News. Apr 2012. [cited 2013 Mar 13]. Available from http://www.autonews.com/apps/pbcs.dll/article?AID=/20120416/OEM02/304169991/1193#axzz2NMlvVQ2y.

[19] Wikipedia. Triangulation [homepage on the Internet]. [updated 6 March 2013; cited 2013 March 13]. Available from http://en.wikipedia.org/wiki/Triangulation.

[20] TYZX. DeepSea G2 Embedded Vision System [homepage on the Internet]. No date [cited 2013 Feb 26]. Available from http://www.tyzx.com/PDFs/tyzxDS_deepSeaG2VISION2.pdf.

[21] TYZX. DeepSea Stereo Cameras [homepage on the Internet]. No date [cited 2013 Feb 26]. Available from http://www.tyzx.com/PDFs/Tyzx%20DS%20Cameras.pdf.