Detection of military objects in LADAR images

Hans Christian Palm, Trym Vegard Haavardsholm and Halvor Ajer

Norwegian Defence Research Establishment (FFI)

12 June 2008

FFI-rapport 2007/02472

1020

P: ISBN 978-82-464-1389-1
E: ISBN 978-82-464-1390-7

Keywords

Avstandsbilder

Preprosessering

Segmentering

Deteksjon

Approved by

Johnny Bardal Director

English summary

This report describes different techniques for preprocessing, segmentation, and detection of vehicle sized objects in LADAR images. Five preprocessing strategies are presented; 1) Median filtering, 2) Two 1-D median filters in cascade, 3) Spoke median filter, 4) Donut filter, 5) Outlier detection and removal. The spoke median and donut filters were virtually worthless. The other filters worked equally well. The outlier detector removed outlers while perserving edges and small structures (and image noise). Concerning segmentation algorithms, we have implemented and tested four groups of region based algorithms and one group of edge based algorithms. Output from the segmentation is input to an object definition algorithm. Two strategies are proposed; one conventional agglomerative clustering approach, and one graph based approach. In essence, they both give the same results. Clusters with height, width, and length within predefined intervals are assumed to be possible objects. All algorithms are tested on real data of various vehicles in different scenes. It is difficult to draw any general conclusions. However, it seems that the region based algorithms perform better than the edge based ones. Among the region based strategies, those based on morphology or filtering operations perform well in most cases.

Sammendrag

Denne rapporten beskriver ulike teknikker for preprosessering, segmentering og deteksjon av kjøretøyliknende objekter i LADAR-bilder. Fem ulike preprosesseringsstrategier er presentert; 1) medianfiltrering, 2) to 1-D medianfiltre i kaskade, 3) spokemedianfilter, 4) smultringfilter, 5) outlierdeteksjon med påfølgende fjerning av outliere. Spokemedianfilteret og smultringfilteret var i praksis ubrukelige. De resterende filtrene hadde praktisk talt samme yteevne. Når det gjelder segmenteringsalgoritmer har vi implementert og testet fire grupper av regionbaserte algoritmer og en gruppe med kantbaserte algoritmer. Output fra segmenteringen er input til en objekt-defineringsalgoritme. To ulike strategier er foreslått; en agglomerativ clusteringsalgoritme og en basert på grafer. Essensielt gir de identiske resultater. Clustre med høyde, bredde og lengde innenfor predefinerte intervall er antatt å være mulige objekter. Alle algoritmene er testet på reelle data av ulike kjøretøy i ulike scener. Det er vanskelig å trekke noen generelle konklusjoner. Imidlertid ser det ut som at de regionbaserte algoritmene har en større yteevne enn de kantbaserte. Blant de regionbaserte algoritmene er det de som er basert på morfologi og filteringer som i de fleste situsjonene fungerer best.

Contents

1 Introduction

This report presents results of our work done on automatic detection of military objects in three-dimensional (3D) images from ladar[1] sensors. The objective of this work is to gain knowledge of and develop methods for evaluating the performance of such sensors in munition applications, in particular in the role as missile seekers for ground-target engagements. Examples of such applications are various concepts for loitering missiles. This particular application places some constraints on both the actual sensors and the measurement geometry, both of which will be discussed briefly in section 1.1 and 1.2. The work has been part of an FFI project[2] addressing, among other topics, future technologies for ground-target engagement with indirect-fire systems. Earlier studies of indirect-fire concepts pointed to a potential for high effectiveness and efficiency from systems comprising loitering missiles with ladar seekers [1]. The main uncertainty in that study was connected with the performance of the missile-seeker algorithms to automatically detect, classify, and recognize targets. Hence, this work seeks to address these questions.

The success of ladars in ATR[3] applications will obviously depend heavily on the system's ability to automatically detect and classify targets. As initially mentioned, the focus of this report is the detection process. This can often be described by the following steps: First a preprocessing is applied. The aim of this is to reduce noise and/or outliers while preserving the object and scene structure. Then, a segmentation algorithm is applied. The segmentation in general partitions the image into regions where all pixels in each region are similar according to a given similarity measure, but dissimilar to pixels in adjacent partitions. In our report, pixels are divided into the categories "interesting" and "not interesting". "Interesting pixels" means pixels which may be part of an object, i.e. pixels higher than its surrounding terrain. Next, pixels close to each other are grouped into clusters. Finally, each cluster is investigated in order to find out whether it is likely to be an object of interest.

The report is organized as follows: In chapter 2 we will address preprocessing of the data. Then, the various segmentation algorithms are described in chapter 3. Next, the strategies for grouping/clustering the segments (point clouds) are presented in chapter 4. Chapter 5 gives a short presentation of the detection criteria. Thereafter, in chapter 6 we apply the algorithms on datasets from various scenes, and present and discuss the results. Finally, conclusions and some topics for future work are presented in chapter 7.

1.1 LADAR sensors

LADAR sensors are active sensors utilizing lasers to illuminate a scene and detectors to measure the return signals. Depending on the ladar design, various parameters characterizing each point

[1] LADAR: *LAser Detection And Ranging*.
[2] *Project 1020 – Indirect Fire*.
[3] ATR: Automatic Target Recognition.

(pixel) in the scene may be obtained. Examples are direction and range to each pixel, intensity of the reflection, relative velocity, polarization, and vibration frequencies. The most common variant associated with the term LADAR is the 3D-imaging sensor. This type of ladar produces a range image where a range is associated with each point in the scene as specified by the direction in azimuth and elevation. Such range images can easily be converted to a point cloud in a given Cartesian coordinate system. Thus, they provide 3-dimensional geometric information about surface points on objects in a scene; in other words, what we get is a 3D image. Our work addresses data from such 3D imagers. Although our available data also include intensity information, the present report only considers the geometric information extracted from the scene.

It might be of interest to consider the differences between the characteristics of 3D-imaging ladar sensors compared to traditional imaging sensors, such as daylight cameras, image intensifiers or infrared imagers. In the latter case, each point or pixel in the scene is characterized based on the properties of these points, e.g. color, reflectivity, temperature, or emissivity. In our case, with only using the range information, no properties of the individual pixels are recorded, only the positions of the points, relative to each other and some chosen coordinate system.

Another interesting effect of ladar sensors is the ability to partly "see" through sparse parts of a scene, such as vegetation and camouflage nets. This can be achieved by processing the temporal behavior of the reflected laser pulses, a very simple scheme being to discard all but the last puls(es).

So far, ladar sensors in the form of 3D imagers have not found widespread use in military applications, although various concepts have been under development. There are some practical challenges in the design of 3D-imaging ladars that may explain this. With the sensor being active, a certain amount of laser energy must be delivered on each point in the scene in order for the detector part of the sensor to measure the distance. The typical measurement method is based on measuring the time between an emitted laser pulse and the reflected return signal. Furthermore, requirements on the transverse spatial resolution together with requirements on the footprint size result in a certain number of measurements to be done for each footprint. In combination with the energy requirement for each point, this leads to a minimum required laser output energy for each sensor footprint. Similarly, with a certain rate of footprints (or "frame rate"), one gets a minimum required average laser output power.

Doing some rough estimates for a small example might illuminate the problem: Consider an application where a ladar sensor in a missile is looking forward in a slant downward angle, let's say 20° under the horizontal. If one requires a transverse spatial resolution of 20 cm on the target together with a footprint width of 200 m, a total of 1000 points is required for each line in the footprint. Furthermore, assuming a relatively flat and horizontal ground, the transversal lines in the image then have a spacing of about 58 cm. Assume a missile speed of 100 m/s and that consecutive footprints are connected without any overlap. Then, about 171 lines or about 171000 points must be measured every second. Compared to a typical laser range finder doing measurements at the same distance at a rate of one pulse per second, this corresponds to 171000 times higher average power. The requirement on laser output power, pulse rate and pulse energy, combined with the application's demand

on small physical size, robustness, high efficiency, and low cost, makes this a challenge. What has made this feasible, are the advances in diode-laser technology in combination with diode-laser pumping of solid-state lasers.

It is possible to distinguish ladar design in scanning ladars and flash ladars. The former works by scanning one or a moderate number of beams across the scene to build the image. The latter illuminates the whole scene (footprint) with one laser pulse and then measures the return signals from each point in parallel. There are benefits and drawbacks with both schemes, but this will not be discussed further in this report.

However, given that ladar sensors are available, a number of military applications are obvious, e.g. missile seekers, as mentioned, in surveillance systems and for reconnaissance. They could e.g. be used as part of a sensor package on reconnaissance vehicles, or as sensors on UGVs[4] or UAVs[5]. LAM[6] is an example of loitering missile concepts which utilize ladar for detecting and classifying military objects.

As we shall see, a ladar sensor produces high-quality three dimensional data on a relatively long range. Research groups have reported impressive results (e.g. [19], [7] and [3]) for detection and recognition of objects in range images. This more than indicates that ladar sensors have a great military potential.

1.2 Ladar images

We are primarily interested in imaging applications, and will therefore assume that the ladar sensor delivers range measurements in a 2D grid. This *range image* may be considered the primary output from a ladar sensor, and can be represented mathematically by

$$r(i,j) = R(\theta_i, \phi_j), \qquad i = 1, \ldots, N \; j = 1, \ldots, M \tag{1.1}$$

where $R(\theta_i, \phi_j)$ is the measured range along the azimuth and elevation angles (θ_i, ϕ_j). The range image $r(i,j)$ thus represents a 3D point cloud, where each pixel (i,j) corresponds to a point with the spherical coordinates $(r(i,j), \phi_j, \theta_i)$. Figure 1.1 shows an example of a range image, as well as a reflection intensity image.

It is often more useful to work with Cartesian coordinates when processing ladar images. We can easily obtain Cartesian images by performing the following transformation:

[4] UGV: *Unmanned Ground Vehicle.*
[5] UAV: *Unmanned Air Vehicle.*
[6] LAM: *Loitering Attack Missile.*

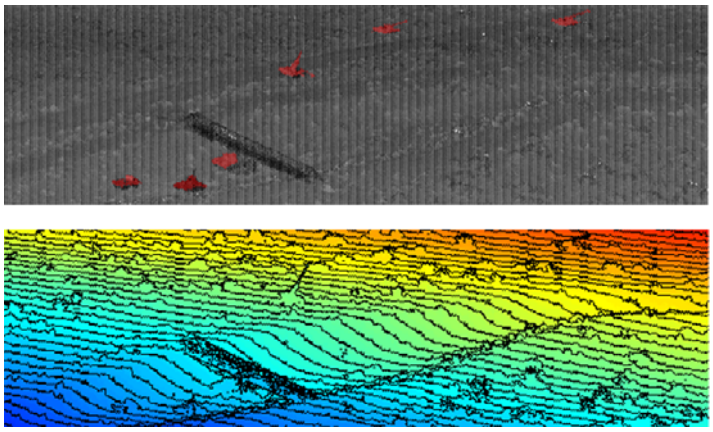

Figure 1.1: Example of the two primary outputs from a ladar sensor. The top image shows a reflection intensity image. Targets are shown in red for emphasis. The bottom image shows a range image of the same scan. The distance between each contour line is 5 meters.

$$\begin{aligned} x(i,j) &= r(i,j)\ sin\ \phi_j\ cos\ \theta_i \\ y(i,j) &= r(i,j)\ sin\ \phi_j\ sin\ \theta_i \\ z(i,j) &= r(i,j)\ cos\ \phi_j. \end{aligned} \tag{1.2}$$

The pixel (i,j) is now given by the Cartesian coordinate $(x(i,j), y(i,j), z(i,j))$. Since $x(i,j)$, $y(i,j)$ and $z(i,j))$ can be thought of as "'range images"' along each dimension, you may see us refer to them as the x-, y- and z-images. Figure 1.2 shows examples of real x-, y- and z-images, while figure 1.3 shows the corresponding 3D image.

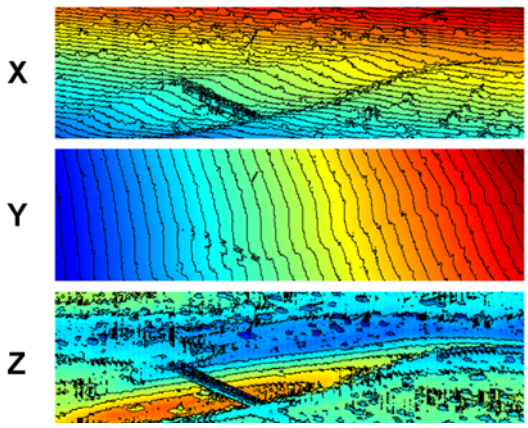

Figure 1.2: The x-, y- and z-images derived from the range image in figure 1.1. The distance between the contour lines is 5 meters for the x- and y-images, and 1 meter for the z-image.

The images we will be using in this report are recorded from the air as part of the CMRTR[7] program,

[7] CMRTR: *Cruise Missile Real Time Retargeting.*

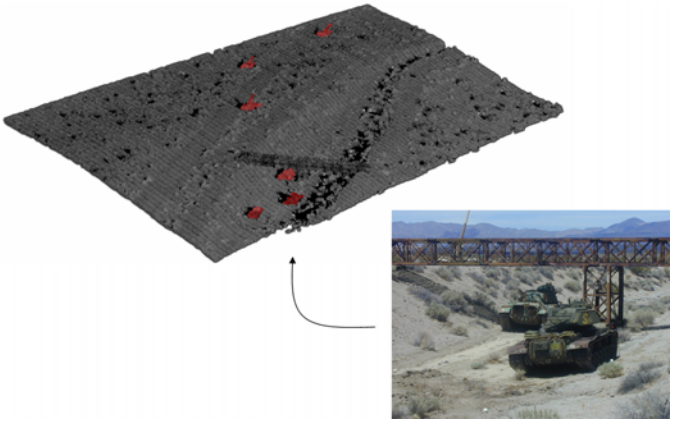

Figure 1.3: A 3D image of the scene in figure 1.1 based on the x-, y- and z-images in figure 1.2. The targets are shown in red. The bottom right image is a photo of some of the targets around the bridge.

which is led by NAVAIR in China Lake, CA, USA. The dataset has been made available through the NATO SET-077/RTG-45 research group. The Cartesian coordinate system used in this dataset is not specified, but is probably tied to the airborne platform. The exact definition of the coordinate system is not important, but it seems that the origin is at the position of the sensor or the platform, while the positive x-axis is pointing in the direction of the nose of the platform, and the positive z-axis is pointing down from the platform, so that it is typically pointing towards the ground[8] (see figure 1.4). The nature of this coordinate system has the following important implications:

- The XY-plane is generally *not* parallel to the ground plane (tangent to the local geoid), but is dependent on the orientation of the platform.
- Assuming that the platform is oriented to be more or less parallel to the ground plane, taller objects on the ground typically have smaller z-values.

The CMRTR images have all been recorded with a forward-looking slant-view sensor, as illustrated in figure 1.4. This geometry also has some important implications for this study:

- Raised objects, both potential targets and other parts of the scene, will occlude relatively large areas on the far side of the sensor. This will often include areas on the objects themselves as well. It is therefore difficult to extract the full top-down contour of targets.
- It will be difficult to observe the ground between nearby objects. This can result in problems with separating objects, as well as difficulty in estimating the ground level in areas with many objects, such as in dense vegetation.

[8]The sensor has in reality been in motion during scanning, so talking about a unique platform position and direction in this context is a simplified explanation meant to explain how we understand the coordinate system.

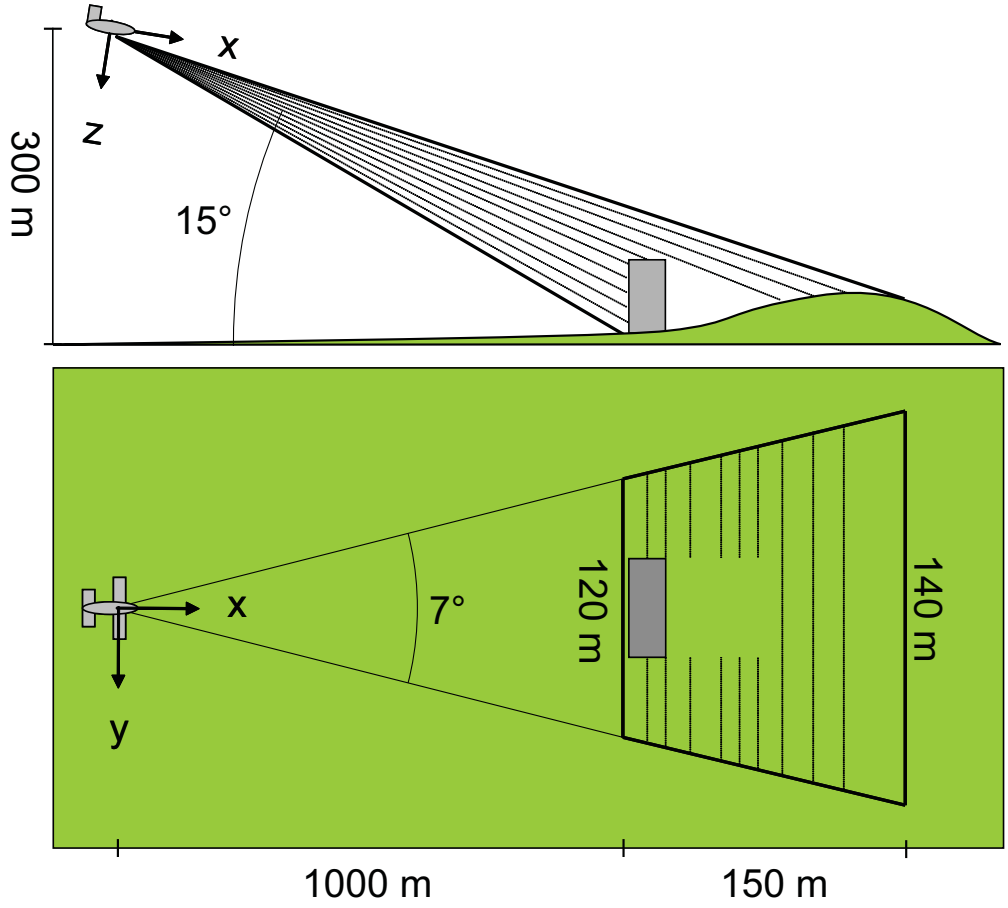

Figure 1.4: Side- and top-view illustrations of the forward-looking slant-view imaging geometry with typical values from the dataset used in this work. Notice the trapeze shape of the sensor footprint on the ground, as well as the shading-effects caused by the raised object in the scene. The coordinate system is associated with the airborne platform, and is independent of the scene and the sensor geometry.

2 Pre-processing

LADAR images might be somewhat "noisy". The most important component of this noise is isolated pixels whose range differs significantly from the ranges in neighbouring pixels. In some cases, this is not only true for isolated pixels, but larger regions as well. There are several sources contributing to this noise. Particles in the air, the reflective properties of the terrain, and general sensor inaccuracy are some possible sources.

But whatever the cause may be, the noise will in general have consequences for the detection of objects. The noise must be handled in some way. Simultaneously, it is important to preserve edges between objects and terrain as well as scene and object structure. For example, an important feature to look for in range images, is the large leap in range between the upper parts of an object and the underlying background. If the pre-processing filter removes or obscures this leap, it could have dramatic consequences for the subsequent object detection process. An important requirement for the pre-processing filter is therefore to remove the noise *while* preserving the *shape* of the objects.

Pre-processing has not been an important subject in LADAR processing literature. This is probably because most of the images that have been used are of high resolution and little noise. The need for pre-processing has thus not been very great. In the extent that pre-processing has been used, it has usually only been commented that one has implemented a *median filter*, which indicates that this filter is useful for these kinds of images. A median filter works on each pixel in an image in turn. The intensity values for the pixels in a small window around the "current" pixel is recorded, and the median of these is computed and stored in a (*new*) image. We have, however, found a filter that has been developed especially for pre-processing of LADAR images, [22]. Unfortunately, this method is very resource-demanding, therefore we have not explored it further[9].

There are, however, a large amount of edge-preserving smoothing filters for "regular" images described and evaluated in literature, see e.g. [5] and [17] for overviews. The median filter shows good results in these evaluations too. From the above remarks and our image processing experience in general, it is reasonable to assume that the median filter will also work well for LADAR images.

Instead of applying a 2D median filter, one may use two 1D median filters in cascade. First a $w \times 1$ filter is applied, and then a $1 \times w$ filter (to the $w \times 1$ - filtered image). This strategy is computationally very efficient, and the results are similar to that of the 2D median filter.

A problem with the median filter is that it *can* remove thin structures, such as a tank gun barrel, or details on a tank turret. This can be mended in several ways. We have implemented and tested two algorithms. The first one is a "spoke median filter", and it consists of the following steps. For each pixel:

[9] The method is based on both a removal of outliers by a (local) minimization of the *median* of a sum of squared deviations, and a following anisotropic diffusion.

1. Apply four 1D median filters (rotated $0°$, $45°$, $90°$, and $135°$ - hence the name; filtering is applied along four spokes). The filter length is an input parameter (we have used length 3).
2. The (filtered) value being closest to the (unfiltered) center pixel value is used as output value.

The second algorithm is a “donut filter”. It is illustrated in figure 2.1.

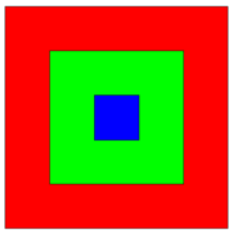

Figure 2.1: The donut filter. The filter consists of two square “donuts”. For each pixel, minimum and maximum values are found in each donut. Output value is the median of these four values in addition to the value in the center pixel.

Each pixel is as usual filtered in turn. For the current pixel, minimum and maximum values are found in each donut. Output value is the median of these four values in addition to the value in the center pixel.

An alternative to filtering the image, is to use an *outlier detector*. An outlier is a measurement which greatly deviates from its expected value. In our datasets, such measurements can be detected by looking at the difference between the filtered and unfiltered values of a pixel. If this difference is “large”, the filtered value is chosen. Otherwise the original value is used. The ordinary median filter has been used in the outlier detector.

We are now left with five possible strategies:

1. Using an ordinary median filter.
2. Using a cascade of two 1D median filters.
3. Using a spoke median filter.
4. Using a donut filter.
5. Using an outlier detector.

3 Segmentation

The segmentation process aims to divide an image into “interesting” and “non-interesting” areas. Since we are looking for military vehicles, the interesting areas will be vehicle-sized clusters of

pixels raised above the ground surface. The ground surface is not generally known, and may take many different shapes.

Moreover, the sensor's elevation angle is unknown. If the sensor axis had been perpendicular to the ground plane, the processing would have been easier. The local (object) height in a given pixel could then e.g. be estimated as the difference between the pixel value and the largest pixel value in the surrounding pixels in the z-image[10]. This information could in turn be input to a region growing process for estimating the ground level (e.g. [21]). However, in general this is not the situation; in UAV or missile applications, the sensor axis will not in general be perpendicular to the ground plane. This causes difficulties due to occlusions; an object (natural or man-made) occludes relatively large parts of the ground behind the object, and part of an object will also be occluded, both by itself and by nearby objects. The occlusion behind the object may both be an advantage and a disadvantage. It is an advantage in the sense that it is a useful feature for object *detection*, and it is a disadvantage because parts of the ground near an object cannot be used for ground level estimation. Occlusion of object parts is obviously a disadvantage only since the length and/or width may be wrong or unreliably estimated.

Our requirements to a segmentation algorithm are identical to those of other applications; we will emphasize:

- Ability to define the object silhouette.
- Robustness.
- Computational efficiency, i.e. real-time processing should be feasible.

3.1 Previous work

Even though LADAR sensors are not yet in any widespread use, a tremendous amount of papers has been published on segmentation, detection, and classification of objects in range images. We cannot provide a complete survey of the area, but will limit ourselves to give some general remarks, and briefly describe a few techniques which we in some sense have found interesting.

In general, image segmentation algorithms can roughly be divided into three categories:

1. Edge-based techniques.
2. Region-based techniques.
3. Primitive-based techniques.

Edge-based techniques apply edge operators to localize surface boundaries. They may be quite effective, but there are, however, several disadvantages involved with these techniques. In noisy images, it is difficult to detect edges robustly; the edges contain high frequency information, and

[10]The z-image is set of coordinates in the direction perpendicular to the sensor plane; in this case this means the distance from the sensor altitude to the ground/object.

so does the noise. Moreover, the edges often become fragmented, and it is difficult to link these fragments together in a reliable way. An example of an edge-based segmentation algorithm can be found in [24].

Opposed to edge-based techniques, region-based techniques determine areas, where the pixels in each area have some kind of similarity. Region-based segmentation comprises techniques like region growing, thresholding, morphological operations, and split & merge. Region-based techniques are usually more robust to noise, but have to rely on a good criterion for homogeneity. An example of a region-based segmentation algorithm can be found in [13].

Primitive-based techniques extract surface primitives directly from the range image. A typical primitive is a polynomial surface (e.g. plane or quadratic surface). Robust methods are often applied for estimating the primitive parameters. Thus, they are robust against outliers and (to some extent) occlusions. The main disadvantage is that they are relatively time consuming. Examples of primitive-based segmentation algorithms can be found in [10], [14], and [23].

In the following, some techniques will be briefly described.

Evaluation methodology

Hoover et al. have presented an interesting work performed by four image processing groups[11] in [11], where they propose a *methodology* for *evaluating* range-image segmentation algorithms. Their motive is simply that they prior to their work find the evaluation to be poorly treated in the publications. They claim that most publications neither utilize ground truth in the evaluation, nor compare results against other algorithms. The methodology of Hoover et al. involves a common set of 40 range images and 40 structured light scanner images that have manually specified ground truth. The resolution in these images is high, and the images themselves comprise planar shaped objects in scenes without any clutter. Furthermore, they define a set of performance metrics for instances of correct segmentation, missed, and noise regions, over- and undersegmentation, and accuracy of the recovered geometry. Each of the groups developed their own algorithms for segmenting range images into planar patches. These algorithms were in turn evaluated utilizing the common database and the common set of performance metrics. All algorithms were, however, based on segmenting the images into planar regions. Due to this limitation, we haven't studied these algorithms any further.

Robust segmentation of range-images

Staff at the image processing group of Monash University in Australia has published papers on robust segmentation. An example can be found in [23]. They are all model-based, i.e. the scene can be characterized by a set of parameters (e.g. parameters of planar patches). In brief, techniques from robust statistics are applied for segmenting range images. They claim that their algorithms are able to handle datasets with 50 - 80% outliers. Their results are evalutated using the dataset in [11]. The techniques are very interesting due to their ability to handle outliers. However, we haven't looked

[11] These groups were located at University of South Florida - Tampa, Washington State University - Pullman, University of Bern, University of Edinburgh.

into them because the object resolution often is too small for extracting any primitives like planar patches.

Range-profile tracking

DePiero and Trivedi have presented a technique they have named *range-profile tracking* (RPT) [6], which is based on Kalman filtering. The input is a (vertical) profile of the scene. Each line and column is treated as a time series and processed separately. Each profile contains (after being processed) a series of end-to-end curved segments, called strips, where each strip designates a portion of the profile having uniform curvature. The Kalman trackers have the capability to follow second order height variation. During processing, the height in the next sample is predicted. If the prediction error exceeds a predefined threshold, the current strip is terminated, and a new one initiated. The RPT is claimed to have real-time capabilities. Thus it is of interest for us. However, we have two reasons for not testing the RPT. First, we don't have any data from a (vertical) profiler. In our data, parts of the vertical profile is in general occluded. Second, we believe that the resolution in our images is too poor for reliable curvature estimation.

Fast algorithms for estimating ground plane

In [20], Stevens et al. have presented three simple algorithms for target detection in LADAR images based on estimating the ground plane. They are all computationally simple and thus suitable for real-time use, and hence they are of interest for us. We have discarded two of the algorithms from any further investigations due to their assumption of flat terrain. This is in general little realistic. However, the third algorithm called *top-down projection segmentation* is interesting because it doesn't make any assumptions regarding the terrain, and it is computationally very simple. This algorithm first projects the samples to a top-down view image. In this top-down view, pixels belonging to a vertical surface are projected into the same "top-down pixel". The maximum difference of the pixels being projected into a given top-down-pixel is considered as the object height for that particular top-down-pixel. This height image is then thresholded, dilated, and labeled.

Algorithms developed of NATO SET-077/RTG45 - groups

Members of the NATO SET-077/RTG45 research project N*-dimensional eyesafe imaging ladar* and their colleagues have published several papers through the years. We will here briefly sketch the recent publications we have found in open literature, which is not based on data captured from a vertical profiler.

- Neulist and Armbruster presented in [16] an application where the line of sight is almost parallel to the ground. Therefore the segmentation can be made very simple. It is based on a 3D connected components connection, which is defined by a distance threshold. This threshold depends on the range. Next, the ground plane is estimated. Due to few ground samples, the ground plane is assumed to be horisontal. Given the ground plane, the pixel heights can be calculated. Armbruster presented another algorithm in [2][12]. He starts with transforming the range image into a Cartesian coordinate system. One of its axis (z-axis) is perpendicular to the horizontal plane. A uniform grid is defined on the horizontal plane. For

[12] [2], p. 9

each grid point, its ground level is defined as the z-minimum in a local neighbourhood. The width of this neighbourhood has been selected to be 5 meters[13]. The height image can now easily be determined, and the segmented image is defined by thresholding the height image.

- English et al. [7] presented an algorithm for automatic target recognition and pose estimation. (Field tests are described in [18].) A segmentation algorithm is a part of this. However, this part have not been focused in their study; hence their segmentation is only semi-automatic. Target designation is manually performed by the user, and then the target is segmented using histogram-based range gating and region growing. Unfortunately, no details concerning the segmentation is given.
- Felip et al. [8] have proposed an algorithm based on robust statistics. The segmentation is based on modeling the ground, estimating the parameters, and then subtracting the ground component. The ground is modeled as a quadratic surface, and the parameters estimated with a least *median* of squares algorithm. Next, the pixels which do not belong to the surface are extracted, and then clustered. Moreover, the bounding box of each cluster is determined, and finally non-target clusters are tried removed by some object-filtering techniques. Algorithms based on robust statistics are applied in all stages in the process.
- Chevalier et al. [4] presented an algorithm for ground segmentation based on watershed and region growing. Input is a point cloud rotated so that the z-axis points upward. First the point cloud is rastered and a $z_{\min}$-image consisting of the lowest point in each pixel/bin is created. Next, a set of possible ground pixels are determined by a morphological watershed algorithm. Finally, these possible ground pixels are input to a region growing. If there are unvisited pixels when the region growing has terminated, the ground level for these pixels is estimated by applying bilinear interpolation.

3.2 Investigated algorithms

In the rest of this chapter, we will describe the segmentation algorithms we have developed for detection purposes. These algorithms have in common that they are computationally fast (if efficiently implemented), as well as suitable for relatively low-resolution images. They may be clustered into five groups:

1. Algorithms based on height estimates from local plane estimates in each pixel.
2. Algorithms based on height estimates from morphological operations.
3. Algorithms based on height estimates from filtering operations.
4. Algorithms based on height estimates from region growing.
5. Algorithms based on detection of vertical surfaces.

[13] If some grid points don't have any measurements within its neighbourhood, a ground level estimate is defined by interpolation or extrapolation.

The algorithms in the first four groups all produce a terrain or ground level estimate, which by subtracting the original image gives us an image representing the height above ground for each pixel. This height image can then easily be thresholded, so that pixels above a desired height are detected. An example of a height image is shown in figures 3.1 and 3.2. The algorithms in the last group focus on detecting the same raised objects by searching for their vertical surfaces, and are therefore similar to the edge detection techniques found in 2D image processing.

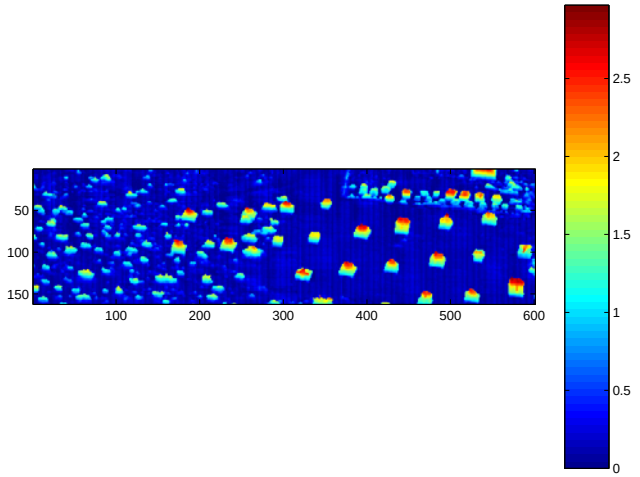

Figure 3.1: An estimated height image.

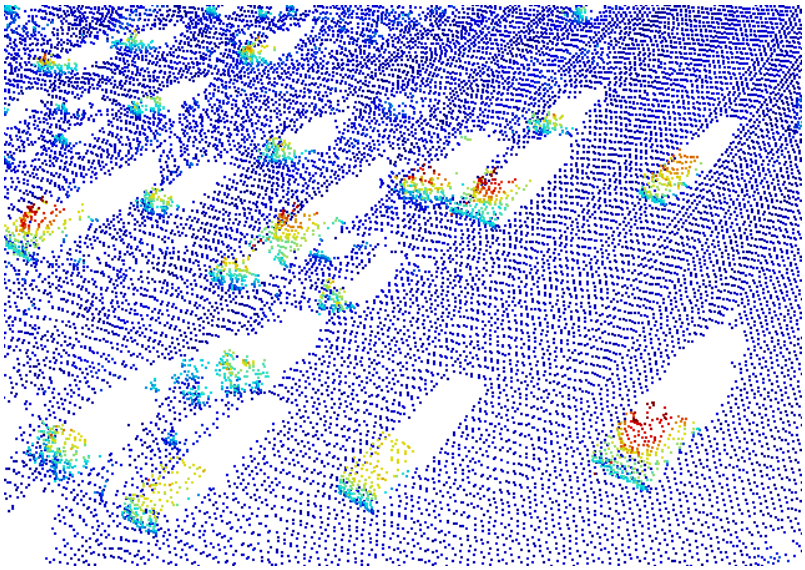

Figure 3.2: Arbitrary angle view of part of scene shown in figure 3.1. The color table is identical to the color table in figure 3.1.

All our algorithms process the 3D ladar images in a 2D view, either in the sensor perspective, or in a top-down projection. Since connected segments in a projection may not be connected in the 3D point cloud, the segmentation results should be post-processed with a 3D connectivity analysis algorithm, like those discussed in chapter 4.

In the following sections, the algorithms will be described in detail.

3.3 Height estimates from local plane estimates in each pixel

The task is to estimate the height above or below the (local) ground level for each pixel. Prior to any height estimation, the ground level has to be determined. When selecting pixels for the ground level determination, (possible) object pixels have to be detected and discarded from the dataset. Given the ground level, the height can be computed. When the height image is generated, this image is thresholded and postprocessed (in order to fill gaps, etc.). Thus, the algorithm consists of the following steps:

1. Determine pixels which most likely do not belong to the ground.
2. For each pixel, select a dataset for ground level determination, and determine the ground level.
3. For each pixel, compute the height above/below the ground level.
4. Threshold the height image and postprocess.

Each of these steps will be described in detail in the following.

Determination of off-ground pixels

We have considered two approaches.

The first approach is simply to use the bounding box of each object detected by the top-down projection segmentation algorithm presented in [20]. This algorithm detects vertical structures around object borders. However, it does not detect interior object points. Thus, we use the bounding box. The second approach is described with respect to the illustration presented in figure 3.3.

From the differentiation of the range image along the row (i.e. the ordinate axis), we see two characterizing facts: 1) The values are negative on vertical surfaces (i.e. the range decreases when we move upwards in the range image), and 2) it is large positive values for the upper edge of the object (i.e. it is a large increase in range between the upper part of an object and the background "above")[14]. The pixels between the vertical part and the upper edge (i.e. the "object roof") will have values close to the background values[15]. Thus, this second approach constitutes three parts:

[14] This last assumption assumes (!) that the object is *not* located close to a vertical surface.

[15] Given that the upper part of the object is parallel to the ground plane

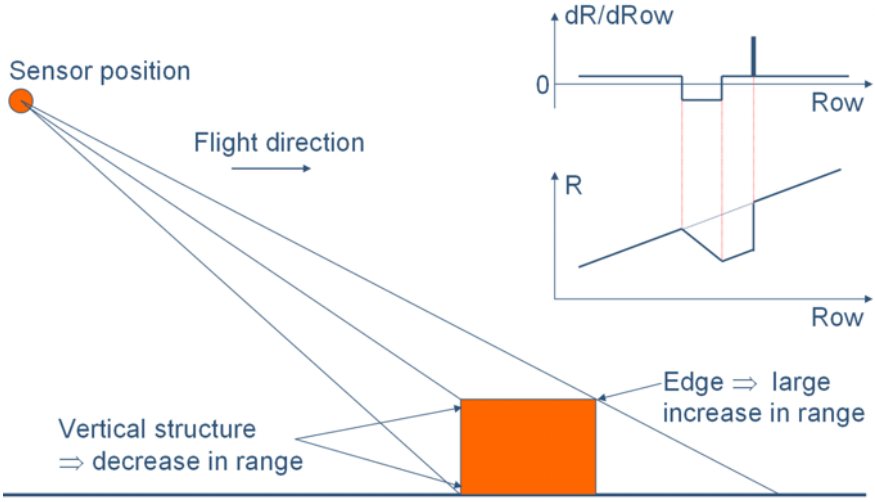

Figure 3.3: Illustration of the range to an object as a function of image row. We observe two characteristic features: 1) It is a large step/jump in range between the upper edge of an object and the background above, and 2) on vertical surfaces the range difference between a pixel and the pixel below is negative.

1. Determine the "leap" pixels (i.e. the upper edge of an object).
2. Determine the pixels which are part of vertical structures.
3. Postprocess (apply logical filters) in order to fill gaps.

The first two parts can easily be determined from the range image by subtracting two consecutive rows; i.e.

$$b_d(i,j) \;=\; b_r(i,j) \;-\; b_r(i+1,j)\,, \tag{3.1}$$

where $\mathbf{B}_r$ denotes the range image. The upper row is defined as the first row. Let us assume that the sensor height is H, and that the range between sensor and object is R. In the upper edge of the object, we will then observe a jump/leap r;

$$r \;=\; \frac{h \cdot R}{H}\,, \tag{3.2}$$

where h denotes the object height. By specifying the minimum interesting object height, h_{min}, the jump pixels are easily found by thresholding the range-differentiation image using the threshold

$$th \;=\; \frac{h_{min} \cdot R}{H}\,. \tag{3.3}$$

Similarly, the "vertical pixels" are determined as the set of pixels in the (vertical) differential image with values less than a threshold ϵ in the range-differentiation image.

As observed from figure 3.3, there may be pixels between the vertical pixels in front and the upper edge in the range-differentiation image which have values similar to the background. These pixels

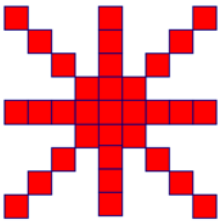

Figure 3.4: An illustration of the spoke filter. The filter is moved over a binary image. If "current" pixel is true*, the output pixel will be assigned* true*. If current pixel is false, the number of spokes which overlaps at least one* true *pixel (0-8) is counted. If this number is larger than a predefined threshold, the output pixel value is assigned* true*. We see that the filter is defined by two parameters; 1) the minimum number of* true *spokes for assigning* true *output, and 2) the filter window size.*

will not be thresholded. Nevertheless, they need to be defined. If not, they will be included in the ground estimation. A Spoke filter, shown in figure 3.4, is applied for this task.

This is a binary filter, and input is an image determined by ORing the "jump pixel image" and the "vertical surface image". The filter is moved over the image. If the ("current") center pixel is *true* (i.e. logical "1"), the output pixel is also *true*. If not, the number of spokes which overlaps with at least one *true* pixel is counted. If this number is above a predefined threshold, the output pixel value will be assigned *true*.

We require that at least 6 spokes shall overlap *true* pixels for assigning a *true* output value. This choice assures that gaps are filled without deforming the shape. In addition to this threshold, the filter is defined by the window size. This window size should reflect the expected object size. An example showing the results using this strategy is shown in figure 3.5.

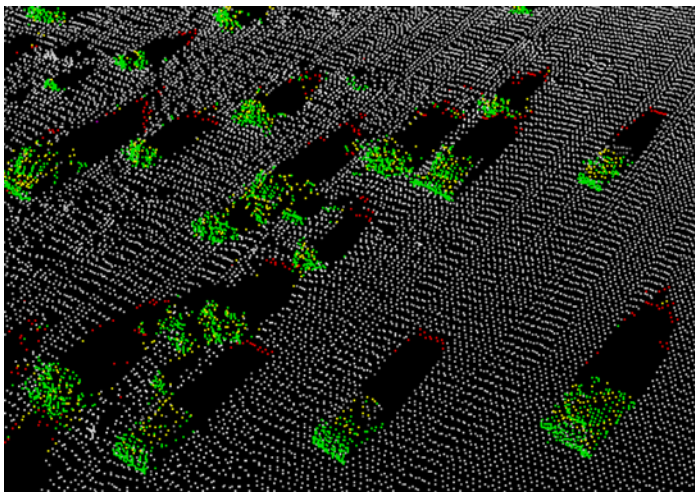

Figure 3.5: Part of a scan. "Jump pixels" (red), "vertical surface pixels" (green), and "gap pixels" (yellow) overlayed the intensity.

Defining a dataset for ground level estimation

For each pixel, a dataset for ground level estimation has to be defined. At least two approaches are reasonable:

1. Use the pixels within a window (in image) which is not defined as possible off-ground pixels.

2. Use the pixels within a predefined *distance* from the "current pixel" which is not defined as possible off-ground pixels.

Defining ground level and height computation

Assume that the ground in each position in a scene can be approximated by a plane. Assume that the height above/below ground level for the position $\mathbf{x} = [x, y, z]^t$ of a given pixel is to be determined. Let $S = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ denote the set of pixels which is available for ground plane estimation. Based on the pixels in S, we want to find the vector $\mathbf{v}$, $\|\mathbf{v}\| = 1$ which minimizes

$$J(\mathbf{v}) \;=\; \sum_{k=1}^{n} \left[(\mathbf{x}_k - \boldsymbol{\mu})^t \mathbf{v} \right]^2, \tag{3.4}$$

where $\boldsymbol{\mu}$ is the mean vector of the vectors in S. $\mathbf{v}$ is assumed to be perpendicular to the local ground plane. We can easily estimate $\mathbf{v}$ from the scatter matrix,

$$\mathbf{C} \;=\; \sum_{k=1}^{n} (\mathbf{x}_k - \boldsymbol{\mu}) (\mathbf{x}_k - \boldsymbol{\mu})^t,$$

because $\mathbf{v}$ is parallel to the eigen vector corresponding to the smallest eigen value. The height estimate, $\hat{h}$ is now given as

$$\hat{h} = (\mathbf{x} - \boldsymbol{\mu})^t \mathbf{v}. \tag{3.5}$$

Thresholding and postprocessing

Pixels corresponding to object heights above 0.5 meter are labeled (i.e. thresholded). Gaps are then filled using the spoke filter in the same way as described in section 3.3, figure 3.4. Finally, all small segments are removed[16].

3.4 Height estimates from morphological operations

Morphology[17] is an important tool in image processing. Very simplified, morphological operations are based on composites of minimum and maximum operations performed inside a neighbourhood

[16] Currently a fixed threshold set to 10 pixels is used. However, in future, this thrshold should be made range dependent.

[17] Morphology is treated in most image processing textbooks, e.g. section 8.4 in [9].

defined by a so called *structure element*. The two primary morphological operations are *erosion* and *dilation*, which correspond to the min and max operations, respectively.

We have seen that the z-image from a ladar point cloud gives us information about the vertical distance between the scene surface and the horisontal plane which passes through the LADARs origin. In our data the z-axis points down towards the ground, which implies that the terrain typically has a larger z-value, while taller objects have a lesser z-value. We are interested in separating the taller objects from the terrain. Obviously, a simple threshold of the z-image cannot be applied as long as the ground in general is not a plane. (And if it had been, it is not guaranteed that the sensor is perpendicular to that plane.) The terrain can instead be a very complex surface, and we therefore need to reference the height of an object to the height of the surrounding terrain. It is in many circumstances reasonable to assume that the terrain covers most of the scene we are looking at, and we know that the objects we are looking for are relatively limited in size. As will be clear from the following subsection, morphological dilation of the z-image followed by morphological erosion, with a structure element large enough to capture at least one terrain pixel around an object, will give us an estimate of the terrain level in the entire image. This operation is called *morphological closing*. When we have an estimate for the terrain level, it is easy to estimate the object height above ground in each pixel, and we see that it is given by

$$\hat{h} \;=\; (z \bullet k) - z \;, \tag{3.6}$$

where $\bullet$ denotes the morphological closing and k is the structure element. The operation in 3.6 is called a *(black) top-hat transform*[18]. Since this method is based on min- and max-operators, outliers have to be removed in order to give reliable height estimates.

We have tested three different strategies for applying the top-hat transform, all of which will be described in the following sections. The first one treats each line in the sensor perspective separately, and performs a top-hat transform line by line. The second one performs a 2D top-hat in sensor perspective. The third approach first transforms the 3D image data to the top-down projection view, and then computes the 2D top-hat.

3.4.1 Line-wise top-hat transform in sensor view

Each line is processed separately. Local changes in height are detected. We will describe the algorithm by using row no. 115 in the ladar image shown in figure 3.6 as an example. The height levels along this row are shown as the blue curve in figure 3.7.

[18] If the changes to be detected are *higher* than the local reference level, the local reference level is determined by an erosion followed by a dilation (i.e. a *morphological opening*). The *(white) top-hat transform* is then calculated by subtracting the ground level image from the original image.

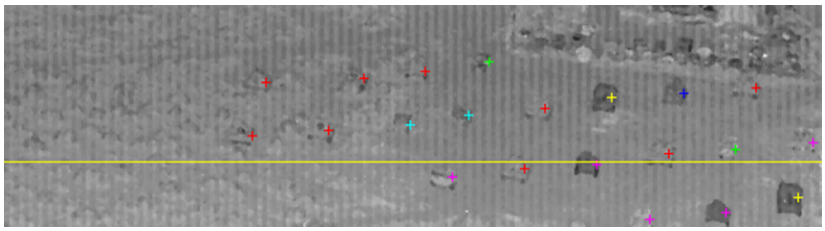

Figure 3.6: Row 115 overlayed the intensity image of a scan.

We clearly see from the z-values (the blue curve in figure 3.7) that the ground level (i.e. reference level) has higher values than the object levels. This is of course obvious because the height difference between sensor and ground is larger than between sensor and object. The morphological closing needed for calculating eq. (3.6) is illustrated by the red and green curves in figure 3.7.

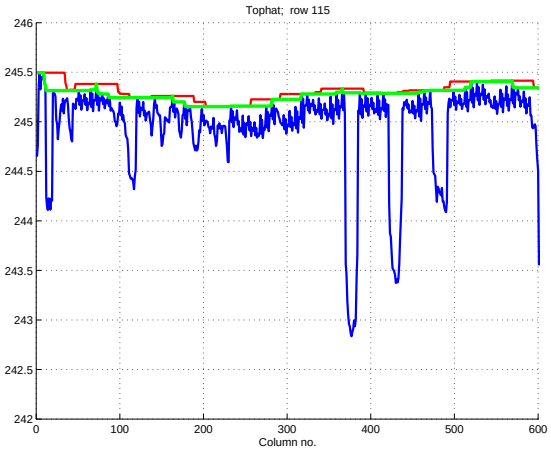

Figure 3.7: Illustration of the morphological top-hat operation. The length of the structure element is 12 meters. See text for details.

In a closing operation, a cube is "pushed" along the curve, and the highest point (value) in each position is determined and stored[19]. The size of the cube has to be determined in advance, and it depends on the size of the objects which are to be detected. In general, it has to be larger than the object size. Otherwise, the cube will fall into the "object pit", and hence the object will be treated as being part of the ground. This operation is the so-called morphological *dilation*, and is in the figure represented by the red curve. Next, the (same) cube is lifted up under the dilated curve and the lowest point of the cube is stored for every position on the dilated curve. This operation is the so-called morphological *erosion*. A dilation followed by an erosion is called a *morphological closing*; it fills ("closes") gaps smaller than the size of the cube. The height estimates for row 115

[19] Pedagogically, instead of pushing a cube, it is perhaps easier to imagine a ball is rolled over the curve, and that for each position on the curve, the highest point/position of the ball is noted. However, we are in the world of computers, and for computers it is easier pushing cubes than rolling balls...

are shown in figure 3.8.

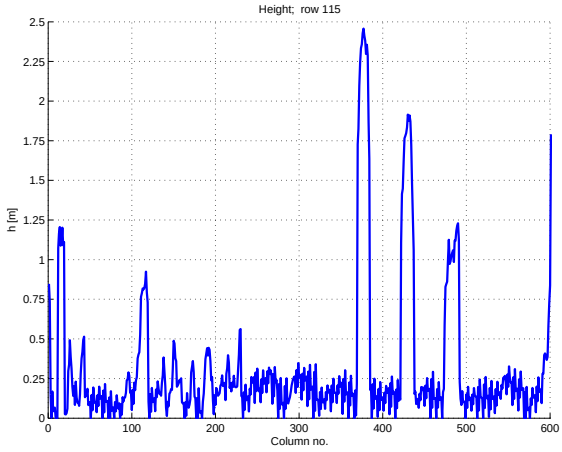

Figure 3.8: Height estimates for row 115 of the scan shown in figure 3.6. The length of the structure element is 12 meter.

3.4.2 Height estimation based on two-dimensional morphological top-hat in sensor view

One disadvantage with the 1D method above is that it ignores information between neighbouring lines. Since the morphological operation is performed along one line at a time, the structure element must be quite wide to ensure that some of the terrain around the interesting objects is covered. This can lead to inaccurate height estimates, since the edges in the terrain are more blurred than is necessary. In addition, since the terrain lines are estimated independently of each other, the estimates can be quite noisy along the image columns.

By constructing a 2D structure element, we are more likely to include a terrain point around an object. We also allow for the fact that the terrain height is correlated along the image columns. Here we have simply performed a 2D morphological top-hat transform directly on the z-image (the z-values in sensor view). The size of the structure element is set according to the sensor resolution at the mean range in the image, but because of the side-view geometry, its footprint (in terrain) will vary as we move it around in the image. This is a considerable drawback with this method.

3.4.3 Height estimation based on morphological top-hat in top-projection view

A better and more complex way to perform a 2D top-hat operation is to apply it in the top-projection view. The structure element will then cover an area given in absolute units (i.e. meters) everywhere in the XY-plane (which we will assume is sufficiently parallel to the ground plane). Although the terrain will influence the exact area covered on the ground, we assume that this contribution is insignificant. By performing the top-hat in this view, we ensure that the real-world distance between pixels is approximately equal everywhere and along both dimensions.

The algorithm is performed in the following steps:

1. Resample the pointcloud onto a uniformly structured grid in the XY-plane.
2. Perform a 2D morphological top-hat operation on the transformed z-image z_{top}.
3. Perform smoothing, using a sliding average filter.
4. Transform the resulting terrain estimate back into its original view.
5. Estimate object height as the difference between the original image and the terrain estimate.

When the new grid has been defined in the XY-plane, the transformed z-image z_{top} is constructed by running through all of the original points and inserting their z-value at the grid point closest to their actual XY-position. If more than one point fall within the same grid point, the tallest object height (in this case lowest z-value) is chosen. Areas in z_{top} that do not cover any points in the pointcloud (e.g because of occlusion) are set to very small values (e.g $-\infty$), so they will not interfere with the estimate.

3.5 Height estimates from filtering operations

We will in this section present two relatively simple ad-hoc strategies. They both determine a ground level through filtering operations, and then compute the object height image.

3.5.1 Height estimation based on a simple two-step averaging technique

An obviously simple way of computing the (local) ground level for a given pixel, is by averaging the z-values (i.e. heights between sensor and ground points) in a small vicinity of the pixel. This is a computationally very simple task, and if it is no objects in this vicinity and the terrain is plane, the ground level will obviously be correctly estimated. However, these assumptions are in general not fulfilled, and in practice a simple averaging is thus useless. Therefore we propose a two-step averaging. The aim of the first averaging step is to identify possible object pixels based on z-image (height image), and the second averaging step is to compute the local ground level based on the z-image and possible objects. The algorithm consists of the following steps:

1. Compute the average z-image (height image).
2. Compute the absolute difference between the height image and the averaged height image. (These two first steps constitute the absolute values of a highpass filtering of the z-image.)
3. Threshold this absolute difference image. All pixels larger than the threshold are considered as possible object pixels. This binary image is used as a mask in the next step.
4. Compute an average z-image based on the pixels (in the z-image) which are not considered to be possible object pixels.
5. Calculate the (object) height image as the difference between the z-image and the two-step averaged z-image (i.e. the output from step 4).

The average z-images (step 1 and step 4) may be computed in two perspectives; either in a sensor perspective or in a top-projection perspective. In the first one the averaging for a given pixel is computed from the pixels in a $w_x \times w_y$ pixel moving window centered in the pixel, and in the latter one it is based on the pixels closer than a distance r from the given pixel. The window size as well as the maximum distance (r) have to reflect the size of our targets. Based on the data available, we have chosen window size of 45×25 pixels, and maximum distance 7 meters.

The threshold (step 3) does not seem to be critical. By trial and error, we have chosen 0.5 meter.

3.5.2 Height estimation based on median filtering

In this very simple algorithm, we assume that most of the points in the image originate from a relatively smooth terrain. By filtering the z-image in the sensor perspective with a large median filter, we obtain a robust but highly smoothed estimate of the terrain. The difference between the z-image and the terrain gives us an estimate of object height.

The filter must be quite large to ensure that the window is dominated by terrain points. We have chosen a fixed window size of 45×25 pixels.

3.6 Height estimates from region growing

Region growing is a region-based segmentation method, where a set of small initial regions are grown according to a *criteria of homogeneity*. The resulting regions in the segmented image must satisfy the criterion, and be "maximal" in the sense that the homogeneity criterion would not be true if we merged a region with any other adjacent region.

We are interested in two types of regions in our ladar images: terrain regions and regions with raised objects. The algorithm presented here finds the terrain regions using region growing, and use these regions to estimate the terrain in the entire image. The object regions are also detected as a secondary result of the growing process. This information is used in the method presented in section 3.7.3.

The algorithm is performed in the following steps:

1. Find high-confidence ground pixels and define them as seed pixels, i.e. initial terrain regions.
2. Transform the image to top-projection view.
3. Perform region growing in the transformed image to find the terrain regions.
4. Construct a terrain estimate from the terrain regions using a median filter.
5. Transform back to sensor perspective view.
6. Fill holes.

The high confidence ground pixels are found in a quite simple way by splitting the original image up in equally large subimages, and locating the lowest pixel in each of these subimages. We have chosen to start with relatively few initial regions, and let the growing process do most of the work. The image is split into 2×3 subimages, which results in 6 initial regions spread out in the image. This simple albeit rather little robust way to initialize the algorithm seems to work well for the data used in this work.

When the image has been transformed to top-projection view, the initial regions are grown. The homogeneity criterion we have used is that a pixel within a region and its highest neighbour cannot have a larger height difference than a predetermined threshold. The threshold we have used is 0.5 meter. The growing process is performed in the following way:

1. The initial regions are labeled as “terrain”, and the rest of the pixels are labeled “unknown”.
2. Each “new” pixel in the terrain region is compared with its “unknown” neighbours to test the homogeneity criterion. Those neighbours that satisfy the criterion are labeled as new members of the terrain region. Neighbours that do not satisfy the criterion are labeled as “raised objects”.
3. Step 2 is continued until there are no “unknown” neighbours to the terrain region.

When the terrain regions have been defined, those regions are extracted, filtered using a 3×3 sliding median filter, and transformed back to image view. To get a terrain estimate for the entire image, we need to fill the holes where raised objects were detected. This is performed by linear interpolation along image rows.

One of the main advantages of this method is that it is invariant to object size and orientation. Large man-made objects like buildings are easily separated from the background. Moreover, the terrain estimate in areas with much terrain is very accurate, even when the terrain is very complex. The method for filling holes under objects is quite crude, however, so height estimates of objects are expected to be inaccurate.

3.7 Detection of vertical surfaces

The algorithms we have discussed so far all focus on estimating the height above the terrain for each pixel, so that tall objects may be detected. The algorithms described in this section focus instead on detecting the vertical surfaces around tall objects. The algorithms assume that the objects of interest are outlined by vertical surfaces, and have much in common with algorithms for extracting edges in 2D image processing. These algorithms possess several advantages; there is no need for ground estimates, and no need for defining any window size and hence no limitations in (predefined) object size. The main disadvantage is identical to that of 2D edge extraction; a properly distinction between object and background is needed (i.e. a properly defined “3D edge”).

We have investigated three different segmentation algorithms based on detection of vertical surfaces:

1. The top-down projection raised-object detection.
2. Object-silhouette detection.
3. Raised-object detection based on terrain growing.

3.7.1 Top-down projection, raised-object detection

One approach to identify (possible) object areas is simply to apply the top-down projection segmentation algorithm [20] as previously described in section 3.1. The ground plane parameters for each possible object area are estimated from the outer boundary pixels of its bounding box (in top-down projection view).

3.7.2 Object-silhouette detection

Another approach is based on first detecting the upper ("jump") pixels, and then determining the "side-boundary pixels" by a growing process. In this growing process, we assume that the objects of interest are fairly "compact", i.e. the boundary pixels in a given row are neighbour pixels to the boundary pixels in the row above/below (which in fact means that U-shaped objects are invalid).

The gradients used for this boundary detection are calculated as

$$b_g(i,j) \;=\; \max\left\{\left|b_r(i+1,j) - b_r(i-1,j)\right|,\; \left|b_r(i,j+1) - b_r(i,j-1)\right|\right\}\;, \qquad (3.7)$$

where $\mathbf{B_r}$ denotes the range image. These gradients are thresholded using eq. 3.3.

The seed pixels for the growing process is identical to the "jump pixels" described in section 3.3. For each jump-pixel segment (e.g. connected set of jump pixels), in the line *below* the jump segment, it is searched for significant gradients (e.g. gradients above threshold) which are neighbours to the segment. These pixels are then included in the segment, and a new search for significant pixels in the line below is performed. This process terminates when no pixel is included.

The bounding box of each (grown) segment[20] is considered as a possible object area. The ground plane parameters for each possible object area are estimated from the outer boundary pixels of its bounding box.

The thresholding and postprocessing of the output pixels are done as described in section 3.3.

Figure 3.9 shows the gradient image of a ladar image[21], figure 3.10 shows the possible object areas.

[20] Actually the bounding box is increased by six rows in the bottom to ensure that it includes the whole object; sometimes we observe that the lower part of the edges are not extracted (examples are seen in figure 3.10).

[21] pass 217, and scan 44

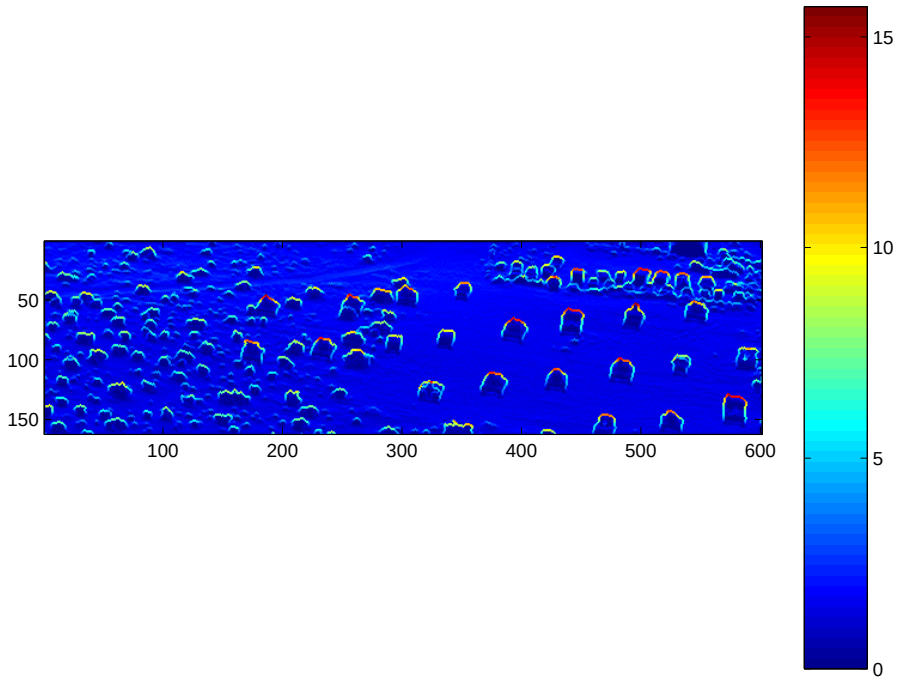

Figure 3.9: Gradient image of scan 44, pass 217. See text for details.

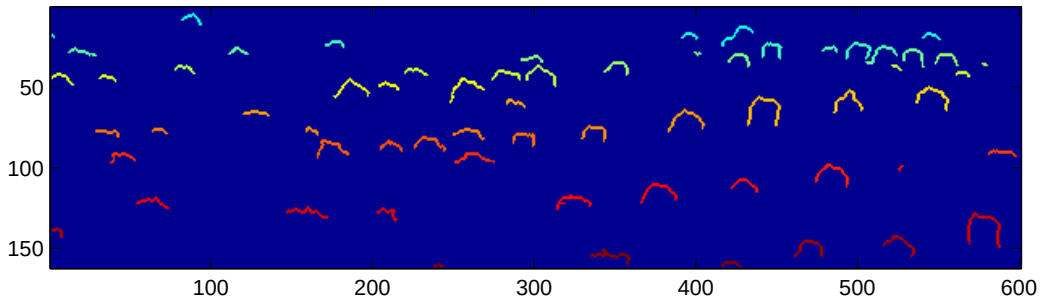

Figure 3.10: Possible object areas of scan 44, pass 217. See text for details.

3.7.3 Raised-object detection based on terrain growing

The idea behind this algorithm is that if we can identify the terrain regions in the image, the remaining regions are probably the raised objects we are interested in. The raised-object regions are thereby identified by *not* being terrain regions. In this simple algorithm, the terrain region is identified by the region-growing algorithm presented in section 3.6.

4 Defining objects

The segmentation process returns pixels which are believed to be object pixels (i.e. non-ground pixels). These pixels have to be grouped and labeled so that each (possible) object has pixels with the same label. These groups will in turn be input to the detection. We have implemented two algorithms for performing this task; a clustering algorithm and an algorithm based on graph theory. They will be briefly described in the following sections.

4.1 Agglomerative clustering for object definition

The clustering algorithm should group pixels (3D-points) in such a way that for each pixel, the Euclidean distance (in 3D) to its nearest pixel *in the same group* is less than a given threshold; i.e. less than a maximum distance. Moreover, the minimum Euclidean distance between a given pixel in a group and pixels *outside this group* should be larger than this threshold. A hierarchical, agglomerative clustering based on the nearest-neighbour distance[22] is able to handle this task. The algorithm is described in pattern-recognition textbooks[23]; it is very simple, and it will thus only briefly be described here.

Input to an agglomerative clustering is a dataset of n samples (here: positions in 3D)[24]. Initially, n clusters are generated, one sample in each cluster. Then the two least separated are determined, and the distance between them measured. If this distance is less than a predefined maximum distance D_0, these clusters are merged, and there are now $n-1$ clusters left. This process continues until the distance between the two least separated clusters are larger than D_0.

Assuming that the range difference in the scene is relatively small compared to the average distance, the minimum-distance threshold D_0 is based on the average distance (in 3D) between two neighbour pixels[25]. This average distance is in turn based on the range between sensor and scene, and the laser's spatial resolution. We have chosen to define it as

$$D_0 \;=\; c \cdot \bar{R} \cdot d_1 \;, \tag{4.1}$$

[22] Also called *single-link* method.
[23] E.g. [12], pp 58ff.
[24] Generally d-dimensional feature vectors.
[25] If this assumption does not hold, D_0 has to be made range dependent.

where $c > 1$ is a constant, $\bar{R}$ is the average distance between sensor and scene, and d_l is the angle of separation between two samples. Based on trial and error (see appendix B for details), we have used $c = 1.5$.

4.2 The graph-based object-definition algorithm

This method is in practice very similar to the clustering method above, but makes use of a graph-based data structure inspired by the Neighbourhood Proximity Graph (NPG) [15].

Let all points in the point cloud be a node in the graph. Then, for each point, construct an edge to all other points that lie within a predetermined distance D_0 from it. We now have a data structure that gives us simple access to the local neighbourhood of all points, which may be practical in many circumstances. Also, since connected nodes in the graph correspond to clusters where the minimum distance between neighbouring points is less than D_0, it is simple to extract such point clusters directly from the graph.

An exhaustive search for neighbours in a N-point cloud demands $N(N-1)/2$ distance calculations, but there are several ways to confine the search by using information from the 2D image projection. We have chosen to perform an initial connectivity analysis in a 2D top-projection image, before performing the graph-based object-definition algorithm. The distance between grid points in this projection is D_0, which is set as described in the previous section.

The object definition algorithm can in summary be performed in the following steps:

1. Perform initial 2D-connectivity analysis in top-down projection.
2. On each 2D-connected component:
   (a) Construct connectivity graph.
   (b) Label each set of connected points.

5 Detection

Detection in our context means to indicate regions in an image where there is reason to believe that an interesting object is located. The detection will *not* be based on the objects' shape. Since we in this context are interested in finding military vehicles, the detection process will in practice be to find objects with a length, width and height within a pre-defined interval, compatible with the vehicle dimensions.

Input to the detection is a set of clusters. The height, length, and width of each cluster has to be determined. The height is simply defined as the height of the highest pixel in the cluster to be

considered. The length and width are calculated by first determining the bounding box (in the XY-plane) with minimum area. The length and width of the cluster are defined as the length and width of this bounding box.

Clusters with height, length, or width outside a predefined interval is discarded from any further investigation. The rest is considered as possible targets. The intervals we have used in our experiments are: Height: 1.5 - 8 meters; Length: 3.0 - 12 meters; Width: 1.5 - 5.5 meters. These intervals may seem to be quite large. However, barrels may be raised, and vehicles may be parked near bushes and small trees. If the interval is to be decreased, more robust (and hence computationally expensive!) methods for estimating height, length, and width have to be applied. Figure 5.1 shows the detection "shoebox".

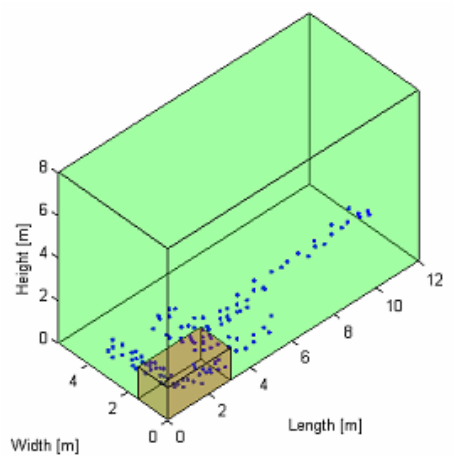

Figure 5.1: The detection shoebox. Objects that fit within the large box, but not the small box are within acceptable limits. An M-110 target is shown for comparison.

Figure 5.2 demonstrates the object detection in a pass. The segmentation is based on the morphological top-hat. The minimum distance threshold in the detection is 1.5 times the average distance between two neighbour pixels. As we see, 44 objects are detected; all 21 vehicles, 8 objects in the scrap yard in the background, and 15 bushes.

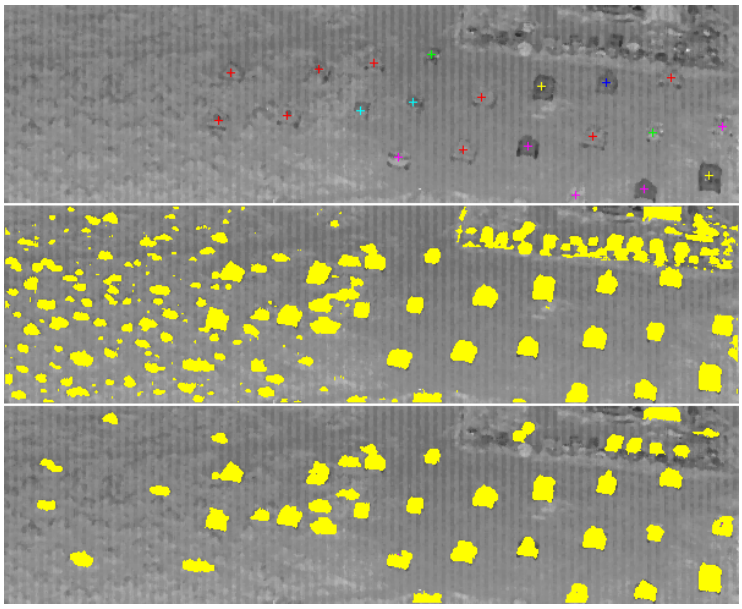

Figure 5.2: Object detection. The upper image shows the ground truth overlayed an intensity image. The middle image shows the results of a morphological based segmentation. The lower image shows the detected objects.

6 Experiments

All methods described in previous chapters are implemented in Matlab. The aim has been performance with respect to detection probability. Thus, the ability in Matlab to run programs written in C has not been utilized.

6.1 Dataset

The dataset we have available is generated as a part of the CMRTR program[26] by using a ladar scanner mounted in a small jet plane, see figure 6.1.

The resolution seems to be around 0.2 mrad, which means 20 cm at a range of 1 km.

The dataset comprises various kinds of vehicles, both military and civilian. Examples of military vehicles are APCs, MBTs, and self-propelled artillery. SUVs and trucks are examples of civilian vehicles. The vehicles are placed in various scenes, i.e.

- in a field (flatland) with some small bushes,
- in a small riverbed,

[26] CMRTR = *Cruise Missile Real Time Retargeting*; it is led by NAVAIR at China Lake, Ca, USA.

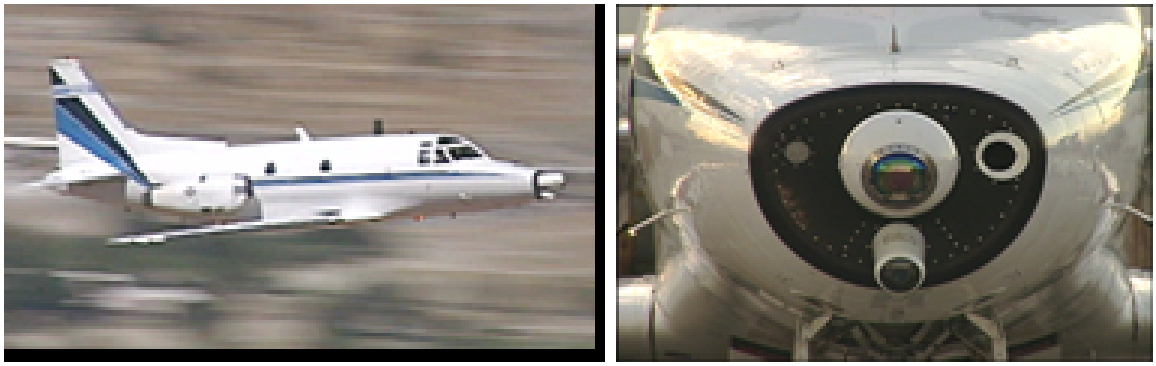

Figure 6.1: Jet plane with ladar scanner.

- in a forest,
- in a camp,
- near to an industrial plant.

There are different types and amounts of clutter in the different scenes. In the field and riverbed scenes, the clutter consists of bushes. Many of these bushes are rather small, but some of them have a physical size similar to small vehicles. Moreover, some vehicles are located quite close to bushes. In the forest scene, there is little “vehicle-sized” clutter. However, there are trees (!) which in parts of the scene hide much of the ground, which in turn may cause the outlier detector (or preprocessing filter) to treat ground pixels as outliers. In the camp and industrial plant scenes, the clutter mostly consists of man-made trash.

We have included scans from all these scenes in our tests. Totally, 36 scans from 27 passes have been picked out. The average distance between sensor and scene is about 1 km. The list of passes and scans along with the average distances is given in appendix A.

Overviews of the various scenes are given in figures 6.2-6.6. In these plots the LADAR intensity is overlayd the 3D point-cloud.

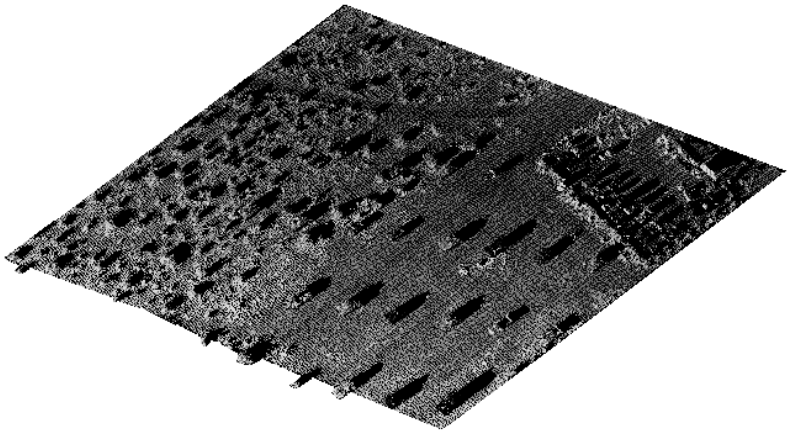

Figure 6.2: An overview of the field.

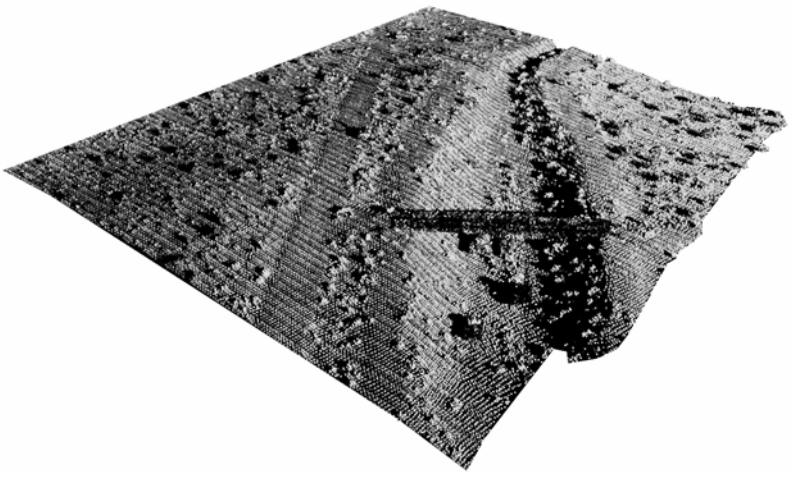

Figure 6.3: An overview of the riverbed.

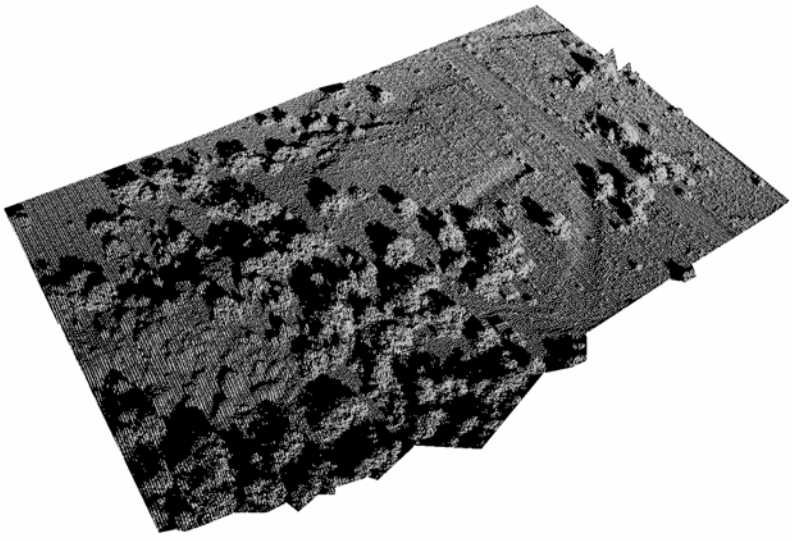

Figure 6.4: An overview of the forest.

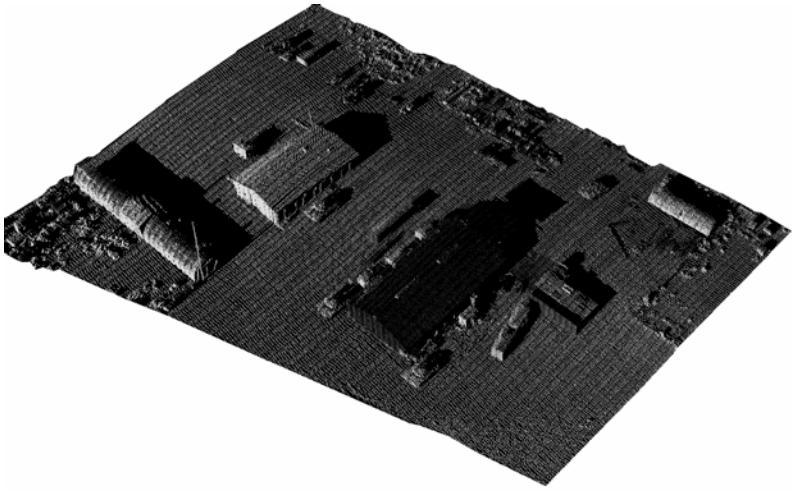

Figure 6.5: An overview of the camp.

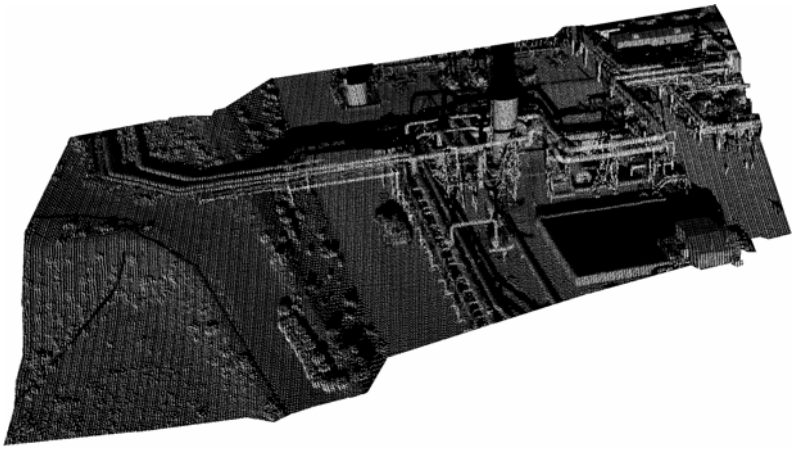

Figure 6.6: An overview of the industrial plant.

6.2 Preprocessing

The experiments concerning the preprocessing which is carried out, may be divided into two parts; one quantitative based on Monte-Carlo simulations, and one mainly qualitative based on real ladar images. Window size has been chosen to 3×3 pixels in all experiments and for all preprocessing strategies but the donut filter[27]. For the outlier detector, a pixel is considered to be an outlier if the difference between unfiltered and filtered values is larger than 20 meter (10 for the simulations)[28].

6.2.1 Simulations

The aim of the Monte-Carlo simulations has been to investigate how well the preprocessing algorithms reduce the image noise, how well they suppress outliers, and how well they preserve thin structures.

Noise reduction is evaluated by repeatingly generating pseudo random noise to the (simulated) image grid, applying the preprocessing, and finally measuring the preprocessing output. The mean absolute difference between the preprocessed output and the mean of the noise distribution is computed and used as evaluation criterion. A number of 1000 replications is used in the simulations. A Gaussian distribution (zero mean and standard deviation σ) contaminated with $\alpha\%$ of another Gaussian distribution (zero mean, standard deviation 5σ) has been chosen for the simulations; i.e. the probability distribution is

$$p(x) \;=\; (1-\alpha)\, N_d(0,\sigma) \;+\; \alpha\, N_d(0,5\sigma)\;, \tag{6.1}$$

where N_d denotes the probability function of a Gaussian distribution. Simulations have been carried out for $0-10\%$ contamination. The results for the various experiments are shown in figure 6.7.

[27] The donut filter demands window size of at least 5×5 pixels.

[28] This difference is only for illustration purposes.

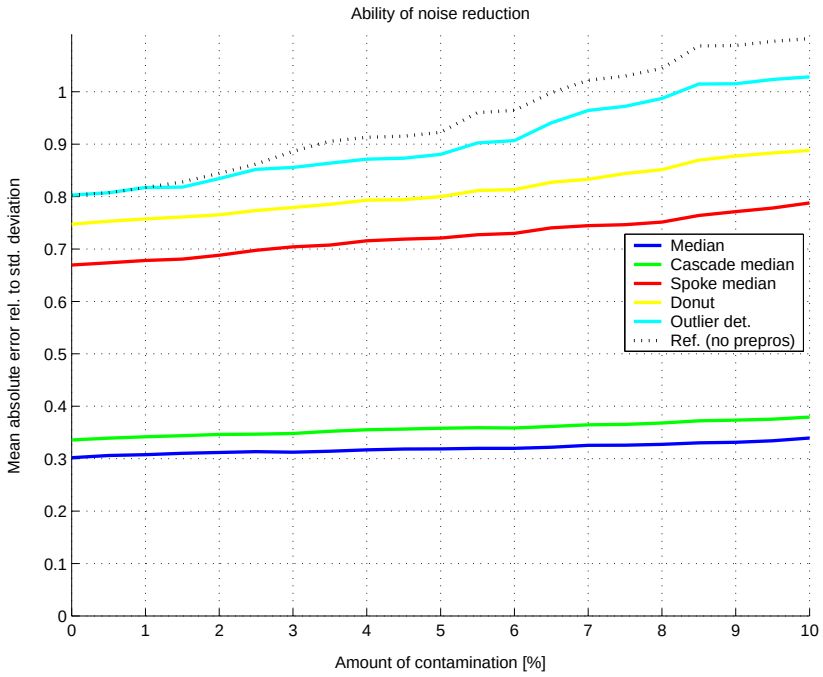

Figure 6.7: Ability of reducing noise.

As we see from the figure, the ordinary median filter and the cascade median filter work very well[29]. The two other filters perform poorly and contribute with little more than waste of computational power.

Suppression of outliers is investigated in the same way as the noise rejection. We assume Gaussian distribution contaminated with outliers. The value of the outliers are $\pm d \cdot \sigma$; σ is the standard deviation, and 50% of the outliers are positive and 50% are negative (we have chosen $d = 20$ meter, a value larger than the outlier consideration parameter (which is 10)). The results are presented in figure 6.8.

The experiment shows that only the ordinary median filter and the cascade median filter are able to suppress outliers. The spoke median and donut filters are virtually worthless. The outlier detector suppresses outliers since it is based on the ordinary median filter.

Finally, we want to simulate the strategies' ability to preserve thin structures. With respect to figure 6.9, pixels in the middle row of the 3×3 grid[30] are taken from a $N(d/\sigma, \sigma)$ distribution, the others are drawn from a $N(0, \sigma)$ distribution. Obviously neither the ordinary nor the cascade 3×3 median filters preserve such thin structures. Nevertheless, we have made this choice in order to investigate the performance of the other strategies. The results are presented in figure 6.10.

[29] This fact is quite clear when we compare them with the averaging filter (which in fact is neither robust nor shape preserving). This filter is optimal for suppressing Gaussian noise. For Gaussian noise, the mean absolute difference is 0.27, and for the ordinary median and the cascade median it is 0.3 and 0.34 respectively.

[30] 5×5 for the donut filter.

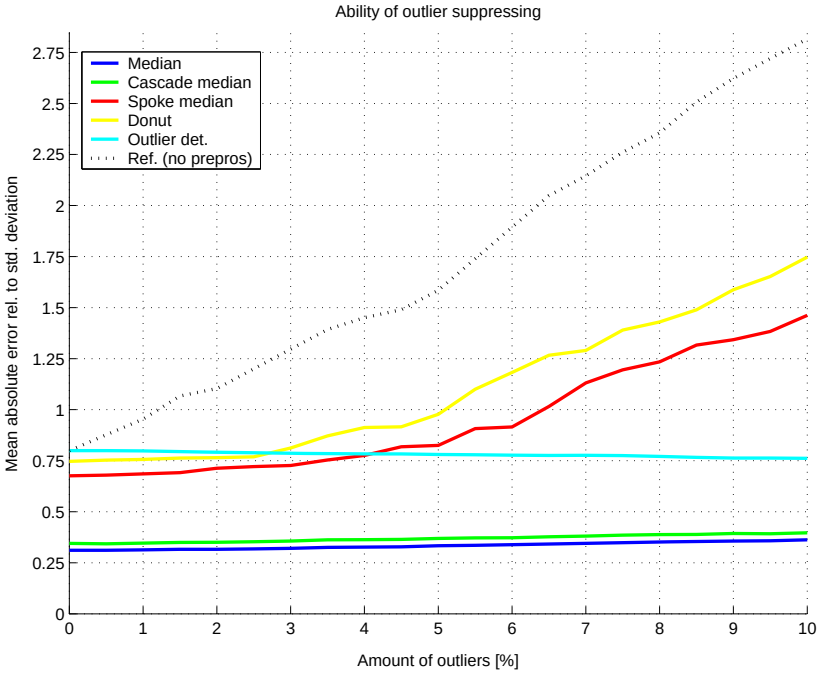

Figure 6.8: Ability of suppressing outliers.

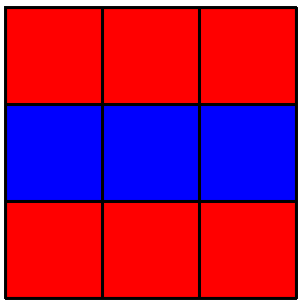

Figure 6.9: Illustration of the grid used in the simulation of the strategies' ability of preserving thin structures. See text for details.

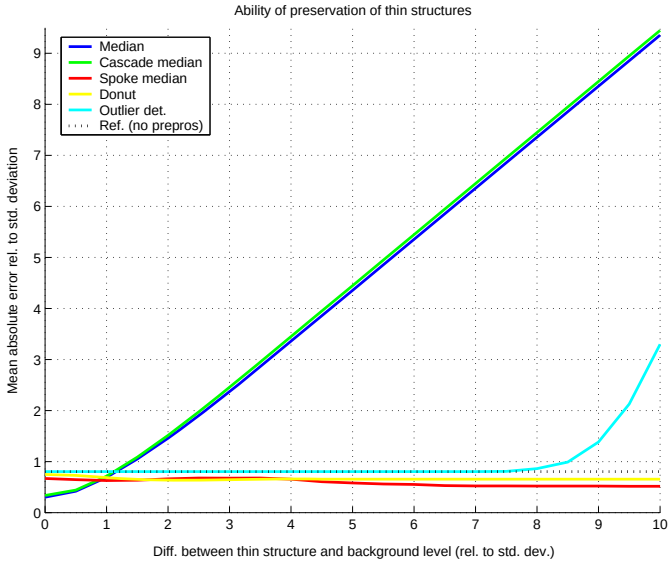

Figure 6.10: Ability of preserving thin structures.

As expected, in figure 6.10 we see that both the ordinary and the cascade 3×3 median filters are unable of preserving such thin structures. The spoke median and the donut filters preserve the structure well. However, the noise reduction is bad for both of them. The outlier detector preserves the structure until the difference (in value) between structure and background becomes so large that the structure is considered to be an outlier.

6.2.2 Results from real ladar data

Part of a camp area has been used for investigating noise reduction. The ground is assumed to be flat in this area. Figure 6.11 shows the border of this part. It is assumed that the XY-plane is horisontal (i.e. the z-image represents the distance between a horisontal plane in the sensor's altitude and the ground). First, the image is preprocessed. Then, ground levels in the evaluation area are estimated. For each sample, the ground level is estimated as the median of all samples closer than 7 meters. Next, the height difference is computed (i.e. the difference in z-direction), and a histogram of the absolute difference is generated. Figure 6.12 shows the results. Table 6.1 presents the mean, median, and maximum absolute deviation.

Scan 44 from pass 217 in the ladar image data base contains several outliers. One part of the scan is presented in figure 6.13, and it shows a rather normal situation. Outliers are not grouped, i.e. an outlier has not any outlier neighbours, which is the situation we most frequently observe. All preprocessing filters are applied, and the results for the z-image (i.e. altitude) are shown in figure 6.13.

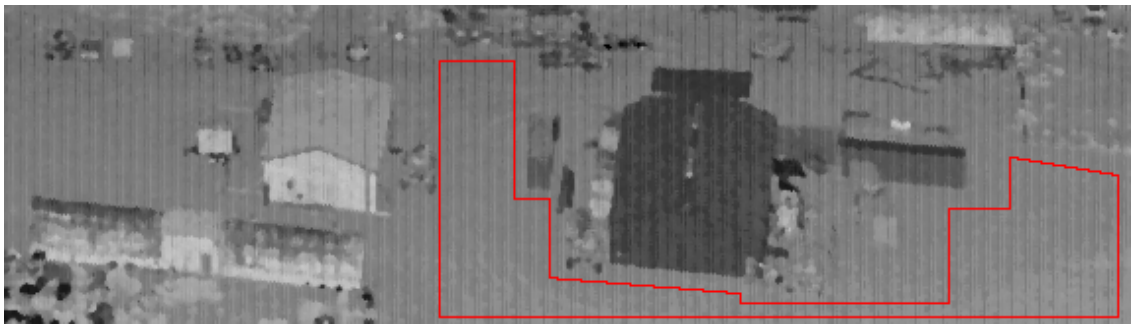

Figure 6.11: Border of the area used for noise reduction evaluation.

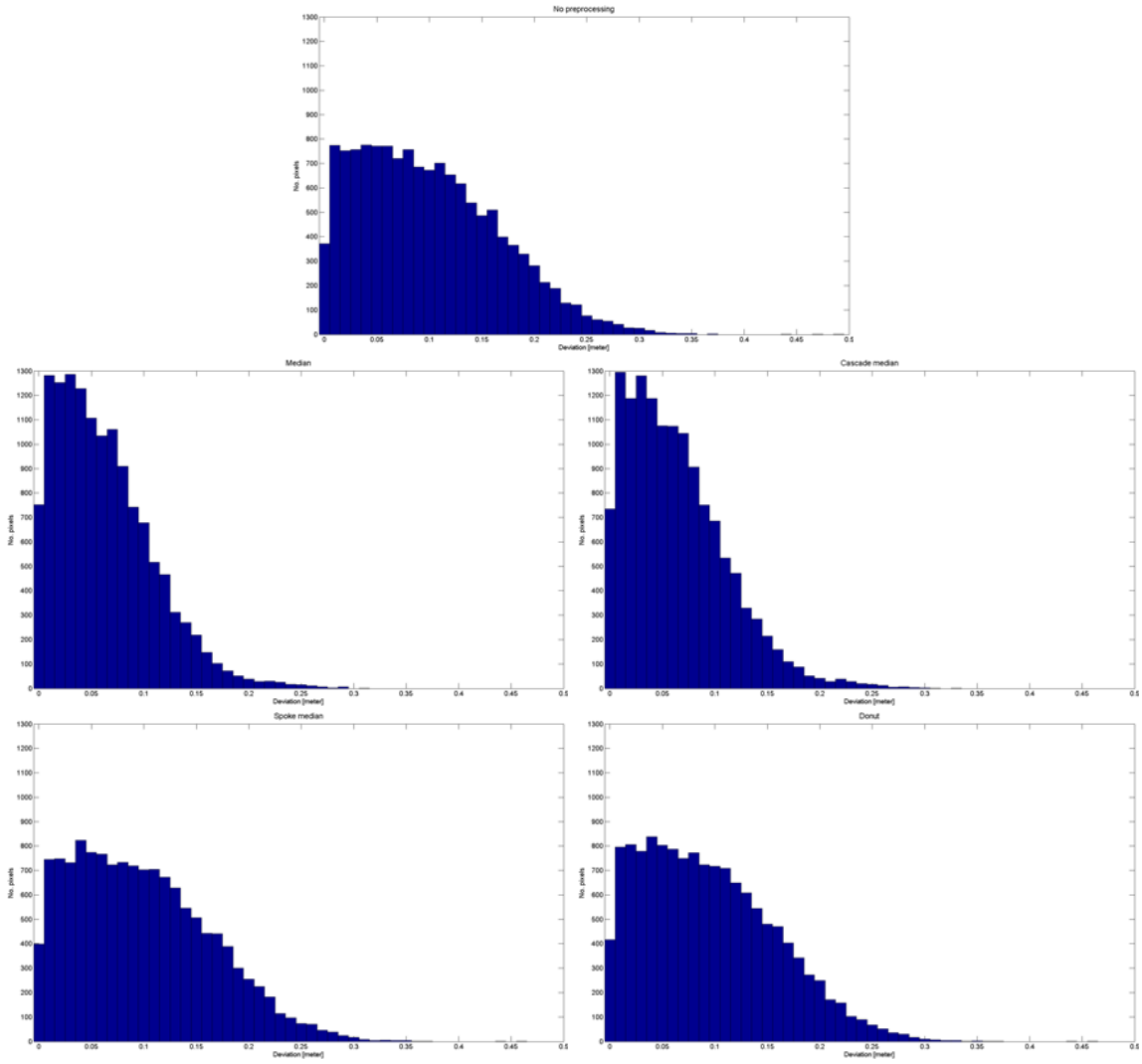

Figure 6.12: Histogram of deviation between (estimated) ground plane and pixels.

Preprocessing	**Mean** [*meter*]	**Median** [*meter*]	**Max** [*meter*]
No prepr.	0.099	0.091	0.491
Median	0.063	0.054	0.308
Cascade median	0.064	0.056	0.326
Spoke median	0.098	0.091	0.459
Donut	0.094	0.086	0.458

Table 6.1: Noise reduction for various preprocessing strategies. Statistics derived from the histogram presented in figure 6.12. Since there is no outliers in the area used for evaluation of noise reduction, the outlier detector gives identical results to that of no preprocessing.

Another part of the same scan contains a challenging part. Four outliers are linked together; one is significantly larger and three significantly lower than surrounding pixels. All preprocessing strategies are applied, and the results for the z-image are shown in figure 6.14.

The largest outlier in figure 6.14 as well as the outliers in figure 6.13 are correctly handled for all strategies. This shows that single pixel outliers does not cause problems for any of the strategies. However, when outliers are grouped, obviously the spoke median and donut filters won't detect (all of) these outliers because they consider them as a thin structure and hence preserve them. But an outlier detector based on either ordinary or cascade median filters will preserve thin structures and suppress outliers as long as difference (in pixel value) between the structure and the surrounding background pixels is smaller than the lowest difference considered to be an outlier.

6.2.3 Preprocessing recommendation

We have seen that the ordinary and the cascade median filters reduce the noise considerably, but they are unable to preserve thin structures. The spoke median filter and the donut filter preserve thin structures. However, they are virtually unable to reduce image noise as well as handle outliers linked together. If preserving thin structures is crucial, other approaches have to be studied.

Based on these observations, we recommend:

- If preservation of thin structures is important: Use an outlier detector based on either the ordinary median filter or the cascade median filter.
- If preservation of thin structures is not of any great importance: Use the ordinary median filter or the cascade median filter. Unfiltered samples may nevertheless be used later in the processing.

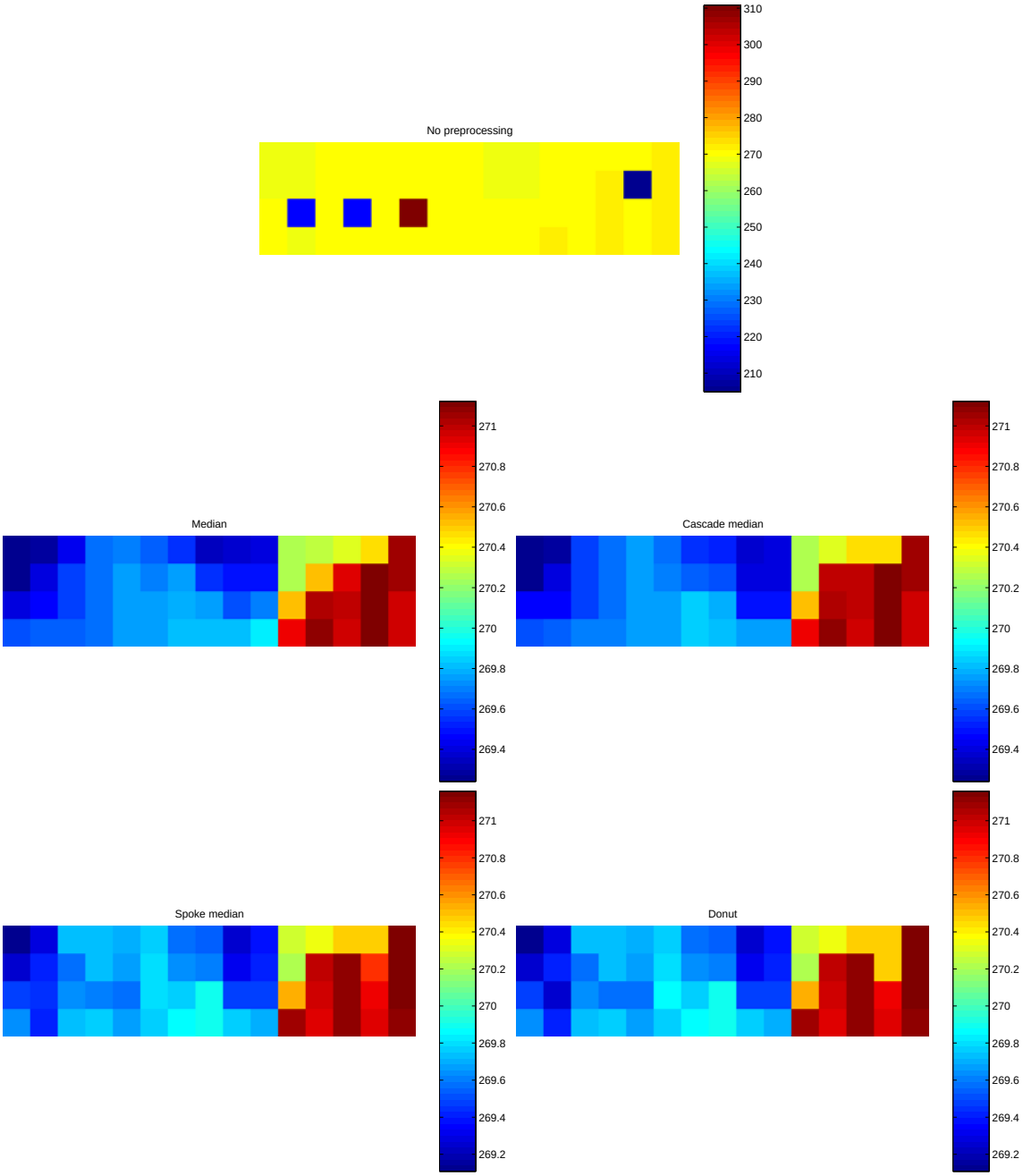

Figure 6.13: Example of outlier rejection of the various preprocessing filters. The subimage is taken from scan 44, pass 217.

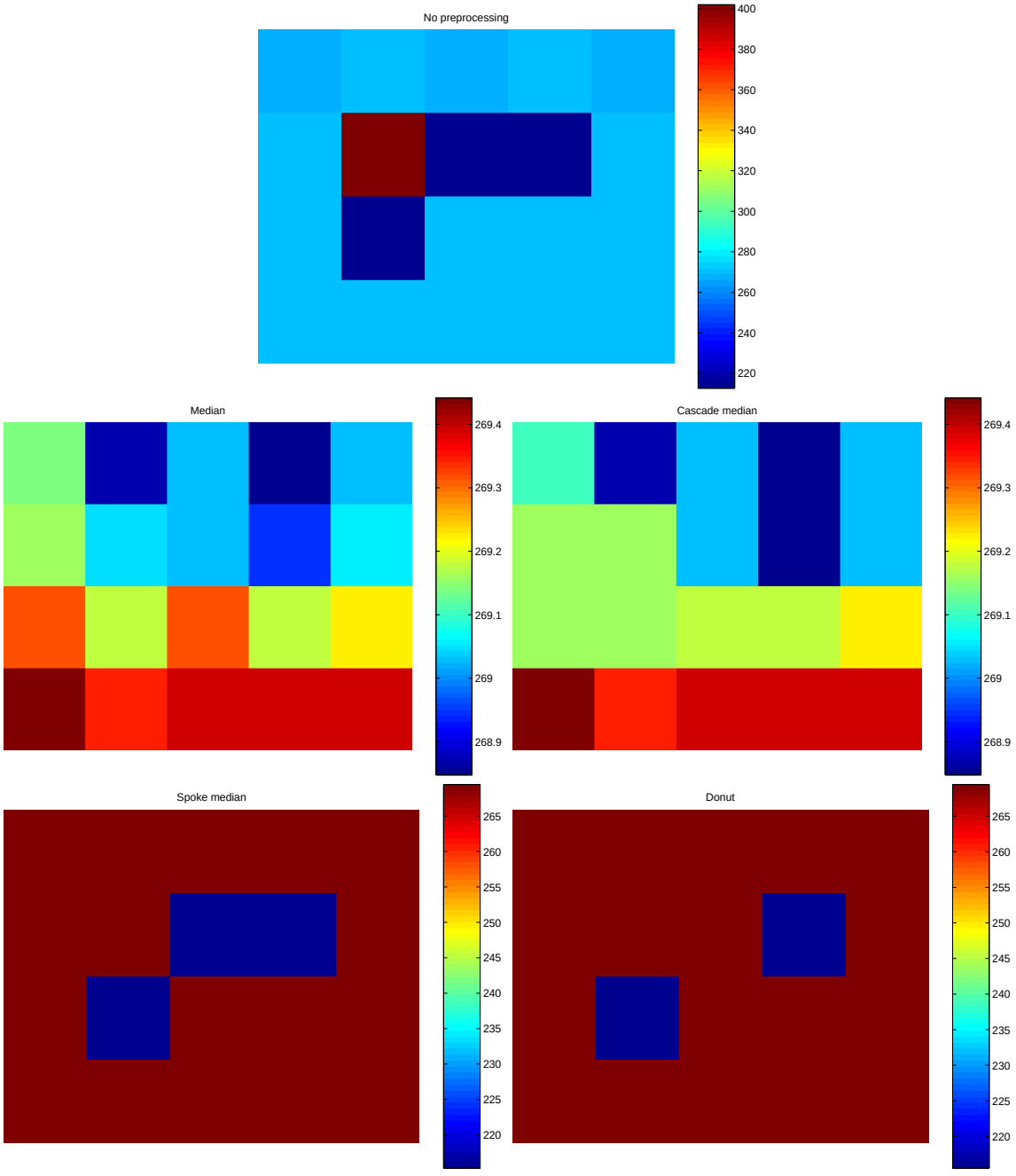

Figure 6.14: Outliers linked together. Outlier rejection of the various preprocessing filters are shown. The subimage is taken from scan 44, pass 217.

6.3 Segmentation and detection

6.3.1 Some introductionary remarks

All methods described in the previous chapter are applied in the tests. Details are given in table 6.2.

The topdown projection segmentation method [20] is used as a reference method. However, no details concerning parameters etc. are given. Based on images in our dataset, we have made the following choices:

- The grid (i.e. pixel) size in the top-down view is 0.5×0.5 meter.
- The threshold used for thresholding the height image (in top-down view) is 0.5 meter; i.e. only object heights above 0.5 meter are considered as possible objects.
- A quadratic structure element, size 3×3 pixels, is used in the dilation.

The minimum distance used in the clustering process is 1.5 times the average distance between two pixels[31].

In the following sections, we will present and discuss results from the processing of scans from each of the different scenes presented in section 6.1.

6.3.2 Performance in scans from the field scene

An overview of the scene is shown in figure 6.2. We have selected 18 passes/scans of a field. Totally, there are 341 objects in these scans. This is a flat area with quite a lot of bushes. Most of the vehicles (75-80%) are located in part(s) of the scene free from bushes. The scans are made from various aspect angles. Some scan examples are given in figure 6.15.

[31] Various distance thresholds have been tested; see appendix B for details.

Method	Details
0	Topdown projection segmentation method as presented in [20]. Pixel size in XY-plane is 0.5×0.5 meter. Size of structure element is 3×3 pixels. This is to be considered as a reference method.
1a	Ground level estimated for each pixel based on pixels in a rectangular window (in sensor perspective), 51×31 pixels, centered around "current" pixel. Off-ground pixels determined by the top-down projection segmentation method.
1b	Ground level estimated for each pixel based on pixels in a rectangular window (in sensor perspective), 51×31 pixels, centered around " current" pixel. Off-ground pixels determination based on finding "jump pixels" and pixels on (near) vertical surfaces. See section 3.3 for details.
1c	Ground level estimated for each pixel based on pixels closer than 7 meter in XY-plane (top-down perspective). Off-ground pixels determined by the top-down projection segmentation method.
1d	Ground level estimated for each pixel based on pixels closer than 7 meter in XY-plane (top-down perspective). Off-ground pixels determination based on finding "jump pixels" and pixels on (near) vertical surfaces. See section 3.3 for details.
2a	Objects determined by line-wise morphological top-hat. Length of structure element is 12 meter.
2b	Ground-level estimation by 2D-morphological top-hat in sensor perspective. The size of the structure element is set to cover about 5×5 meters in the image.
2c	Ground-level estimation by 2D-morphological top-hat in XY-plane (top-down perspective). The size of the structure element is set to cover about 5×5 meters in the image.
3a	Ground-level estimation by a two-step averaging process. Object-height determination for each pixel is based on pixels in a rectangular window (in sensor perspective), 45×25 pixels, centered around "current" pixel.
3b	Ground-level estimation by a two-step averaging process. Object-height determination for each pixel is based on pixels closer than 9 meter in XY-plane (top-down perspective).
3c	Ground-level estimation by a large rectangular median filter (in sensor perspective), 45×25 pixels, centered around "current" pixel.
4	Ground-level estimation using region growing in top-down perspective.
5a	Terrain plane estimated around each possible object area determined by the top-down projection segmentation method.
5b	Terrain plane estimated around each possible object area determined by a region-growing process using the "jump pixels" as seed pixels.
5c	Raised object detection based on terrain growing.

Table 6.2: Overview of the various methods used in the experiments. See section 3 for further details.

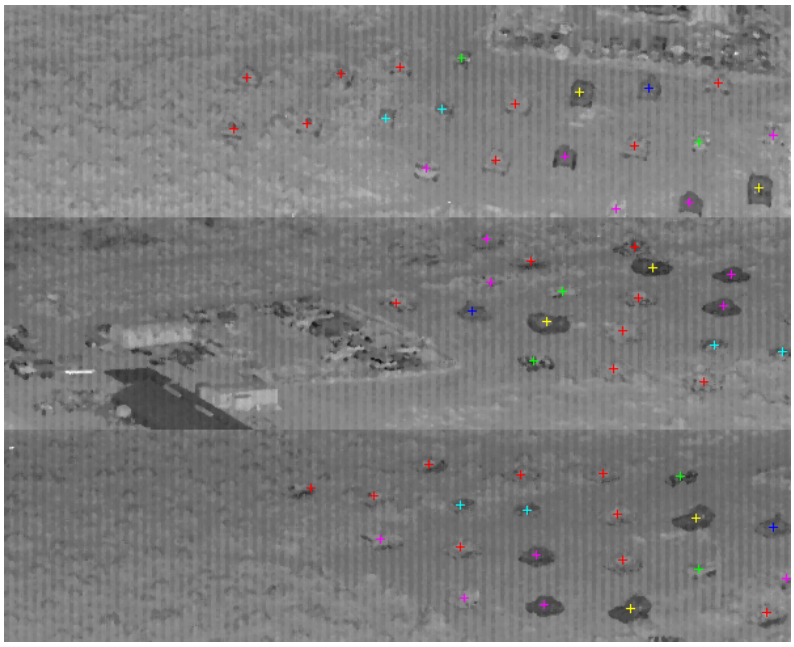

Figure 6.15: Some examples of scans of the field site. Red markers denote M-60, green M-35W, blue ZSU-23, yellow M-53/55, cyan M-113, and magenta M-110A2, M-47, BL, and TR.

Figures 6.16 and 6.17 present the detection rate and the average number of false detections respectively.

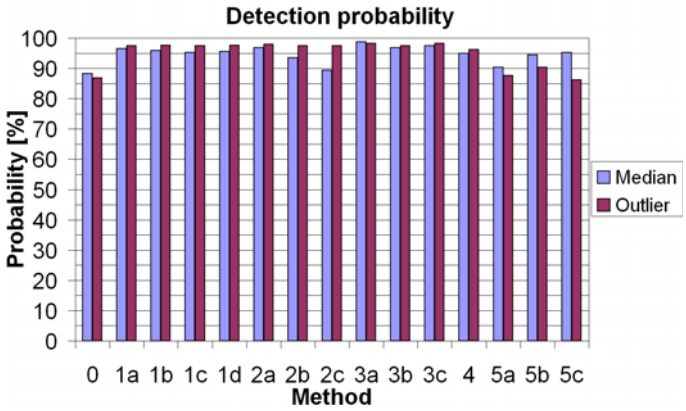

Figure 6.16: Detection probability for the objects in the field scans.

The first finding to notice is the relatively poor detection probability of the reference method. This is due to its morphological dilation which easily groups vehicles and nearby clutter (i.e. bushes). Among the methods presented in this report, methods 5 show the lowest detection probability. This seems to be caused either by a poorly defined area of interest (in which a target is assumed to be located), or due to a too small number of samples for estimating the ground plane. The rest of the methods does not show any great differences in probability detection, which in general is

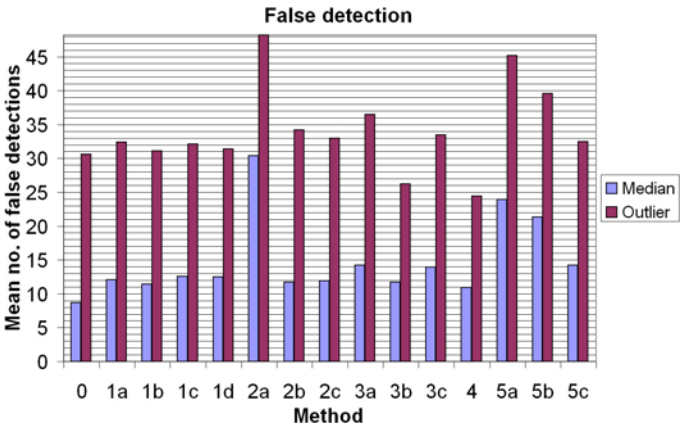

Figure 6.17: Average number of false clusters in the field scans.

very high. The majority of undetected objects (with these methods) stayed undetected because they were located close to the image border. Pixels/samples used for estimation of object height (for a given pixel) have been defined as pixels close to "current" pixel in either pixel coordinates (e.g. method 3a) or terrain coordinates (e.g. method 3b). Both strategies give more or less identical detection probability, which means that the way of defining pixels close to "current" pixels most likely is uncritical. Methods 5 show some lower probability of detection for outlier-removed data than for median-filtered data. This is simply because it is less noise in the median-filtered data, and illustrates the well known fact that edge-based segmentation methods are quite sensitive to noise. For all other methods, the difference between the two preprocessing strategies was marginal. For all but the reference method, the detection probability becomes slightly higher by using the outlier detection preprocessing. The difference is so small that it is difficult to point to some general trend. However, a good guess is that the median filter will blur target objects and nearby clutter together.

Concerning the number of false detections[32], we note that method 2a produces the highest number of false detections. This is due to the use of minimum and maximum operators in the row-by-row morphological operations. Methods 5a and 5b produce unexpectedly higher number of false detections than methods 1 and 3. We have no clear indication of why; likely it could be due to the ground estimation. We observe a much larger number of false detections when using the outlier-removed data than using the median filtered data (in average it is 2.5 times higher). This is not unexpected since the noise level is much higher in the first case than in the latter.

6.3.3 Performance in scans from the riverbed scene

Six scans from three passes have been selected. Totally, there are 29 objects in these scans. An overview of the scene is shown in figure 6.3. As one might see, there are three objects in the bottom

[32] False detection is not a correct description. These detections are, with few exceptions, physical objects like bushes and trees, and man-made objects in a scrap yard outside the field.

of the riverbed, and three on the rim. Examples of intensity images of these scans are shown in figure 6.18.

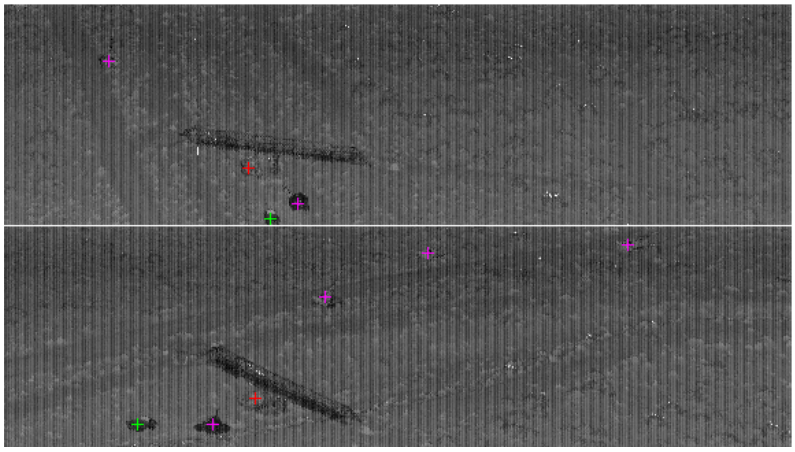

Figure 6.18: Some examples of scans of the riverbed site. Red markers denote M-60, green M-35W, blue ZSU-23, yellow M-53/55, cyan M-113, and magenta M-110A2, M-47, BL, and TR.

As we see, the riverbed sides are relatively steep (especially in sensor perspective, the riverbed side nearest the sensor is so small that it looks more like an edge than a slope), which makes the scene difficult to process. It should also be noticed that there is vegetation in most of the scene. The mean range is also larger than for the other scenes in our dataset. Thus the mean number of false detections should be expected to be higher than for the field.

Plots of the detection performance and mean number of false detections are shown in figures 6.19 and 6.20, respectively.

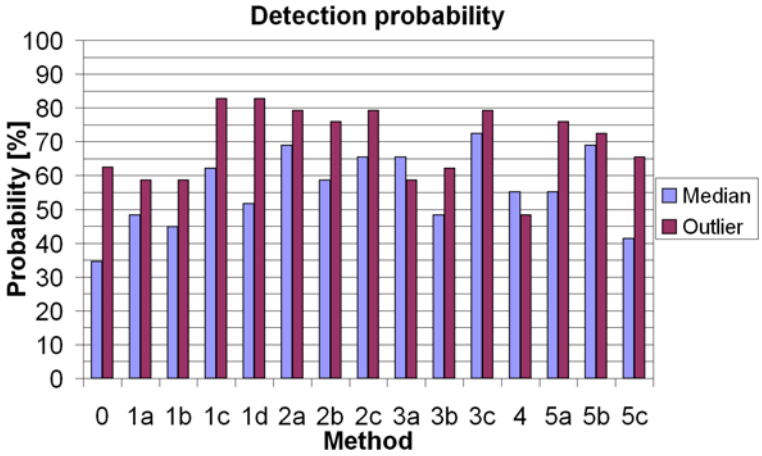

Figure 6.19: Detection probability for the objects in the riverbed scans.

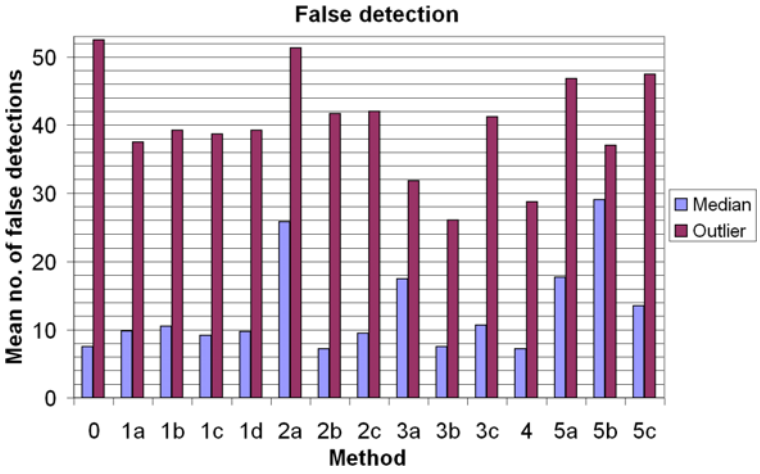

Figure 6.20: Average number of false clusters in the riverbed scans.

The first we notice is that the performance is in general significantly poorer compared to the results from the field. The detection probability is lower, and the mean number of false detections is comparable to the field results for median filter preprocessed data, for outlier removed data, it is higher. (The ratio between false detection in median filtered data and outlier removed data is hence higher (3.8 vs. 2.5 in the field case).) Looking into the results, it seems that it is the objects in the bottom of the riverbed which are difficult to detect. Especially the object below the bridge is only rarely detected. We believe that these findings are partly due to that the range between sensor and objects is being too large (i.e. the object resolution is too low), partly that objects and clutter are clustered, and partly that the riverbed may cause a poor ground plane definition. Concerning the poor performance when using median filter preprocessing, this is probably due to too small objects relative to the filter size (which in fact is only 3×3 pixels).

6.3.4 Performance in scans from the forest scene

Six scans from three passes have been selected. Totally, there are 14 objects in these scans. An overview of the scene is shown in figure 6.4. Examples of intensity images of these scans are shown in figure 6.21.

Plots of the detection performance and mean number of false detections are shown in figures 6.22 and 6.23, respectively.

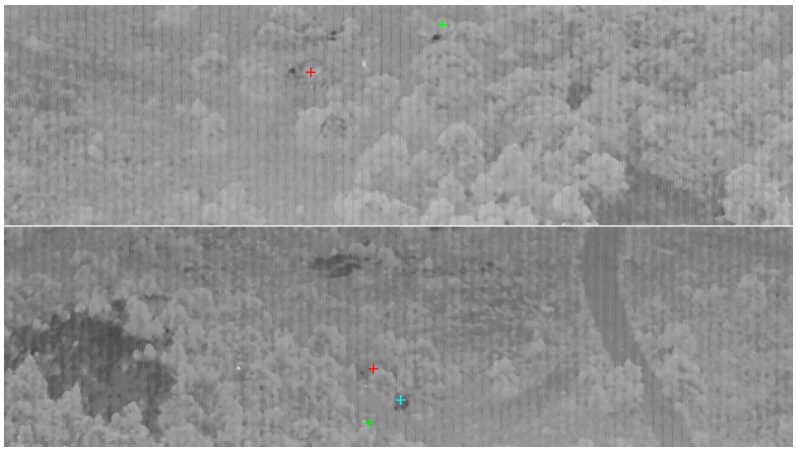

Figure 6.21: Some examples of scans of the forest sites. Red markers denote M-60, green M-35W, blue ZSU-23, yellow M-53/55, cyan M-113, and magenta M-110A2, M-47, BL, and TR.

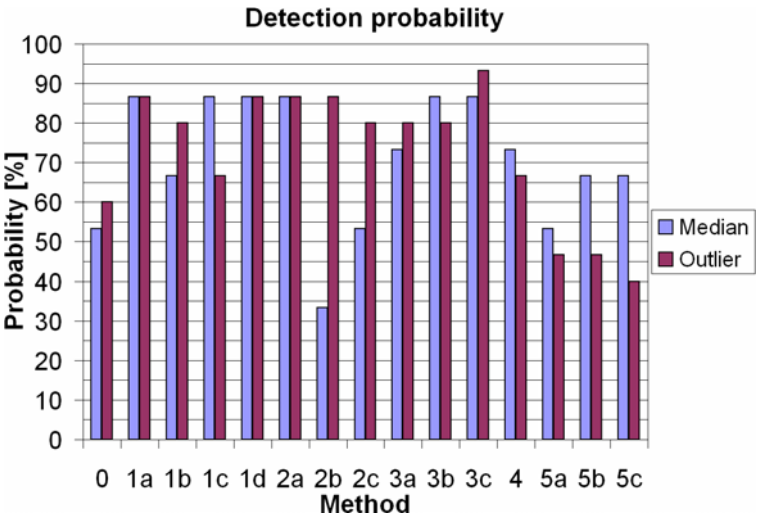

Figure 6.22: Detection probability for the objects in the forest scans.

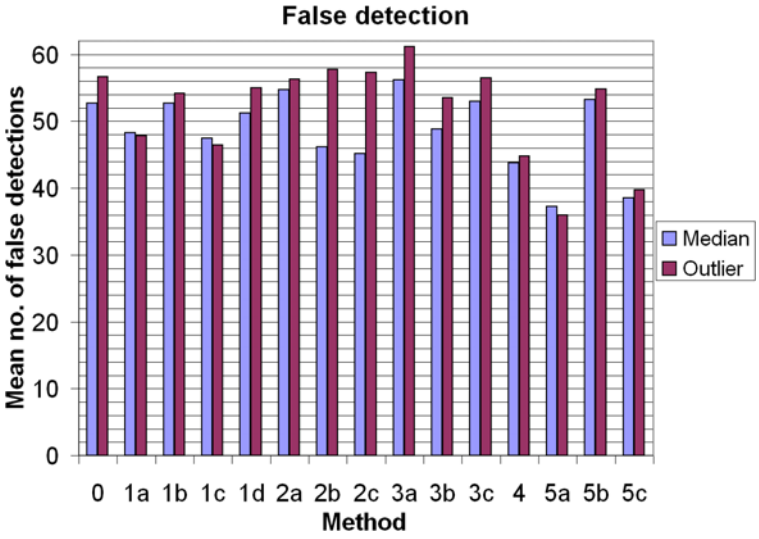

Figure 6.23: Average number of false clusters in the forest scans.

The dataset is small, and it is difficult to draw any conclusions. However, we had expected that it would be difficult to detect any objects in this scene. Thus, all results but those of methods 5 are better than expected. Methods 5a and 5b (both based on edge detection) have problems because the areas of interest are ill defined. As before, the region growing (methods 4 and 5c) perform better on median filtered data. This is also as expected. Region growing depends on little image noise. However, median filtering does also cause trouble. In this scene, the ground pixels are quite sparsely distributed in the forest part of the scene. Hence, many of them are removed by the median filter. This motivates for developing a better preprocessing algorithm.

6.3.5 Performance in scans from the urban scenes

Six scans from three passes have been selected. Totally, there are 17 objects in these scans. An overview of the scenes are shown in figures 6.5 and 6.6. Examples of intensity images of these scans are shown in figure 6.24.

Plots of the detection performance and mean number of false detections are shown in figures 6.25 and 6.26, respectively.

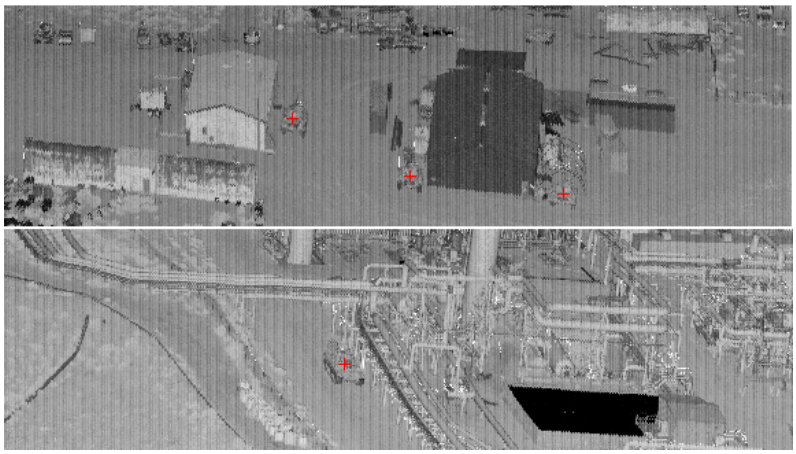

Figure 6.24: Some examples of scans of the urban sites. Red markers denote M-60, green M-35W, blue ZSU-23, yellow M-53/55, cyan M-113, and magenta M-110A2, M-47, BL, and TR.

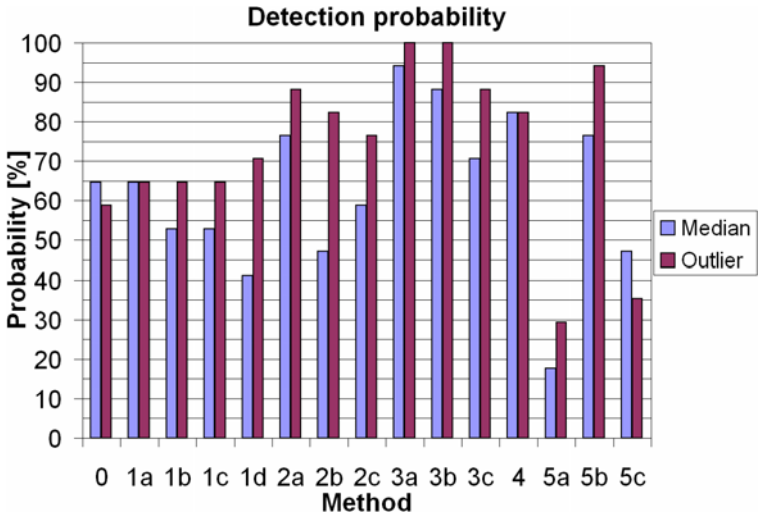

Figure 6.25: Detection probability for the objects in the urban scans.

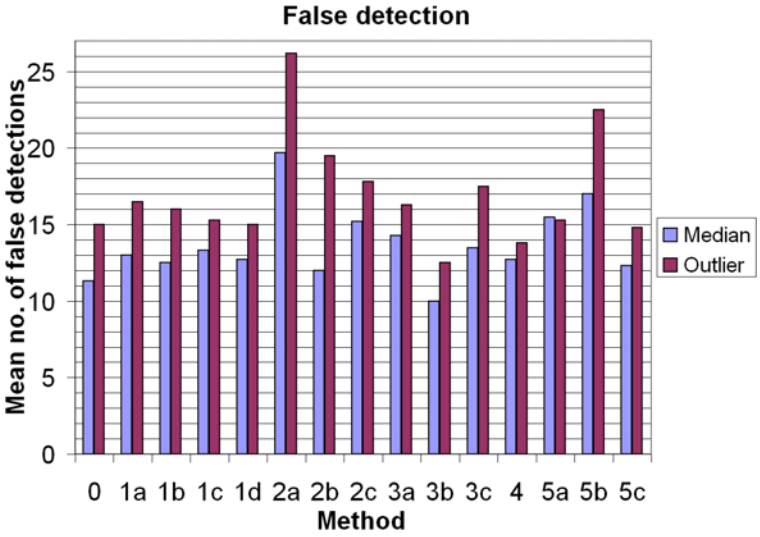

Figure 6.26: Average number of false clusters in the urban scans.

Method 5a shows a virtually worthless performance. Methods 1 are also poor. This is because parts of nearby houses are included in the ground estimate, which in turn becomes wrong. Method 5c also performs badly because it has the tendency to group nearby objects together. The other methods show a good performance, and better than we had expected, even though many of the objects are located quite near to houses and other objects.

6.3.6 Segmentation quality assessment

So far we have only been concerned about whether or not an object is detected. However, it is also important to assess the segmentation quality; i.e. does the extracted segment reflect the object *shape*. We will here only do the assessment qualitatively by presenting some plots of the segmentation[33] of some objects. The reason for doing this assessment is that at a later stage in the processing, each detected object has to be classified[34] into classes like *MBT*[35], *APC*[36], *SUV*, etc. The classification performance (i.e. the probability of correct classification) depends of course heavily on the segmentation quality. Some examples from pass 217, scan 44 are picked out as indicated in figure 6.27. Scatter plots of extraction of the M60 (red bounding box) for some of the segmentation algorithms are shown in figure 6.28[37].

[33] Actually, it is the output of the object-definition process.
[34] Classification will be the topic for a future report.
[35] MBT = Main Battle Tank.
[36] APC = Armored Personnel Carrier.
[37] The segmentation results for M53 and M113 are quite similar, and thus skipped.

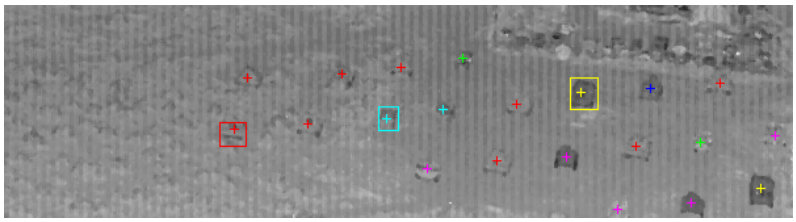

Figure 6.27: Objects chosen for assessment.

The height, length, and width of the object clusters are shown in table 6.3.

Similar results for data preprocessed with the outlier detector are shown in figure 6.29 and table 6.4.

As expected, we have in general found Method 0 (the reference method) to be unable to extract a useful object description. This is both due to (local) object height is being calculated from pixels within a single "top-down pixel" only, and due to the morphological dilation which is needed to link (fragmented) object parts together.

The region-based methods (Methods 1-4) seem to work well in most situations. Using the median filter in the preprocessing causes the object height to be underestimated. This is of course to be expected since the highest parts of the object are small, and thus they will likely be removed by the filter. In addition, the vehicles are surrounded by (tall) grass. This causes the median filter to smear out the (few) *real* ground pixels, which in turn causes the ground level to be estimated higher than it really is. In addition, the ground-level averaging in methods 1 and 3 also contributes to underestimation. We also see that "radial depth" (the object side which is parallel to the "optical axis"), is in general underestimated. This is especially the case for tanks. The turret causes only small parts of the most distant part of the object to be imaged, which is often removed by the filter. Using the outlier detection in the preprocessing causes the height underestimation to be smaller. However, it is in most cases still underestimated due to the grass around the objects. The estimate of the length (here: The "radial depth") is slightly larger for the M-53 and the M-113. This is because more of the most distant pixels of the objects are extracted and included. For the M-60, we observe a small reduction in length estimate when using an outlier detector in the preprocessing. The reason is that pixels from the most distant part of the object are so sparsely extracted that they are not linked together with the other object pixels. We see that the width estimates are slightly larger for outlier-detected preprocessing than for median-filtered preprocessing. In our examples, the increase is small and most likely caused by noise samples close to the objects. In other situations there may be bushes etc. close to an object which have been included. Since the length and width estimators are totally unrobust, these few samples are not ignored in the estimation, and hence the estimates may easily become too large.

The edge-based methods (Methods 5) work best for median-filtered preprocessing. In "clean" scenes it often work well. This is e.g. seen in figure 6.28. In such cases the edges are well de-

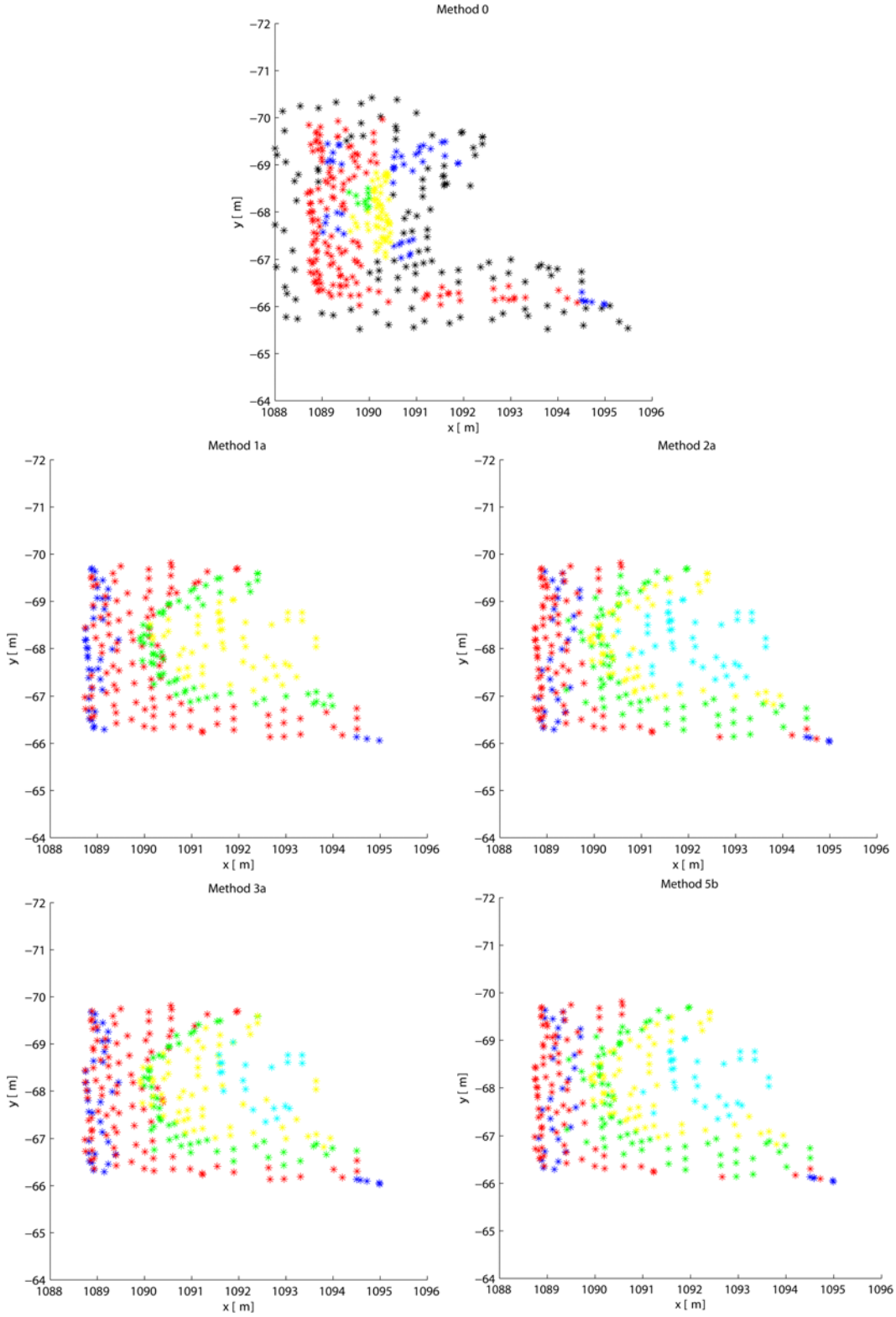

Figure 6.28: Scatter plots of an M-60 for some methods (marked with a red bounding box in figure 6.27). Black samples denote object heights below 0.5 meter, blue in the interval 0.5-1.0 meter, red 1.0-1.5 meter, green 1.5-2.0 meter, yellow 2.0-2.5 meter, cyan 2.5-3.0 meter, and magenta above 3.0 meter. The data is median filtered.

Object	Method	Height	Length	Width
M-60	TRUE	3.21	6.95	3.63
	0	2.25	7.49	4.91
	1a	2.35	6.32	3.61
	2a	2.91	6.36	3.59
	2b	2.41	6.06	3.58
	2c	2.63	6.13	3.54
	3a	2.76	6.36	3.59
	3c	2.55	6.13	3.53
	4	2.51	6.13	3.53
	5b	2.81	6.36	3.59
	5c	2.12	4.65	3.55
M-53/55	TRUE	3.47	7.91	3.58
	0	2.10	7.15	4.43
	1a	2.11	6.56	3.35
	2a	2.66	6.56	3.35
	2b	2.66	3.31	2.66
	2c	2.69	6.55	3.37
	3a	2.68	6.56	3.35
	3c	2.73	6.55	3.37
	4	2.69	6.55	3.37
	5b	2.83	6.56	3.35
	5c	2.33	6.64	3.36
M-113	TRUE	1.85	4.86	2.69
	0	1.72	5.17	3.16
	1a	1.59	4.50	2.39
	2a	1.83	4.50	2.39
	2b	1.66	4.50	2.39
	2c	1.80	4.51	2.39
	3a	1.69	4.50	2.39
	3c	1.76	4.51	2.39
	4	1.66	4.51	2.39
	5b	1.90	4.50	2.39
	5c	1.71	4.52	2.39

Table 6.3: Size of the object clusters. The data is median filtered.

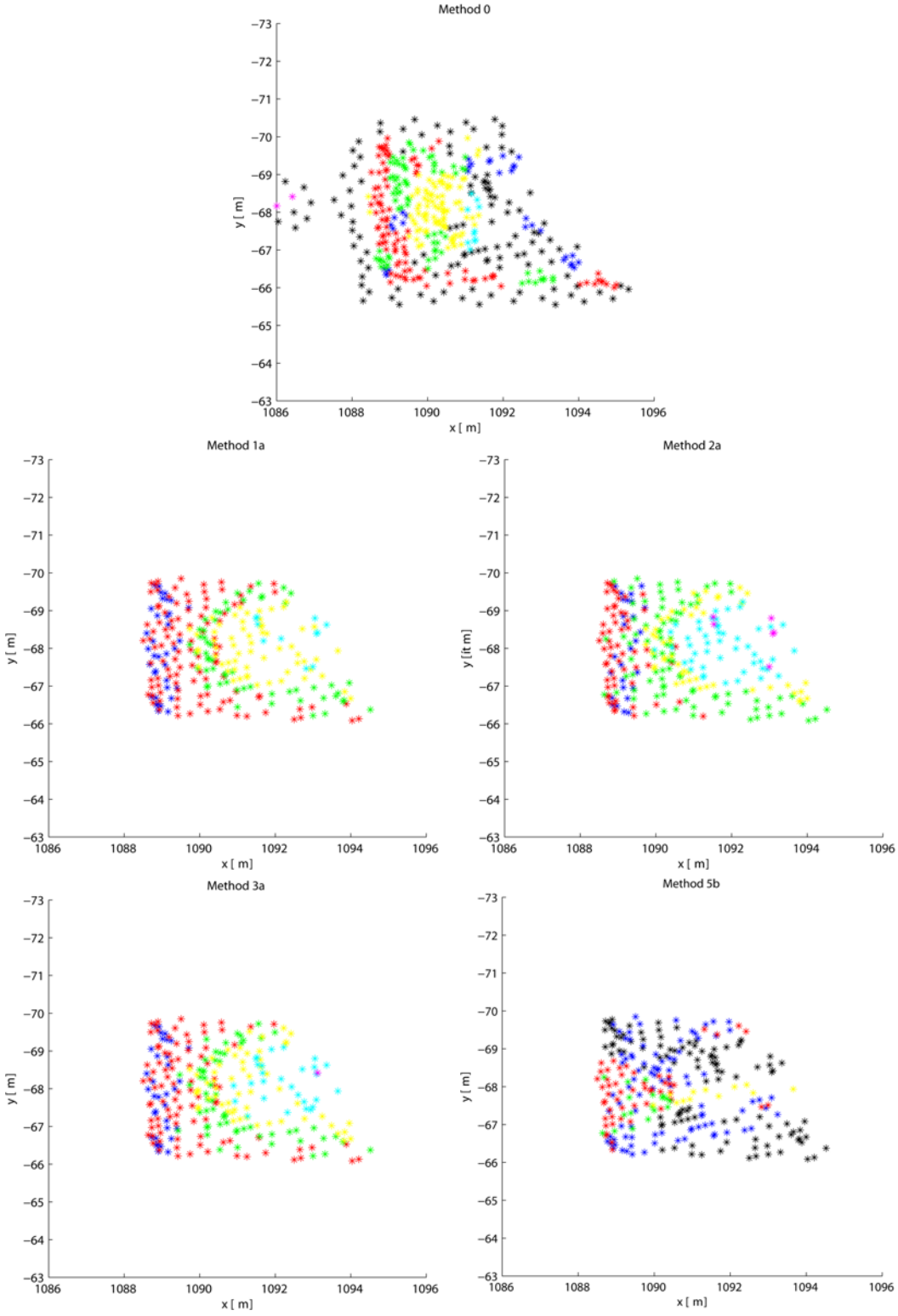

Figure 6.29: Scatter plots of an M-60 for some methods (marked with a red bounding box in figure 6.27). Black samples denote object heights below 0.5 meter, blue in the interval 0.5-1.0 meter, red 1.0-1.5 meter, green 1.5-2.0 meter, yellow 2.0-2.5 meter, cyan 2.5-3.0 meter, and magenta above 3.0 meter. The data is preprocessed with the outlier detector.

Object	Method	Height	Length	Width
M-60	TRUE	3.21	6.95	3.63
	0	3.13	9.66	4.91
	1a	2.88	6.08	3.63
	2a	3.30	6.08	3.63
	2b	3.28	5.87	3.63
	2c	3.10	6.08	3.63
	3a	3.13	6.08	3.63
	3c	3.16	6.08	3.63
	4	3.07	6.08	3.63
	5b	2.42	6.08	3.63
	5c	2.61	6.09	3.64
M-53/55	TRUE	3.47	7.91	3.58
	0	2.14	10.39	4.41
	1a	2.71	6.73	3.73
	2a	2.93	6.73	3.73
	2b	2.96	3.68	3.29
	2c	3.04	6.71	3.75
	3a	2.93	6.73	3.73
	3c	2.96	6.71	3.75
	4	2.91	6.70	3.75
	5b	2.90	6.73	3.73
	5c	2.16	6.24	3.45
M-113	TRUE	1.85	4.86	2.69
	0	2.08	4.76	3.97
	1a	2.11	4.83	2.50
	2a	2.42	4.83	2.53
	2b	2.33	4.83	2.50
	2c	2.33	4.83	2.52
	3a	2.11	4.83	2.50
	3c	2.20	4.83	2.50
	4	2.02	4.83	2.50
	5b	2.42	4.83	2.53
	5c	2.18	4.83	2.52

Table 6.4: Size of the object clusters. The data is preprocessed with the outlier detector.

fined and the “clutter noise” sufficiently low. However, this might not be the situation when the outlier preprocessing is applied. Then the “clutter noise” is higher, and the edges might be more difficult to extract reliably. The consequence is a poorer result as is seen in figure 6.29. As can easily be seen, the result for method 5b is virtually worthless for the classification.

6.3.7 Discussion / some general remarks

Many findings for the particular scenes are already commented. Here we will make some general remarks.

The first to notice is that for most scenes and methods, the detection probability is larger for outlier removed input than for median-filtered input. There are several reasons for this. One situation arises when an object is sufficiently close to another object. Then, the median filter may link the two objects together. Another situation is that the area of the highest part(s) of an object may be small and hence removed by the median filter. If the height of the rest of an object is small (relative to the minimum assumed object height), an object may not be detected. This is especially a problem if a method tends to underestimate the height (as methods 1 do). A third situation arises if the object (in no. of pixels) is small. Then sufficiently large parts of the object may be removed by the median filter so that the estimated size becomes too small (to be considered as a target).

We also notice that the number of false detections is significantly larger for outlier-removed data than for median-filtered data. This does not surprise. The median filter removes/smears out small bushes etc. that otherwise could have been linked together to clusters having physical size similar to a target object.

The region-based methods generally perform well. This is to be expected as long as the size of the height-estimation window is properly adjusted (relative to the size of interesting target objects). The region growing performs in general worse than the rest of the region-based methods. This does not surprise. In many applications, it is experienced that it is difficult for a region-growing algorithm to perform sufficiently robust. This is mostly due to its stopping criteria (for further growing). Due to the stopping criteria, the region growing also works best for median-filtered data.

The edge-based methods have the advantage that they don’t need any information concerning the object size. We have found them to work well as long as the contour is well defined and the amount of surrounding clutter, whose contour could be linked to the object contour, is sufficiently low.

7 Summary and further work

In this report we have presented algorithms for detecting vehicles in LADAR images. Algorithms for preprocessing, segmentation, and detection have been described. The algorithms have been

tested on a dataset kindly provided by US Navy through the NATO SET-077/RTG-45 project *N-dimensional eyesafe imaging ladar*.

We have presented five preprocessing strategies: 1) Median filtering, 2) Two 1D median filters in cascade, 3) Spoke-median filter, 4) Donut filter, 5) Outlier detection and removal. The two first strategies gave almost identical results. They reduced the noise level efficiently, and preserved edges well. However, thin and small structures were removed. The spoke-median and donut filters were found to be virtually worthless. The outlier detector removed outliers, while edges and thin and small structures were preserved. However, the noise level is of course remained unchanged. Our data clearly show that outliers occur. Thus some kind of preprocessing is needed. Based on our experiments, either one of the two first median filters or the outlier detector should be applied. All algorithms have been applied in sensor plane. This may cause problems e.g. in a forest scene where a few ground pixels are surrounded by tree branches, foliage etc. In such situations, ground pixels may be treated as outliers, and thus removed. Therefore, a preprocessing in terrain plane (i.e. XY-plane) should be considered in the future.

We have implemented and tested four groups of region-based segmentation algorithms and one group of edge-based segmentation algorithms. Output from the segmentation is input to an object-definition algorithm. Two strategies are proposed: One conventional agglomerative clustering algorithm, and one graph-based approach. In essence they both give the same results. Clusters with height, width, and length within predefined intervals are assumed to be possible objects. It is difficult to draw any general remarks from our experiments. However, it seems that the best region-based algorithm is better than the best edge-based algorithm. Among the region-based algorithms, those based on morphological or filtering operations give good results in most cases.

However, partly due to all physical target-sized objects being extracted, and partly due to that our detection algorithm is unrobust (i.e. the estimation of the physical size), the number of false detections is large. This fact/problem will be the topic for our future work. We will study more robust detection algorithms, classifiers and/or strategies for discriminating between man-made and non man-made objects, as well as classifiers for discriminating between various man-made objects.

References

[1] Halvor Ajer and Nils Ulrik Andresen. Landbasert, tung ild - rollefordeling, struktur og volum ut fra effektønsker. FFI/RAPPORT 2003/01530, Forsvarets forskningsinstitutt, 2003. in Norwegian.

[2] Walter Armbruster. Model-based object recognition in range imagery. FOM-Bericht 2004/21, Forschungsinstitut für Optronik und Mustererkennung, 2004.

[3] Walter Armbruster. Comparison of deterministic and probabilistic model matching techniques for laser radar target recognition. *Proceedings of SPIE*, 5807:233–240, 2005.

[4] Tomas Chevalier, Pierre Andersson, Christina Grönvall, and Gustav Tolt. Methods for ground target detection and recognition in 3-d laser data. FOI-R 2150-SE, Totalförsvarets forskningsinstitut, 2006.

[5] R. T. Chin and C-L. Yeh. Quantitative evaluation of some edge-preserving noise-smoothing techniques. *Computer Vision, Graphics and Image Processing*, 23:67–71, 1983.

[6] F. W. DePiero and M. M. Trivedi. Real-time range image segmentation using adaptive kernels and kalman filtering. *Proceedings of the 13th International Conference on Pattern Recognition*, C:573–577, 1983.

[7] Chad English, Stephane Ruel, Len Melo, Philip Church, and Jean Maheux. Development of a practical 3d automatic target recognition and pose estimation algorithm. *Proceedings of SPIE*, 5426:112–123, 2004.

[8] Ramon Ll Felip, Sira Ferrandans, Jose Diaz-Caro, and Xavier Binefa. Target detection in ladar data using robust statistics. *Proceedings of SPIE*, 5988-0J:1–11, 2005.

[9] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[10] Paulo F. U. Gotardo, Olga R. P. Bellon, and Luciano Silva. Range image segmentation by surface extraction using an improved robust estimator. *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*, 2003.

[11] Adam Hoover, Jean-Baptiste Gillian, Jiang Xiaoyi, Patrick J. Flynn, Horst Bunke, Dmitry B. Goldgof, Kevin Bowyer, David G. Eggert, Andrew Fitzgibbon, and Robert B. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 18:673–689, 1996.

[12] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[13] Xiaoyi Jiang and Horst Bunke. Fast segmentation of range images into planar regions by scan line grouping. *Machine Vision and Applications*, 7:115–122, 1994.

[14] Kil-Moo Lee, Peter Meer, and Rae-Hong Park. Robust adaptive segmentation of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:200–206, 1998.

[15] James R. Lersch, Brian N. Webb, and Karen F. West. Structural-surface extraction from 3-d laser-radar point clouds. *Proceedings of SPIE*, 5412:345–353, 2004.

[16] Jörg Neulist and Walter Armbruster. Segmentation, classification, and pose estimation of military vehicles in low resolution laser radar images. *Proceedings of SPIE*, 5791:218–225, 2005.

[17] Hans Christian Palm and Knut Ole Hovda. Evaluation of edge preserving smoothing filters. FFI/NOTAT 90/4019, Forsvarets forskningsinstitutt, 1990. in Norwegian.

[18] S Ruel, C. English, L. Melo, A. Berube, D. Aikman, A. Deslauries, P. Church, and J. Maheux. Field testing of an 3d automatic target recognition and pose estimation algorithm. *Proceedings of SPIE*, 5426:102–111, 2004.

[19] Ove Steinvall, Lena Klasen, Christina Grönvall, Ulf Söderman, Simon Ahlberg, Å Persson, Magnus Ehlquist, Håkan Larsson, Dietmar Letalick, Pierre Andersson, Tomas Carlsson, and Markus Henriksson. 3d laser sensing at foi - overview and a system perspective. *Proceedings of SPIE*, 5412:294–309, 2004.

[20] M. R. Stevens, M. Snorrason, D. W. Stouch, and S Amphay. Estimating the ground plane in ladar 3-dimentional imagery for target detection. *Proceedings of SPIE*, 5094:1–9, 2003.

[21] Gustav Tolt, Å Persson, Jonas Langård, and Ulf Söderman. Segmentation and classification of airborne laser scanner data for ground and building detection. *Proceedings of SPIE*, 6214-0C:1–10, 2006.

[22] M. Umasuthan and A. M. Wallace. Outlier removal and discontinuity preserving smoothing of range data. *IEE Proceedings of Vision, Image, and Signal Processing*, 143:191–200, 1996.

[23] Hanzi Wang and David Suter. A model-based range image segmentation algorithm using a novel robust estimator. *Proceedings of 3rd International Workshop on Statistical and Computational Theories on Vision*, pages 1–21, 2003.

[24] M. A. Wani and B. G. Batchelor. Edge-region-based segmentation of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:314–319, 1994.

Appendix A Passes and scans used in the experiments

Pass	Scan	Scene	Range
217	44	field	1114
218	43	field	1249
219	30	field	1230
220	32	field	1046
221	35	field	1143
222	29	field	1161
223	36	field	1281
224	35	field	1078
226	36	field	1069
227	28	field	1109
228	38	field	1304
230	26	field	1220
231	34	field	1143
232	13	field	1099
233	19	field	1225
234	19	field	1281
235	20	field	1209
236	13	field	1062
523	38	urban	1339
523	40	urban	1064
524	32	urban	1318
524	34	urban	1045
526	48	urban	1043
531	36	riverbed	1643
531	39	riverbed	1257
532	28	riverbed	1579
532	29	riverbed	1431
534	28	riverbed	1495
534	29	riverbed	1344
539	44	forest	1225
539	45	forest	1078
539	46	forest	947
540	36	forest	1410
540	37	forest	1251
540	38	forest	1095
567	41	urban	939

Table A.1: Overview of passes and scans used in the experiments.

Appendix B Minimum sample distance between clusters

As mentioned in chapter 4, a cluster consists of samples where the distance between an arbitrary sample and its nearest sample is less than a given threshold. The question to be considered is how this threshold should be defined. If it is too low, object silhouettes will be fragmented; the object pixels are contained in several clusters. Thus, an oversegmentation occurs. If, however, the threshold is too high, nearby objects are contained in the same cluster as the object pixels, which means undersegmentation. Moreover, the threshold (in meter) should be dependent on the distance between sensor and scene (because the angular difference between two neighbour pixels is constant).

Based on these considerations, we have defined the threshold as

$$D_0 \;=\; c \cdot \bar{R} \cdot d_1 \;, \tag{B.1}$$

where $c > 1$ is a constant, $\bar{R}$ is the average distance between sensor and scene, and d_1 is the angular difference between two samples. We have carried out tests for $c = 1.25$, $c = 1.50$ and $c = 2.00$. The angular difference (and beam divergence) of the sensor is assumed to be 0.4 mrad, which implies that the threshold is in the interval 0.5-0.8 m if the average distance to the scene is 1000 meter. In addition, we have also carried out tests using a fixed threshold of 1 meter.

We have picked out 18 scans from the field scene. There are two reasons for selecting scans from this scene; 1) they (totally) contain as much as 341 vehicles, and 2) some of the vehicles are parked close to small bushes. Therefore, this dataset will provide us with sufficient statistics for selecting the threshold definition.

Figure B.1 shows the probability of detecting vehicles, and figure B.2 presents the average number of false objects (bushes etc.) per scan.

We see from the figures that $c = 1.5$ in most cases gives the highest detection rate, and $c = 1.25$ generally gives the lowest average number of false detections. However, the difference in performance between $c = 1.25$ and $c = 1.5$ is far from significant. Thus, we have chosen $c = 1.5$ because it gives a more robust clustering (with respect to fragmentation of objects into several clusters), and at the same time it is sufficiently small so that clustering objects and nearby bushes in most cases is avoided.

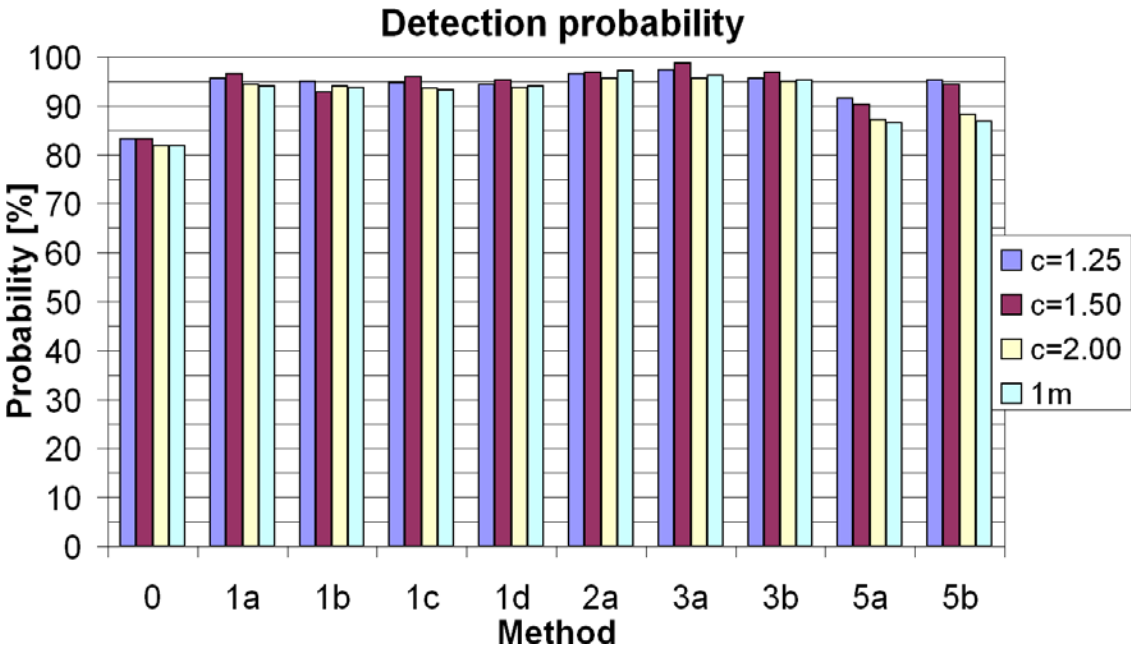

Figure B.1: Detection probability for the various clustering thresholds and some segmentation methods.

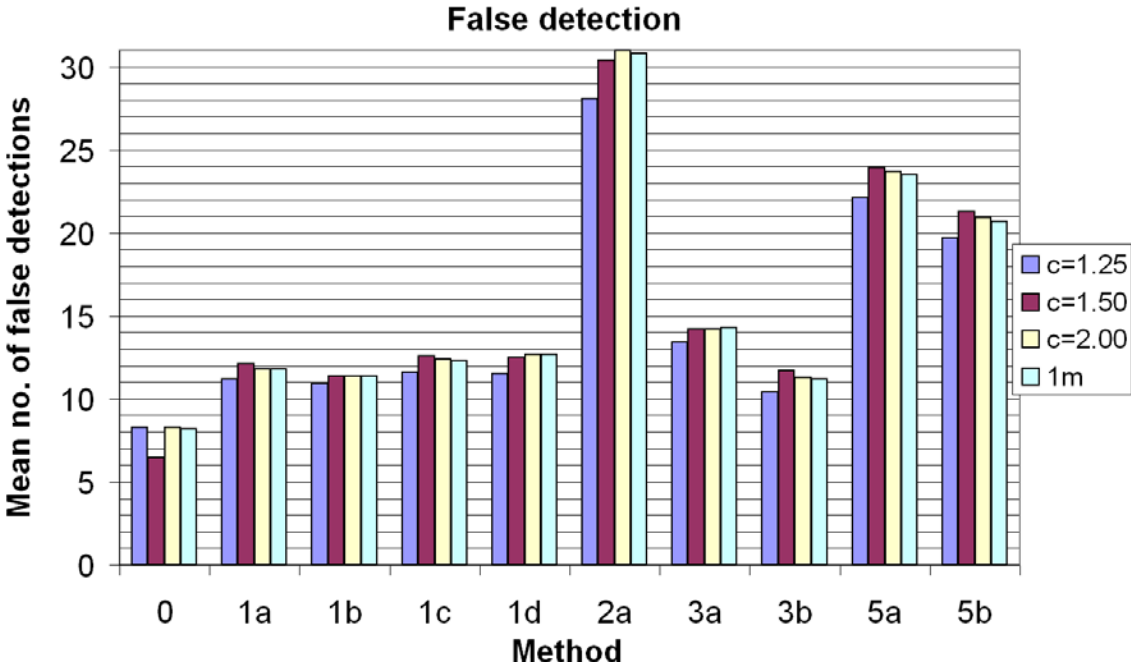

Figure B.2: Average number of false clusters for the various clustering thresholds and some segmentation methods.