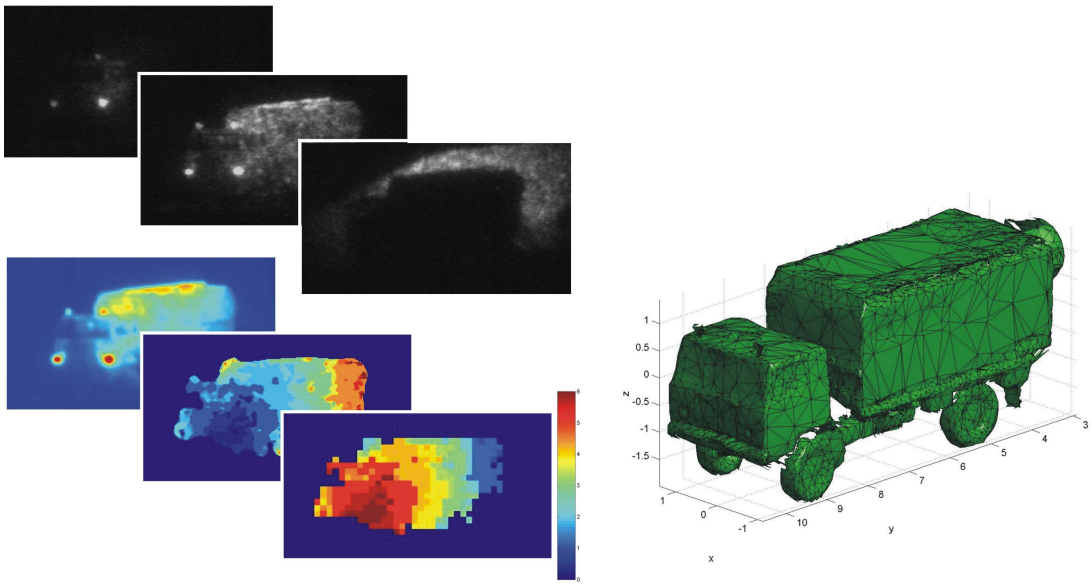
FOI-R--0829--SE
March 2003
ISSN 1650-1942
Scientific Report

Pierre Andersson

Automatic Target Recognition from Laser Radar Data

Applications to Gated Viewing and Airborne 3D Laser Radar



Division of Sensor Technology
SE-581 11 Linköping

SWEDISH DEFENCE RESEARCH AGENCY

Division of Sensor Technology
581 11 Linköping

FOI-R--0829--SE

March 2003

ISSN 1650-1942

Scientific Report

Pierre Andersson

Automatic Target Recognition from Laser Radar Data

Applications to Gated Viewing and Airborne 3D Laser Radar

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency 581 11 Linköping	FOI-R--0829--SE	Scientific Report
	Research area code 4 C4ISR, 5 Combat	
	Month year March 2003	**Project no.** E3036, E2025
	Customers code 5. Commissioned Research	
	Sub area code 42 Reconnaissance and Surveillance 51 Weapons and Protection	

Author/s (editor/s)	Project manager
Pierre Andersson	Lena Klasén, Fredrik Näsström
	Approved by Lena Klasén
	Sponsoring agency Swedish Armed Forces
	Scientifically and technically responsible Ove Steinvall

Report title

Automatic Target Recognition from Laser Radar Data - Applications to Gated Viewing and Airborne 3D Laser Radar

Abstract (not more than 200 words)

A laser-based research system for gated viewing is currently being developed at FOI. One of its possible applications is long-range automatic target recognition. This report treats an implementation of a viable target recognition algorithm based on one-, two- and three-dimensional (1D, 2D and 3D) comparisons of target features. The work starts with a strategy for the construction of three-dimensional objects from gated image sequences, and covers the synthesis of a candidate library. The next step is a course pre-screening test based on one-dimensional projections, followed by a more thorough test based on surface and boundary comparisons. The algorithm is tested on real gated image sequences, and robustness tests are also performed with synthesized objects.

An approach to automatic target recognition from airborne scanning 3D laser radar data is under development, and a brief summary of the main steps of the method is given.

Finally, a method of generating adaptive 3D mesh surface representations from 3D laser radar data is implemented and tested. The implemented method is found to produce acceptable results, but to suffer from computational efficiency problems. However, a constituent surface-matching function is found to have an application to target recognition based on airborne scanning 3D laser radar data.

Keywords

Laser Radar, Gated Viewing, Target Recognition

Further bibliographic information	Language English
ISSN 1650-1942	**Pages** p. 66
	Price acc. to pricelist

Utgivare	Rapportnummer, ISRN	Klassificering
Totalförsvarets Forskningsinstitut - FOI	FOI-R--0829--SE	Vetenskaplig rapport
581 11 Linköping	**Forskningsområde**	
	4 Spaning och ledning, 5 Bekämpning	
	Månad, år	**Projektnummer**
	Mars 2003	E3036, E2025
	Verksamhetsgren	
	5. Uppdragsfinansierad verksamhet	
	Delområde	
	42 Spaningsssensorer 51 VVS med styrda vapen	
Författare/redaktör	**Projektledare**	
Pierre Andersson	Lena Klasén, Fredrik Näsström	
	Godkänd av	
	Lena Klasén	
	Uppdragsgivare/kundbeteckning	
	Försvarsmakten	
	Tekniskt och/eller vetenskapligt ansvarig	
	Ove Steinvall	

Rapportens titel (i översättning)

Automatisk måligenkänning från laserradardata – Tillämpningar på grindad avbildning och luftburen 3D-laserradar

Sammanfattning (högst 200 ord)

Ett laserbaserat försökssystem för grindad avbildning är under utveckling vid FOI. Ett möjligt tillämpningsområde för tekniken är automatisk måligenkänning på långa avstånd. I denna rapport behandlas implementeringen av en möjlig igenkänningsalgoritm, baserad på jämförelser av målegenskaper i en, två och tre dimensioner (1D, 2D och 3D). Arbetet behandlar en strategi för uppbyggnad av tredimensionella objekt från grindade bildsekvenser och även syntes av 3D-objekt till ett kandidatbibliotek. Arbetet omfattar också ett grovsorteringstest baserat på endimensionella projektioner samt noggrannare tester baserade på matchning av avstånds- och silhuettbilder. Algoritmen har testats på autentiska grindade bildsekvenser. Robusthetstester har också utförts med hjälp av syntetiska objekt.

Huvuddragen i en första ansats till automatisk måligenkänning från en luftburen skannande 3D-laserradar summeras också kort.

Slutligen implementeras och testas en metod för anpassning av tredimensionella triangulerade ytor till data från 3D-laserradar. Den implementerade metoden ger acceptabla resultat, men beräkningstiden är lång. En av metodens beståndsdelar är en funktion för ytjämförelser, som visar sig ha ett tillämpningsområde inom måligenkänning från luftburen 3D-laserradar.

Nyckelord

Laserradar, Grindad avbildning, Måligenkänning

Övriga bibliografiska uppgifter	**Språk** Engelska
ISSN 1650-1942	**Antal sidor:** s. 66
Distribution enligt missiv	**Pris: Enligt prislista**

Contents

Chapter 1

Introduction

The aim of this Master of Science thesis project has been to study methods for automatic target recognition (ATR) for different types of laser radar systems. The interest has mainly been directed towards the technique of gated viewing, but applications to other laser radar systems, such as airborne 3D laser radar, have also been considered. The study of target recognition algorithms for this type of data is motivated by current work on the development of a laser-based gated viewing system with long-range imaging capabilities at the Swedish Defence Research Agency, FOI.

Laser radar has a great potential in non-cooperative target observation and recognition. Advantages of laser radar systems include the ability to provide range data, long-range capability, high resolution and vision under conditions where the operating capabilities of passive imaging systems are limited. Laser radar technology has several military as well as civilian applications.

The task of target identification can be performed on different levels of automation. These span over identification assistance by enhanced vision, through completely autonomous recognition, with various levels of operator assisted identification between the extremes.

1.1 Outline & Main Contributions

In this report, chapter 2 provides an introduction to different methods of 3D laser radar sensing. Data from the sensors are used for target recognition, and ATR from gated viewing data is the subject of chapter 3. A method for processing the raw data into three-dimensional objects is presented, and an algorithm for model-based target recognition is implemented and tested on such objects. Chapter 4 contains a description of an implemented method for the construction and reduction of triangular 3D mesh structures, with applications to automatic target recognition from different laser radar systems. A method for ATR from airborne scanning 3D laser radar data is also outlined in this chapter. Finally, chapter 5 summarises the discussions and conclusions of the entire work, and suggests some areas of future work.

The main contributions of this work are found in the data pre-processing algorithm and the application and evaluation of a model-based target recognition method in chapter 3. In chapter 4, contributions have been made with the im-

plementation and evaluation of the mesh construction and reduction algorithm, especially the surface error measuring function.

Chapter 2

Methods of 3D Laser Radar Sensing

Depending on its application, laser radar technology and systems can take on numerous shapes. This chapter gives an overview of three methods of relevance to this work. Some data on the performance of the particular systems used for acquiring data for this work is also presented.

2.1 Gated Viewing

Gated viewing refers to the principle of time-controlling the gain of a passive camera with respect to a pulsed illuminating source, usually a laser. After the emission of each laser pulse, the camera remains inactive during a time corresponding to the light travelling twice the distance to the target. Then, the camera is activated for a short period of time so that the obtained image contains information only about the laser pulse reflections from a certain range interval, in which the entire or part of the target is located. Thus, we get a view of any reflecting objects in a gate that is adjustable in range and width (see figure 2.1), while rejecting reflections and back-scattering from other distances.

The technique of gated viewing can be useful in directly segmenting the tar-

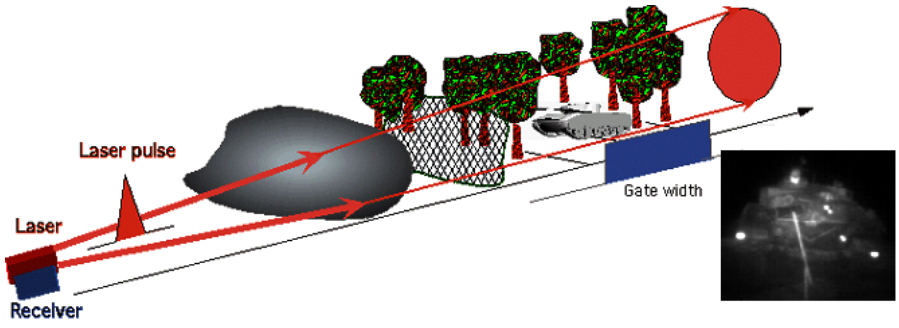


Figure 2.1: The basic principles of gated viewing.

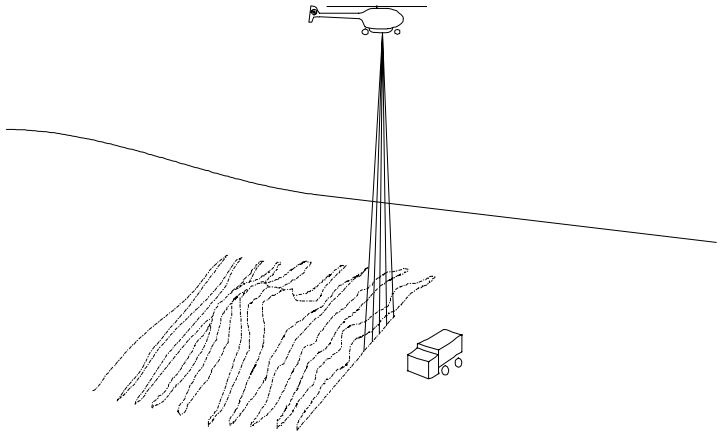


Figure 2.2: Illustration of data acquisition with an airborne scanning 3D laser radar system.

get from its background and foreground. Figure 2.1 also shows the principle for viewing a vehicle whilst sorting out reflections from a camouflage net and atmospheric back-scattering. The application of interest to this work is, however, the one of long-range (∼10 km) target recognition. By taking a series of gated viewing frames, with the gate shifted in range by a fraction of its width in each new frame, information about 3D-structure can be obtained as the gate is moved over the target. This additional information is then used to improve the robustness of the ATR, compared to 2D methods.

The method for creating a range-resolved object from gated viewing data begins by placing the gate at a range where it is just in front of the target, and in each new frame moving it by a suitable amount. As the gate begins to envelop the target, the difference between successive images in the sequence can be used to gain information on the target's three-dimensional appearance. The same applies when the gate gradually leaves the target.

The smallest possible gate displacement with the system that is being developed and tested at FOI is 1 ns. During this time the laser pulse travels 0.3 m, which corresponds to a difference in range of 0.15 m. This is, thus, the range-resolution limit of the system. The minimum gate time is 40 ns, which corresponds to a gate width of 6 m. More detailed information about measurements with and the performance of a gated viewing system can be found in [1].

2.2 Airborne Scanning 3D Laser Radar

While the gated viewing technique obtains 3D information by multiple exposures at different positions of the gate, the scanning 3D laser radar instead measures the range separately for multiple directions. In this work, data from a helicopter-based scanning laser radar system, operated by the Swedish company TopEye AB, has been used. The TopEye system is one of several commercially available systems developed for the purpose of airborne terrain mapping, for ex-

ample described in [2]. Data is acquired with the system carried by a helicopter. As the helicopter moves forward, the laser range-finder beam sweeps from side to side, resulting in a zigzag-shaped scanning pattern on the ground, see figure 2.2. GPS satellite navigation and an inertial navigation system are used for determining the speed, position and orientation of the helicopter. This complementary information is combined with the range data to form a three-dimensional representation of the ground and objects on it in Cartesian coordinates.

The TopEye laser operates at a wavelength of 1.06 μm, and has a laser pulse repetition frequency of 7 kHz. The accuracy in the position of a measured data point is typically 0.1 m in all directions [3], and the width of the covered ground track is 0.35 times the helicopter's altitude above ground level. The TopEye system can be used from altitudes up to 960 m [4].

2.3 3D Focal Plane Arrays

In the two preceding sections, we have briefly reviewed the gated viewing and scanning 3D laser radar techniques. Basically, 3D laser radar constructs 3D objects from sequences of individual 3D measurements, and the gated viewing technique constructs 3D objects from sequences of images. The needs to scan or to shift the gate position, respectively, represent drawbacks to these techniques. In the ideal case, a 3D-resolved object is obtained faster by detecting the reflection of a single laser shot with an array of range-sensitive pixels. This is the concept of the 3D focal plane array, or FPA. The development of 3D FPA technology is ongoing, and may revolutionise [2] laser system capabilities in the near future. When devising methods for automatic target recognition from laser radar data, it is therefore important to keep the concept of the 3D FPA in mind, in order for the method to be applicable to this type of data. For an overview of 3D FPA and other laser radar technologies for the purpose of target recognition, see [2].

2.4 Noise Sources in Laser Radar Sensing

Laser radar sensing is always complicated by the presence of noise. The noise sources include the atmosphere, the target surface, the transmitter and the receiver itself. Direct detection methods, such as gated viewing and the scanning laser radar systems considered in this work, will also be influenced by solar radiation, and require that the laser irradiance dominates the solar irradiance at the target.

The laser transmitter is the first component to introduce noise into the measurement. The irradiance distribution over the cross-section of the beam is not uniform, and the laser pulse energy varies between pulses.

As the laser beam propagates through the atmosphere it is attenuated, and influenced by variations in the refractive index of the air. Such variations are caused by temperature fluctuations arising from atmospheric turbulence, which is generally strongest close to the ground on sunny days. The random intensity fluctuations caused by refraction of the beam in the turbulence cells are called scintillations, which is a well-known phenomenon that, among other things, is responsible for the twinkling of stars at night.

When the coherent light of the laser beam illuminates a coarse surface, the surface will act as an array of spatially incoherent scatterers. In the superposition of the contributions from these scatterers, constructive and destructive interference will occur, creating a spatial modulation of the reflected light profile known as speckle cells.

The total noise is the sum of the contributions of all the different sources. However, at the long distances involved in the gated viewing measurements of chapter 3, we expect the atmospheric turbulence to dominate. We will se examples of the laser radar noise, and find ways of dealing with it by image processing in that chapter.

Chapter 3

Model-Based ATR from Gated Viewing Data

One form of object recognition is the human brain associating visual data with previous experiences, stored in its memory. To reduce the workload of the human operator, or to deal with data that are of a less visual character, we may resort to various degrees of automatic target recognition. The matter of model-based automatic or semi-automatic target recognition is one of taking sensor data in an original or processed form and comparing it, either directly or by means of extracted features, to one or more computer objects and possibly obtaining a match.

Gated viewing data represent a special case of laser radar data. Of course, different approaches can be taken to target recognition from these data. In this Master of Science thesis project, a model-based method has been chosen. Figure 3.1 presents a road-map of the constituent parts of this method. The reader is encouraged to return to this map in the course of reading this chapter, to obtain a clear view of which parts of the method are sensor-dependent, and which ones are not. This chapter will present the method, using the processing of an authentic gated viewing sequence as an example in every step.

As we start the process of ATR by acquiring data, the subject of our laser radar measurements is the *target*. The data from the measurements of the target are then processed into a target *object*. The method for doing this will depend on the type of laser radar system used. In our model-based approach, we also start with a library of *models* that are computer representations of elements of the real world, e.g. vehicles. From the model library, we generate a library of candidate *objects* of the same type as the target object. A *candidate* is defined as a particular model in a specific orientation.

The target recognition is then performed by comparing the target object with all the candidate objects. One candidate object will, in some sense, be more similar to the target object than the other candidate objecs are, and we will conclude that the target has been recognised as this candidate.

The method which is used to perform the object matching in this work, is a slightly modified version of a method described in [5], because the raw data and CAD-models could be converted into objects of a type that this method can handle. This method was also pointed out as an interesting candidate for

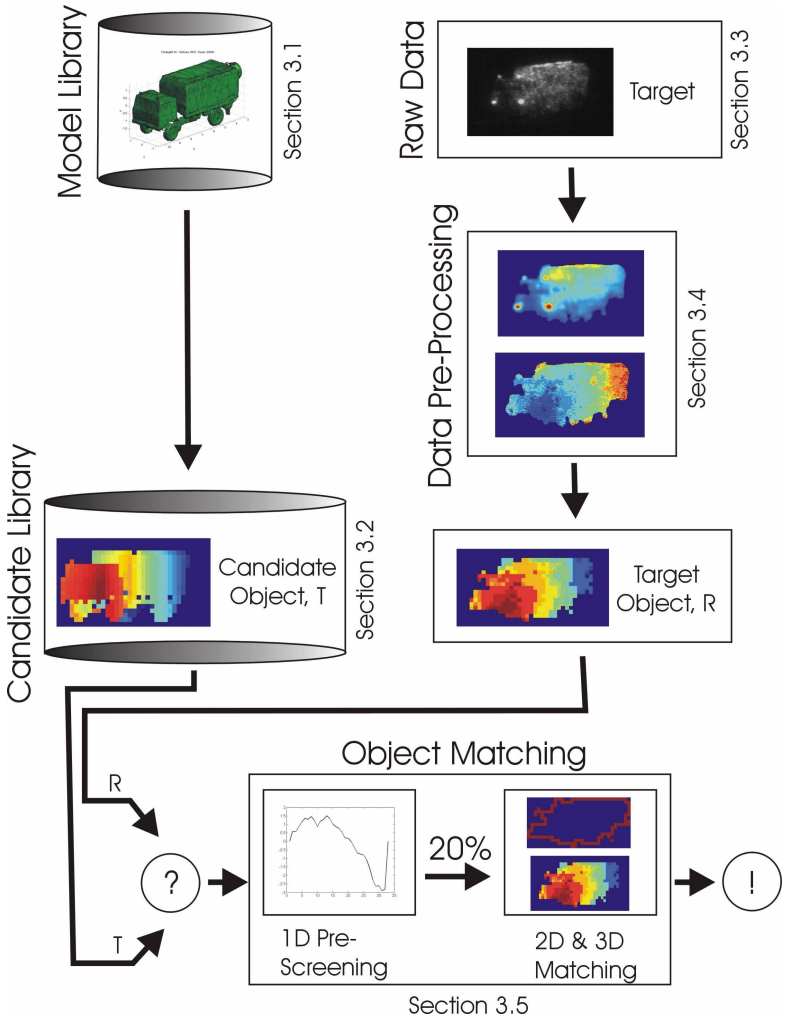


Figure 3.1: Road-map showing the structure of the ATR method of this chapter.

implementation, based on the results from other studies [1][2][6] made at FOI.

In short, the method is a two-stage process, where the large candidate library is first subjected to a pre-screening test that uses one-dimensional features of the objects to sort out the majority of non-likely candidates. The surfaces and boundaries of the remaining candidate objects are then matched with those of the target object in question, and a weighted linear combination of the results forms a matching score for each candidate. The target is recognised as the candidate with the highest score. The slight modifications made to the method of [5] in this work, will be explained in section 3.5, where the entire method is described more thouroughly.

Table 3.1: The model library.

Name	Description	Faces	Size (m^3)
bmp1	Anti-tank gun vehicle	532	6.6×3.0×2.2
bmp3	Anti-tank gun vehicle	558	7.2×3.3×2.2
Centurion	Centurion tank	754	9.6×3.4×2.9
ikv91	Tank destroyer	552	8.8×3.0×2.6
lvkv90	Anti-aircraft gun vehicle	718	5.6×3.1×2.6
mtlb	Armoured troop transport vehicle	300	7.1×3.0×2.4
T72	T72 tank (generic)	966	9.7×3.5×2.5
T72_kvarn	T72 tank (FOI)	16780	9.5×3.8×2.5
tgb11	Terrain vehicle	432	4.4×1.8×2.1
tgb13	Terrain vehicle	1298	5.7×2.5×2.3
tgb30_10000f	Terrain vehicle (FOI)	20000	7.7×2.5×3.4

3.1 The Model Library

In order to obtain a match between a sensed target and a model, we need a library of models to start with. The model library that is used in this work contains eleven CAD-models of vehicles, each represented by a triangular mesh surface with a vertex list and a face list. Some properties of the models are presented in table 3.1. The models in the library differ in origin, and are thus represented with different degrees of resolution and in different orientations in the coordinate system. Information about the 3D rotations required to put them in uniform orientation is stored along with every model. Figure 3.2 displays the eleven CAD-models of the library.

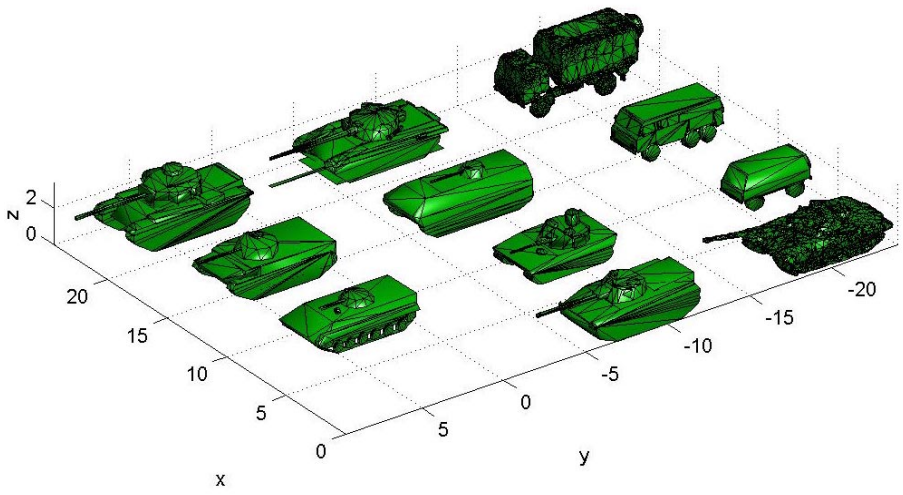


Figure 3.2: The eleven CAD-models of the model library.

3.2 The Candidate Library

The target object that is created from the collected target data (the method for doing this will be explained in section 3.4) is not compared directly with the models in the model library, but with candidate objects, synthesized from the models. Inspired by [5], the model library was used for building a candidate library consisting of one range template T, which we define as the candidate object, and its vertical and horizontal projections for each model and orientation. The candidate library was in this work limited to contain views from an elevation angle of zero degrees only.

In conformity with the method in article [5], a candidate is stored for every azimuth angle around the full 360^o in 5^o increments. This gives 72 range templates and the same number of vertical and horizontal projection vectors per model. An additional two vectors per candidate are also stored, containing the number of pixels used to form each element in the projection vectors. The section on 1D pre-screening explains how these are used for weighting the matching scores. An example of a 3D candidate object and its 1D projections is shown in figure 3.17.

The resolution of a candidate object is 0.2 m per pixel. The candidate objects are created with the aid of a mesh sampling function written by Tomas Carlsson [7], FOI, which samples the range to the CAD-model meshes in a parallel projection from the points of a uniform rectangular grid.

The vertical projection curves (e.g. as in the bottom part of figure 3.17) are formed by taking the mean of the range values in each column across the range templates (top part of figure 3.17), and then extracting the mean of the resulting curve. The horizontal projections are formed analogously by averaging the range values along the rows of the candidate objects.

3.3 Data Acquisition

In October 2002, a series of measurements were made in Älvdalen, Sweden. A TGB30 target was viewed at ranges of 7-14 km. In this work, two gated viewing sequences at a range of 7.2 km were used exclusively. In the first sequence, the target was oriented so that it was seen from the camera location at an azimuth angle of 210^o-220^o, and in the second it was viewed from the side, closer to an angle of 270^o. An azimuth aspect angle of zero degrees is here defined to be when the observer sees the target straight from behind, and the positive direction is clockwise as seen from above. The definitions of the angles of azimuth and elevation are illustrated in figure 3.3. Images from the first sequence are shown in figures 3.4 - 3.6, where the effect of passing the gate through the target can be seen. The original images are 768×576 pixels in size.

It was mentioned in section 2.1 that the gated system is capable of moving the gate in steps of 1 ns and thus, in principle, providing range data with a resolution of 0.15 m. The two image sequences that are used here, however, were collected in a somewhat different way to the one described. Instead of using the smallest possible step of 1 ns between successive frames, the system was set to acquire images at a constant frequency of 10 Hz, while the gate position was manually stepped in 10 ns increments. The width of the gate was 40 ns, or 6 m. This procedure produced sequences consisting of a varying number of

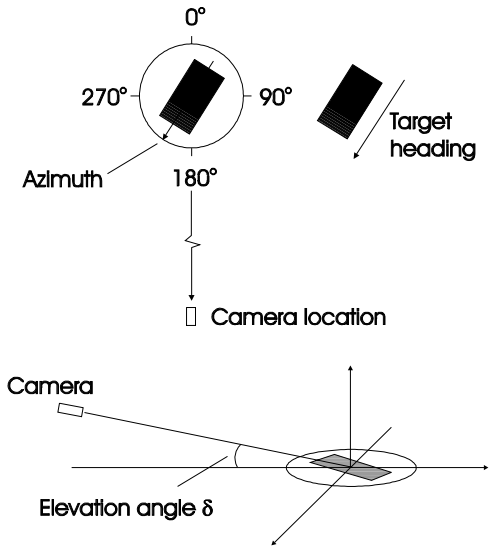


Figure 3.3: Top: Top view sketch of the relative positions of the camera and target, defining the azimuth. Bottom: Perspective view sketch, defining the angle of elevation.

images (typically between five and ten) at each gate setting, and with a step in range of 1.5 m rather than 0.15 m. Furthermore, within the subsequences of unchanged gate setting in the total sequence, a pronounced wavering motion of the gate can be seen along the range axis. A visual inspection indicates that the maximum error can be larger than 1 m. A possible explanation for this behaviour is that the laser and camera were not satisfactorily synchronised, especially when the gate setting was changed. Further investigation of the cause of this effect is not within the scope of this report. Since the technique is currently in its try-out stages at FOI, a solution to the problem can be expected in the future. Improvements to the system will also eliminate the need for manual stepping. Despite these problems, the sequences in question have proven useful for purposes of demonstration, as we are about to see.

3.4 Data Pre-Processing

Before we can match an unknown target against the candidates in the library, we need to create a suitable type of target object from the raw data. The target object should, of course, be of the same type as the candidate objects, i.e a range resolved image with the same cross-range resolution as the candidate objects. Two main tasks can be identified in this process: segmentation of the images to separate the target from its background, and then 3D-, or range-, resolving it from the sequence.

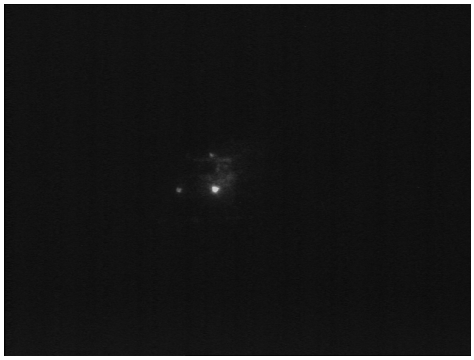


Figure 3.4: Range 7.2 km. The front of the target is now inside the gate.

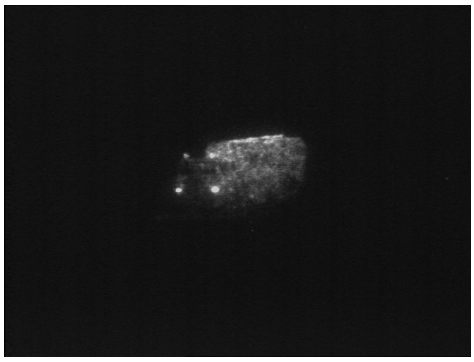


Figure 3.5: Range 7.2 km. The entire target is now inside the gate. The bottom of the target is occluded by the foreground.

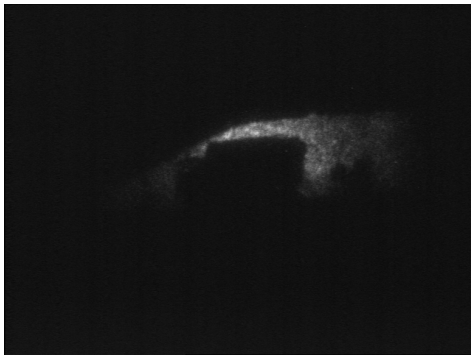


Figure 3.6: The gate is now behind the target, on a slope in the landscape.

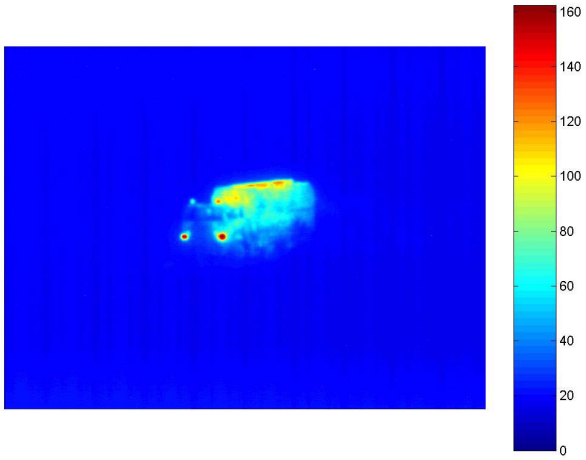


Figure 3.7: The mean target image: Pseudo coded mean of the intensity values from all the 71 frames that contain any part of the target.

3.4.1 Image segmentation

It was mentioned in section 2.1 that the gated viewing technique produces images in which the target has been isolated from its background and foreground. This property facilitates the process of segmenting the target from the images. However, the original gated images tend to be noisy and the target is not uniformly illuminated because of the irradiance distribution over the cross-section of the laser beam. The illumination of the target also varies from frame to frame due to atmospheric turbulence affecting the beam. An image of improved quality can be obtained by fusing several images from the sequence [6]. The algorithm that was implemented in this work uses all frames that contain any part of the target to create the mean image shown in figure 3.7. In its present shape, the position of the first and the last frame containing the object in the sequence has to be fed to the algorithm by the user. The remaining frames are used to construct an improved image of the background, which can be seen in figure 3.8. The background was in this case a slope in the landscape behind the vehicle, also seen in figure 3.6.

The segmentation is carried out through thresholding. However, simple thresholding is complicated by the non-uniform illumination of the target. A closer study of the mean image of the target (figure 3.7) reveals that a threshold set to a level just below that of the low-intensity areas on the front of the driving compartment of the truck, will cause the silhouette of the resulting binary image to bleed over the target's edges in the high-intensity areas at the edge of the roof of the truck, and also near its headlight reflectors. The term bleed is used here to signify that background pixels are mistaken for target pixels, creating false target hits in the proximity of the boundary of the target. This effect is compensated for by using the background image in the following way:

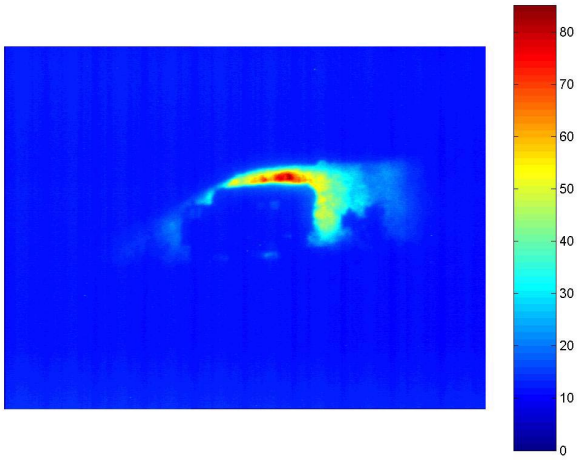


Figure 3.8: The mean background image: Pseudo coded mean intensity values of the 45 frames with the gate behind the target.

The high-intensity area at the roof of the truck in figure 3.7 is caused by the laser beam cross-section irradiance maximum, which also causes a high-intensity area in the background image (see figure 3.8). The idea is now to threshold the mean target image to catch the whole target and turn it into a binary image. Because of the mentioned bleeding effect, this binary image of the target will contain some pixels that are false target hits. Therefore, we also threshold the mean background image to obtain a binary image containing those pixels that are represented by a particularly high intensity in it. These pixels are likely to be on the background, so we classify them as false target hits and remove them from the binary target image. The resulting binary image will then be the silhouette of the target.

For automatic thresholding of the target and background images in figures 3.7 and 3.8, a method based on histogram analysis was implemented. The curve of figure 3.9 shows the outline of the cumulative histogram of the target image in figure 3.7, i.e. it represents the fraction of image pixels with a value higher than a certain threshold. The implemented method locates the maximum second derivative of the curve, which corresponds to a threshold just above the majority of non-target pixels. Experiments with different threshold values suggest that the best threshold should be somewhat larger than that associated with the maximum second derivative. The detected threshold value is therefore multiplied by a constant >1. The background image is also thresholded, but since we only want the high-intensity pixels from this image, a larger value is chosen for the corresponding constant in this case. For the first sequence at 7km, suitable constant values were determined to be 1.45 and 2.5, respectively. These are values which also prove to work reasonably well for the second sequence at the same range. Nevertheless, it is probable that this method will need refinement to handle other sequences acquired under different conditions.

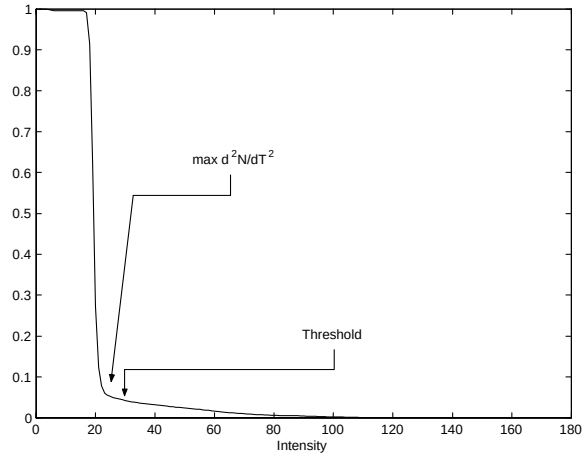


Figure 3.9: Thresholding from the cumulative histogram.

In figure 3.10, the images in figures 3.7 and 3.8 have been thresholded and a morphological "close" operation by a 3×3 element of ones has been applied. The purpose of the "close" operation is to fill in occasional empty pixels inside the silhouette of the target. The set of pixels in the binary background image is subtracted from the set of pixels in the binary target image, and the remaining template is used to select the area of interest when we use the sequence of gated images to obtain a range resolved image. In figure 3.11 the template has been applied to the mean target image. We see that the bleeding effect on the roof of the truck is no longer present. There is still some bleeding in the region of the headlight reflector to the left in the image, since this intensity maximum is due to reflection rather than an irradiance maximum of the laser, so there is no corresponding bright region in the background image. The windows of the truck are not visible, because the laser used in the gated viewing system had its wavelength in the visible part of the spectrum where the window-glass is transparent. It is therefore possible to distinguish the silhouette of a person sitting in the driving compartment of the truck.

3.4.2 3D-resolution

Now that we have obtained the 2D silhouette of the target, we want to use the range-information contained in the gated image sequence to create a 3D object. This is done by analysing the intensity variations of each pixel inside the target's silhouette, as the gate is stepped over it. Each pixel will be assigned a range-value, resulting in a range-coded image. Due to the unstable gate position and the noise from target speckle and atmospheric turbulence in each image, we do not get satisfying results if we just assign a range value to each pixel on basis of the first frame in which its value exceeds some threshold in the sequence. Remember also that the sequences used in this report were non-uniformly stepped with a gate displacement step of 1.5 m rather than 0.15 m. To cope with these problems, a substantial low-pass filtering is performed of each pixel in the time, or range, domain. In figure 3.12 this process is illustrated for

Figure 3.10: The thresholded images of the target (left) and the background (right).

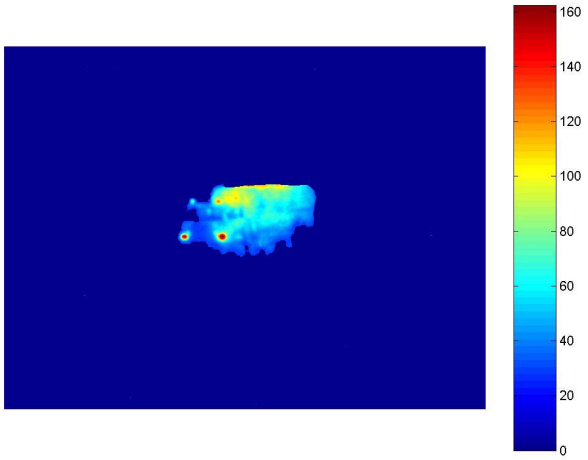


Figure 3.11: Silhouette image: Pseudo coded intensity image of the segmented target. Note the difference between this image and the original one in figure 3.7. The bleeding effect at the top of the target, found in figure 3.7, is no longer present, due to the use of the mean background image in the segmentation method.

300 of the pixels in figure 3.11. Each row of pixels in figure 3.12 represents the intensity variation of a single target image pixel through the sequence. Each row is extended in both directions by padding with its end values, and a horizontal low-pass filter is then applied. Since the range, and time, axis goes from left to right in figure 3.12, we can see by studying the left part of the filtered column that we have a range differentiation of the target image.

Because of the non-uniform illumination, each pixel needs a separate threshold to determine where in the sequence it appears for the first time. After some experimenting, the threshold was set to the 40% level between the signal floor and maximum levels. In figure 3.13, the process is illustrated for two arbitrary target pixels. The top row of the figure shows the original intensity/range variations for the two pixels. In the bottom row, the signals have been low-pass filtered, the threshold level has been set and the range value of each pixel has been determined. The range value of a pixel is set to the number of the frame where it appears for the first time, multiplied by the gate step (in m) between frames.

We have now arrived at a range-resolved image. In the process of getting here, we needed to step through the sequence three times. Once to form the mean images; once again to get the intensity/range variations of each pixel and finally to find the first position where the time-filtered image sequence exceeds a certain threshold, individually determined for every pixel. This is of interest when we evaluate computational efficiency, since the images are re-read from disk in every pass.

The range-resolved image is shown in figure 3.14. The scale to the right of the image is in meters. Note that in this case there will be deviations from the true scale, because the number of frames at each gate position is unknown, and the gate position itself is not stable. These effects both add to the uncertainty in the gate position in each sequence frame, and this uncertainty will be incorporated into all the range values of figure 3.14. To simulate the result from a proper sequence, a step of 0.30 m between frames was assumed, as this value gave reasonable agreement of the range-resolved image with the known dimensions and orientation of the target. All pixels that lie outside the segmented silhouette are set to a large negative value. In figure 3.14 there is some noise, resulting mainly from speckle noise in the original images in the sequence. Figure 3.15, is a 5×5 median-filtered version where the range levels are seen more clearly.

The effect that there are more levels in this image than would be expected with the 1.5 m step, can be due the unstable gate position in combination with the low-pass filtering. For example, if a part of the target lies just behind the gate at a certain position, the jittering of the gate can make it visible in the next frame, despite that the gate should not have moved. If this happens close enough to the next gate step in the sequence, the low-pass filter will tend to move the pixels in question closer in range when thresholding to obtain the range-resolved image. This effect may be seen in the filtering and thresholding of the leftmost curve in figure 3.13. Note that we cannot tell if the behaviour of this particular curve is due to gate fluctuations or speckle/turbulence noise, just by studying the intensity variations of this single pixel. In order to draw such conclusions, we would also have to study the intensity variations over the target as a whole.

By visually inspecting figure 3.15, we find that the number of pixels in the image is much larger than what is needed to represent the actual resolution of the

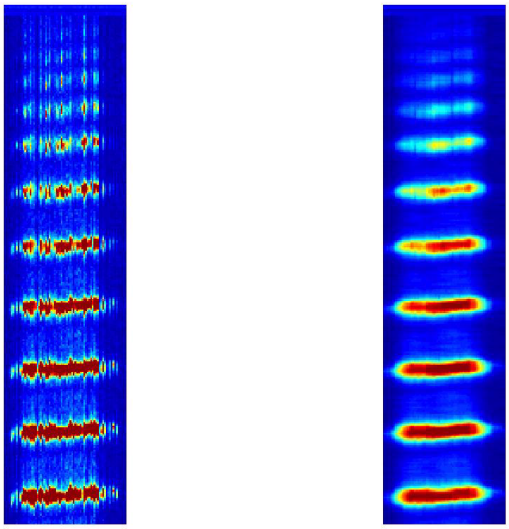


Figure 3.12: Illustration of the range domain filtering. Each pixel row, from left to right, represents the intensity variation of a single target pixel through the sequence. Left: Original intensity variations of 300 pixels. Right: Low-pass filtered version.

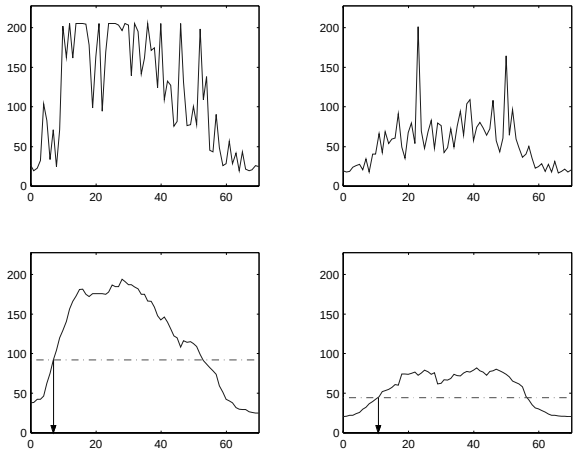


Figure 3.13: Examples of range filtering and thresholding. Top: Actual time-response of two different image pixels. Bottom: Filtered response (solid), threshold levels at 40% (dash-dotted) and the assigned range values (arrow). Frame number, corresponding to range, is indicated on the x-axes. The values on the y-axes represent pixel intensity.

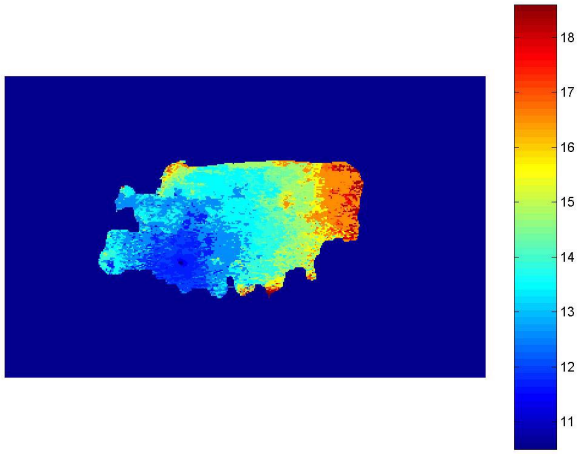


Figure 3.14: The range-resolved image. The range scale is in m. The absolute range values are arbitrary.

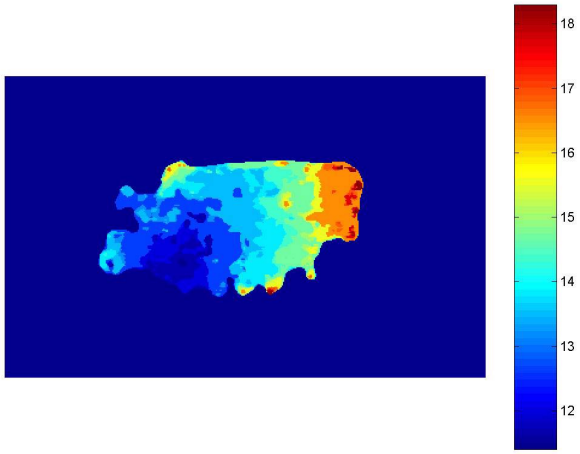


Figure 3.15: A median filtered version of the range-resolved image of figure 3.14.

collected data. This means that we can re-sample the image to a certain extent without losing great amounts of information. Furthermore, to promote speed in the recognition process we do not want to deal with more data than necessary. Hence, the range-resolved image (figure 3.14) is re-sampled by dividing it into squares of the size 0.2 m $\times$ 0.2 m, and taking the median of all the pixels in each square as the sample value for this square. The number of pixels in each square will depend on the range to the target and the focal length of the lens in the laser radar system. The resulting object is presented in figure 3.16. Note that the scale in the range direction has been inverted compared to that of figure 3.14 so that it is now positive in the direction towards the observer, and the zero plane goes through the most distant pixel of the object. This adaptation was made in order to obtain a target object of the same type as the candidate objects. The image of figure 3.16 is now the object used as input to the recognition algorithm. The reader should compare the target object of figure 3.16 with the candidate object example of figure 3.17. Both objects are range-coded images with a cross-range resolution of 0.2m/pixel.

3.5 Target Recognition by Object Matching

Up to now, this chapter has treated the construction of 3D objects from the model library and the target data. We now want to compare the target object with the candidate objects and find the closest match. A method described in [5] is used for doing this. At this point, the reader is recommended to briefly return to and review the road-map of figure 3.1.

The object matching algorithm consists of two main steps. In the first step, the candidate set is subjected to pruning by an algorithm that correlates one-dimensional features of the objects. Although a 1D screening in most cases will not be as accurate as a complete 2D and 3D match, we can use it as a fast way of discarding the majority of non-likely candidates, and thus saving computational time. The pre-screening is described in section 3.5.1. The second step of the matching algorithm is the matching of object surfaces and boundaries, which is presented in section 3.5.2.

The processing that turned the raw data into the target object in figure 3.16 was highly sensor-specific, but a similar object can just as well be obtained from data from a scanning system or a 3D FPA, with other data processing methods. In an ideal case, the object matching strategy, which is the topic of this section, would thus be sensor-independent. Since the situation is less than ideal, however, there are some differences between an object obtained from the gated viewing system used in this work and the scanning laser radar that was used in [5]. The implementation of the algorithm in this work basically follows that of [5]. The sensor-dependent differences, however, motivate the omission of a penalty term in the surface matching step in this work, a change which is explained in section 3.5.2. The algorithm implemented in this work also differs from that of [5] by a different way of calculating the boundary matching score. The boundary matching algorithm of this work is also presented within section 3.5.2.

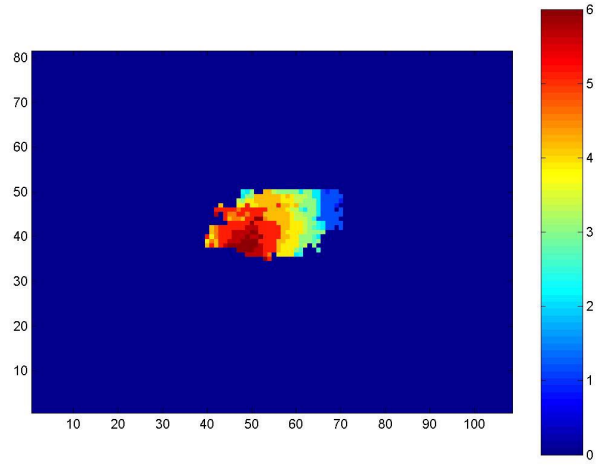


Figure 3.16: The target object that is used as input to the matching algorithm. Compare the target object to the candidate object example shown in figure 3.17, which is of the same type. Both objects have cross-range resolutions of 0.2m/pixel.

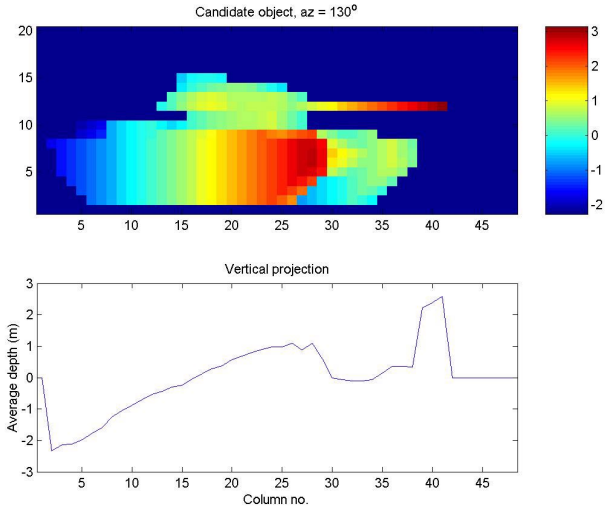


Figure 3.17: Stored candidate features in the candidate library. A 3D template image, constituting the candidate object, and a 1D vertical range projection of a Centurion tank at 130 degrees azimuth angle. A corresponding horizontal projection (not shown in the figure) is also stored for each candidate.

3.5.1 1D Pre-Screening

One-dimensional horizontal and vertical projection vectors of the same kind as those stored in the candidate library can similarly be extracted from the target object (figure 3.16), and then used in a first coarse step in the matching process. In [5], this pre-screening process is used to reject about 80 % of the candidates, keeping the 20 % with the best scores for further matching. Whether or not this is the appropriate fraction of templates to be rejected is a question that may be subjected to further investigation (see also section 3.7, which deals with algorithm robustness). The correlation measure used to measure the degree of similarity between the target and candidate object curves is (correlation measure and notation as in [5]):

$$m_c = \frac{N_{f \cap g}}{N_{f \cup g}} = \frac{\sum_i [(n_{f_i} + n_{g_i})(\overline{f_i} - \mu_f)(\overline{g_i} - \mu_g)]}{\sigma_{\overline{f}} \sigma_{\overline{g}} \sum_i (n_{f_i} + n_{g_i})} \tag{3.1}$$

where $\overline{f_i}$ is the value of the i:th element of the average range vector from the target object; $\overline{g_i}$ is the value of the i:th element of the average range vector from the candidate object; μ_f, μ_g are the means and $\sigma_{\overline{f}}, \sigma_{\overline{g}}$ the sample standard deviations of the curves $\overline{f}$ and $\overline{g}$, respectively. n_{f_i} is the number of pixels of the column or row used for calculating the i:th element of the average range vector of the target. n_{g_i} is the corresponding number for the candidate. The vector $(\overline{g} - \mu_g)$ is the vertical or horizontal projection vector that is calculated in advance and stored with each candidate object in the candidate library (see section 3.2 and figure 3.17). The vector $(\overline{f} - \mu_f)$ is obtained analogously from the target object. If the object and candidate vectors are of different lengths, the relative position of the two vectors is shifted so that the elements corresponding to the leftmost or topmost part of the objects, respectively, become aligned.

For candidate objects created from the same model, the projection curves for similar aspect angles are similar to each other [5]. This should favour a robust pre-screening even if the aspect angle of the actual target should be somewhere between that of two neighbouring candidates. Candidates are stored in intervals of 5^o, and hence we can consider the aspect angles of candidate neighbours to be similar.

The correlation measure m_c in equation 3.1 is computed for both the horizontal and vertical projection vectors of the target/candidate pair for every candidate, so we want a way to weight the two numbers together:

Figure 3.18 shows the placing of all the 72 TGB30 candidates, among the total 792 candidates in the library, after correlating with the 1D-projections of the TBG30 target object of figure 3.16. A desirable property of a pre-screener would be to give a fair placing of the correct candidate and probably a few of its neighbours, which have similar projections. Candidates which are completely misaligned or otherwise incorrect should score less "points". We see from the results in figure 3.18 that the correlations of the vertical projections in this particular case are better at capturing this behaviour, since those of the horizontal projections leave only a single TGB30 candidate in the top 20%, and this candidate is not the correct one. Actually, the exact orientation of the target during the data acquisition was never measured during the tests, but the azimuth is likely to be somewhere between 210 and 220 degrees, as was stated in section 3.3.

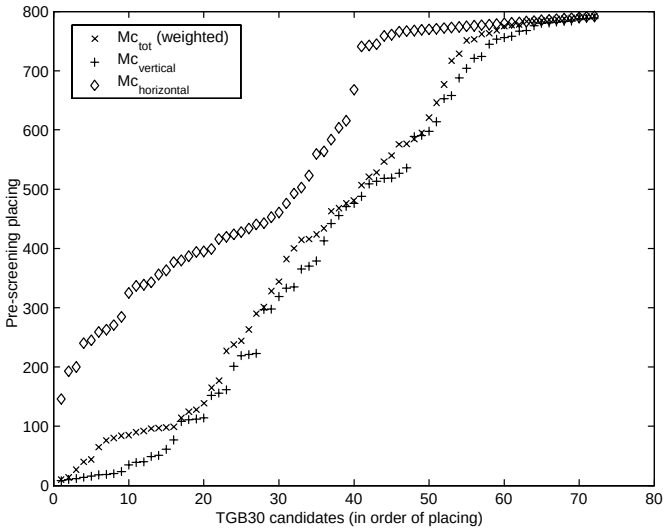


Figure 3.18: The placing of all TGB30 candidates in the pre-screening.

The difference between the performances of vertical and horizontal projection correlations might have been expected, since the vertical projections would be better at discriminating between different azimuth angles and the horizontal projections would do best at different angles of elevation. The azimuth is the only angle that was varied in the candidate library in the course of this work.

Let us now look more closely at how the pre-screener ranks the three TGB30 candidates oriented at 210^o, 215^o and 220^o. The horizontal projection matching ranks these 285th, 337th and 399th, and the vertical matching at 114th, 51st and 18th, among all the 792 candidates. However, we want to use information from both projections by weighting the two correlation scores together for every candidate. The angle of elevation was zero in the candidate library, and approximately zero in the acquired gated viewing image sequences. We expect the vertical projections to be better at discriminating between different azimuth angles than between different elevation angles. The opposite would then apply to the horizontal projections, and therefore it seems reasonable that the vertical projections should be given more weight the the horizontal ones. A small amount of testing suggests that a ratio of 4:1 with the majority of weight on the vertical projections (i.e. $m_{c,tot} = 0.8m_{c,vertical} + 0.2m_{c,horizontal}$) produces gratifying results, at least in this particular case. The three candidates in question now rank 97th, 44th and 10th.

The general behaviour of the different projections in the pre-screening is presented in figure 3.18. Note that the TGB30 candidates are sorted along the x-axis by their relative placing in each case, and that it is not possible to see how the placing of any particular candidate differs between the projections.

3.5.2 Surface and Boundary Matching

We now turn our attention to those candidates that remain after the prescreening, and continue with a more detailed analysis. Still following the course of [5], we will perform 2D and 3D analyses by matching the boundary and surface of the target object to those of the remaining candidate objects.

In principle, the process of matching surfaces and boundaries is straightforward. Before we start, however, we need to align the target and candidate objects so that we match the corresponding parts with each other. A simple method of doing this was chosen. Using translations, and not considering rotations, we align the objects in the following way: The target part which is least likely to be obscured by the foreground is probably the roof (or its equivalent), so the relative position of the objects is shifted to align the topmost non-empty rows. Distant parts of the target will be more prone to be concealed than those at closer range, so we shift the objects along the range axis to bring the least distant pixels of the candidate and target objects to the same range value. As there is no preference in the left-right direction concerning partial occlusion during the acquisition, it is arbitrarily elected to align the rightmost pixels both objects.

Surface Matching

The surface matching score for a target/candidate pair is obtained simply by counting the number of pixels that have similar range values in the target and candidate objects. By subtracting the pixel values of the target object from those of the candidate object, we produce a difference image. The number of pixels in this image whose absolute values lie within some tolerance is then counted. More explicitly, the surface matching score M^{srf} for a particular candidate object T is given by:

$$M^{srf} = \sum_i I(|R(i) - T(i)| \leq \tau_{srf}) \tag{3.2}$$

where the index i runs through all the pixels of the target object. R is the target object and I is an indicator function which assumes the value one if the argument is true and zero otherwise. According to [5], the basis for setting the value of the surface matching threshold τ_{srf} should be the width of the peak of the range error density function of the range return of the laser radar. The statistical properties of the range error of the gated viewing system have not been investigated in this work. A heuristically chosen value for τ_{srf} of 0.2 m has been used throughout the testing and has performed satisfactorily.

The authors of [5] now proceed by calculating a penalty term, arising from pixels where the system has returned a more distant range value than that of the candidate template. The recommended threshold for this penalty is greater than the largest dimension of the largest model in the library. The gated viewing system, however, is different from the laser radar used by [5] in that it is not as easy to determine if a missing pixel in the object is due to that part of the target being obscured, or behind the gate. The penalty term is not used in this work.

Boundary Matching

In this work, the boundary matching score is calculated in a somewhat different way than in [5], although the main idea is the same. The matching score is essentially the number of boundary pixels common to the target and candidate objects. Such a measure will work when the cross-range resolution of the objects is not too large. If the pixels were too small for the actual resolution achieved by the system, we would risk getting misleadingly low scores even for an essentially correct candidate.

To count common boundary pixels, we first need a method of picking out the boundary pixels from the target and candidate objects. The particular class of images representing the objects guarantees an easier-than-average edge detection. First, we produce a binary image from each target or candidate object, where the background pixels are set to zero and the remaining ones to one. This is easily accomplished by applying a threshold at a small negative value, because the background pixels are the ones previously set to a large negative value in both the target and candidate objects, and all other pixels have values greater than zero. Next, the binary image is convolved with the element:

$$\frac{1}{18}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 10 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.3}$$

This convolution results in an image where all the boundary pixels on the object have values in the interval $]0.5, 1[$. No pixels inside the object or background, or on the boundary pixels of the background will get values in this range. Thus, the object boundary is detected. The shape of the convolving element could, of course, be varied. For example, the element

$$\frac{1}{9}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 5 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.4}$$

would give similar results, but would behave differently at concave corners of the objects, where the corner pixel would be left out from the set of boundary pixels..

The boundary matching score, designated M^{sil}, can now be calculated for each target/candidate pair. Formally, it is the number of elements forming the intersection of the two sets that contain the boundary pixels of the target and candidate object, respectively. Because the bottom part of a target will often be likely to suffer from partial occlusion, this part should be omitted from the boundary match. In this work, the score is obtained by comparing only the top 2/3 of the boundary images.

Weighting of Surface and Boundary matching scores

For a particular target, we have now obtained a surface matching score and a boundary matching score for each of the candidates that remained after the prescreening. In order to produce a ranking list of these candidates, where the top candidate is declared to be the recognised target, we will weight the two scores together in a linear combination. The relative weights of the scores should be

determined on the basis of the relative resolutions along the range and cross-range axes. Lacking such information, the two scores are given equal weights, but both scores are first normalised so that the maximum value of each will be that of unity. Explicitly, the total matching score for a particular candidate will be:

$$M = \frac{M^{srf}}{N^{srf}} + \frac{M^{sil}}{N^{sil}} \tag{3.5}$$

where N^{srf} is the total number of pixels in the target object and N^{sil} is the number of pixels in the top 2/3 of the target object boundary.

3.6 Results

The entire algorithm, including data pre-processing and object matching as described in sections 3.4 and 3.5, was applied to the two gated viewing sequences at 7.2 km available from the measurements in Älvdalen in October 2002. One algorithm parameter was changed between the tests on the two sequences. In the second sequence, an assumed gate step of 0.15 m produced better agreement with the actual scale and orientation of the target than the 0.30 m used in the first sequence Note that the gate was really stepped manually, and thus in a non-uniform manner during the measurements, so a certain amount of manual parameter fitting was necessary to get results that corresponded acceptably to the actual situation.

In this section, we will review some aspects of the results from these two algorithm tests. We will start by looking at the execution times of the constituent parts of the algorithm, because these times are important factors in determining the possible areas of application of the algorithm. The algorithm was tested on a PC with a 400 MHz processor, which is several times slower than most PCs available today. MATLAB was used for the implementation.

The most time-consuming part of the entire process is the data pre-processing into a range-resolved target object, which takes in the order of 90 seconds to perform. Partly, this is because the memory of the computer used was to small to contain the entire GV sequence at one time, which meant that every image had to be cached from disk on every pass through the sequence, slowing the process. A rough estimation suggests that the reading of the data from disk may be accountable for up to about 45 seconds of the total execution time. The method for thresholding the mean target and background images is another area of possible improvment, which currently stands for about 30 seconds, or 1/3 of the execution time of the data pre-processing.

The pre-screening part of the recognition algorithm, whose main purpose is to reduce the total mass of computations by rejecting the bulk of incorrect candidates by dealing only with 1D data, finishes with the 11 models and 792 candidates of the library in about 3.5 seconds on the PC that was used. If all the 792 candidates in the library are instead fed directly to the surface and boundary-matching algorithm, without any prior pre-screening, the execution finishes in only about 6 seconds. This is approximately twice the time of the pre-screening. The times of execution of both processes are expected to increase linearly with the number of candidates. We see that in order for the pre-screener to serve its purpose as a time-saving device, about 60% or more of the candidates

Table 3.2: The results from passing the two gated viewing sequences through the ATR algorithm, as described in sections 3.4-3.5.

Sequence	Identified as	Total	Surface	Boundary
1	TGB 30 at 215^o	0.6830	0.1901	0.4930
2	TGB 30 at 265^o	0.8751	0.5788	0.2963

will have to be discarded in this step. The performances of both the pre-screener and the complete matches are also studied more thoroughly in section 3.7.

We now turn our attention from the computational efficiency of the algorithm, and instead study its output. From the two data sequences on which the algorithm was applied, the target was successfully identified as a TGB30 terrain vehicle at azimuth angles of 215^o and 265^o, respectively. The matching scores of these candidates are presented in table 3.2.

A few comments on these figures: The gate step parameter was adjusted to give a reasonable overall agreement with the scale of the target, but the non-uniform gate stepping can have produced deviations from the true surface inside the object. This can result in lower surface matching scores for the correct candidate than would otherwise have been expected.

The identification accuracy was a perfect 100% in these two tests. To evaluate this result, however, we should compare the appearance of the TGB30 vehicle with the other models in the model library (table 3.1). Some of the models that are of roughly the same size as the TGB30 are quite different in overall shape, and the models which roughly resemble the TGB30 in shape (the TGB11 and the TGB13) are significantly smaller. This would indicate an easy match of this target in this particular model library. The robustness of the matching algorithm is explored further in section 3.7.

3.7 Algorithm Robustness Tests

In this section, we will study the robustness of the ATR algorithm to perturbations to the input data. The robustness was tested using a Monte-Carlo method with synthesized target data. The target data were synthesized by directly creating artificial target objects, rather than by simulating gated image sequences at the acquisition level. The data pre-processing stages were thus exempt from the robustness tests. There are many possible ways of disturbing data to simulate different situations which can arise in real life. This work was restricted to deal with two of these types of disturbances:

- *Angle of elevation:* The candidate library created from the model library was restricted to contain only candidates at an angle of elevation, δ, of zero degrees, i.e. the models were only viewed from an observation point on the same level of height. We shall examine how great a restriction this will be on the applicability of the target recognition algorithm if data is acquired at positive angles of elevation, i.e. when the target is being viewed somewhat from above.

- *Range-noise:* One type of data inaccuracy was also simulated by adding normally distributed range-noise to each pixel of the synthesized objects.

This was done to examine the order of the range-accuracy that is required from the acquisition and pre-processing stages, to provide a basis for robust target recognition.

For each test, a relatively large set of simulated target objects are created. The simulated objects are range images of 0.2m × 0.2m/pixel resolution, corresponding to the type of target object seen in figure 3.16. The synthesized objects are then sent to the matching algorithm to give us some statistics. For each of the two tests, i.e. the elevation angle and noise sensitivity tests, we study how the accuracy of the surface and boundary matching algorithm varies with increasing levels of perturbation. We also look at the variation of the average placing of the correct candidate in this match, and evaluate the performance of the pre-screener. The sensitivities to elevation angle variations and range-noise are measured separately. Sets of 1000 synthetic target objects are generated at every level of disturbance, and 16 different levels are tested in each of the two tests.

The creation of each simulated object follows a few basic steps. First, a model is randomly picked from the model library and rotated to an arbitrary azimuth angle. The azimuth angle can take on any value in the interval 0^o to 360^o, and not just the 72 values in 5^o increments that were used to build the candidate library. Next, the mesh of the CAD-model is sampled to obtain a range image in the same way as when the candidate library was created. However, to avoid always getting the exact same sampling of the models as in the candidate library, we first displace the uniform sampling grid by random amounts up to 0.2m in the two cross-range directions. Any positive elevation angle is also specified in this sampling step. Once the range image has been created, noise of a specified variance can be added to each pixel before sending the image to the matching algorithm for testing.

In table 3.3, a confusion matrix based on tests of 1000 synthesized objects at zero degrees of elevation and zero noise is presented. It was created from the results of the 1000 tested objects on the first disturbance level, i.e. no disturbance, of the elevation angle test (see section 3.7.1 and, for example, figure 3.19). Thus, in table 3.3, the only differences between the synthesized objects and the objects in the candidate library were that the test objects were created from a randomly displaced sampling grid and that the azimuth angle was arbitrary. We expect a high degree of accuracy under these conditions.

Each row of the confusion matrix represents the model used for generating synthetic images, and each column represents the parent model of the candidate that the synthesized object was identified as. (In the case of a correct match, these two will of course be the same). Note that at a total of 1000 tested images from eleven models, an average of approximately 91 tests have been performed for each model, but since they were picked randomly the actual number varies somewhat between models.

We see that for three of the eleven models, the accuracy (as measured in this case) was a complete 100%. The lowest accuracy is found for the T72 Kvarn model, which was incorrectly recognized as the generic T72 model in 11% of the cases. This result should not be too surprising, considering the similarity of these two models. The main difference between the two is that the T72 Kvarn model was acquired from a laser radar scan of a target tank which had its turret and barrel rotated by about 15^o in the clockwise direction relative to its body,

Table 3.3: The confusion matrix for zero elevation and no added noise. The synthetic target objects differ from the candidate objects in their sampling and arbitrary azimuth angle. Synthesized target objects from the models in each row have been identified as candidate objects of the models in each column in the shown percentage of cases. The percentages have been rounded to integer values .

Target\Ident.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1. bmp1	97	2						1			
2. bmp3	2	96						1			
3. Centurion			100								
4. ikv91		2		96			3				
5. lvkv90					100						
6. mtlb				2		97			1		
7. T72				2			95	2			
8. T72 Kvarn							11	89			
9. tgb11									99	1	
10. tgb13									2	98	
11. tgb30											100

while the barrel of the generic model points straight ahead. The two T72 models also differ somewhat due to some dents in the surface of the T72 Kvarn.

The accuracy percentages along the diagonal of table 3.3 concern only whether or not the correct model was recognized, and does not give information on the recognition of the correct candidate. We now change our measure of accuracy to one considering the correct identification of both model and azimuth angle. The correct candidate corresponding to a particular synthetic target object is defined as any one of the two candidates, generated from the same model as the synthetic image, which are closest to it in azimuth angle.

3.7.1 Sensitivity to Variations in the Angle of Elevation

In the investigation of the algorithm's sensitivity to variations in the angle of elevation, sets of 1000 synthetic images were generated at each such angle from 0^o to 30^o in increments of 2^o. All parts of the algorithm (pre-screener, surface match and boundary match) were included in the test, but the pre-screener was in this case not used to reject any candidates before the 2D and 3D matches. This setup was chosen in order to study the behaviours of the pre-screener and surface and boundary matches separately. The resulting accuracy values for the combined surface and boundary match are displayed in figure 3.19. The accuracy starts off at about 90%, decreases with increasing elevation and has dropped to 73% already at $\delta = 4^o$ and below 60% at $\delta = 6^o$. The accuracy is the percentage of times that the correct candidate is placed first (i.e. is given the highest matching score of all candidates) by the surface and boundary matches combined.

Now knowing how the *accuracy* deteriorates with increasing elevation, we may also want to know what happens to the average *placing* of the correct candidate among the total 792 candidates. While the accuracy is an important measure for a stand-alone ATR system, it is not the only way of measuring the

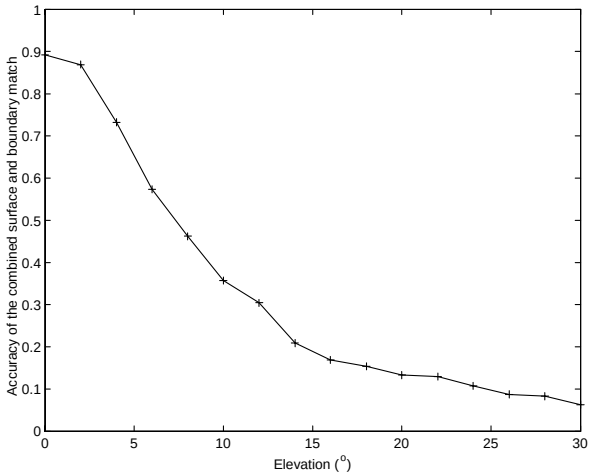


Figure 3.19: The variation with elevation angle of the accuracy of the combined surface and boundary match. One thousand synthetic objects were tested at each of the indicated angles. The accuracy is the relative frequency of the event that the correct candidate (= correct model + azimuth) is given the highest combined matching score of all the candidates (= is placed first).

capabilities of a system that works in a multi-sensor environment or in operator assisted target recognition. Even if the correct candidate does not place as number one, the algorithm can still give valuable information if the correct candidate is placed sufficiently high. Ranking data for the correct candidate in the surface and boundary match is shown in figure 3.20. Up to $\delta = 6^o$, the correct candidate is, on average, placed among the best 7 ($< 1\%$ of all 792) candidates. At this angle of elevation we also see that in 90% of the cases the correct candidate is placed in the top 11 ($\approx 1.4\%$) and in 95% of the cases it places better than 22 ($\approx 2.8\%$). The behaviour at angles of elevation up to 6^o can be studied more easily in figure 3.21, which is an enlargement of the lower left part of figure 3.20. The placing of the 5% worst cases is seen to deteriorate significantly for $\delta \geq 4^o$.

In figure 3.22, the mean ranking of the correct candidate by the total matching algorithm is compared with the placing that would be given by the surface or boundary matches alone. The two-dimensional boundary match is seen to be much less capable of distinguishing between different candidates as the angle of elevation increases.

We now turn our attention to the performance of the first part of algorithm, namely the pre-screener. In figure 3.23, we can see that the pre-screener on average places the correct candidates well inside the top 10% of all candidates for all the elevation angles that were studied (At $\delta = 0^o$, the average placing is at about 2%), which seems promising at the first glance. It is suggested in [5] that the pre-screener be used for discarding 80% of the candidates. At small elevation angles (0^o-2^o) more than 95% of the correct candidates are seen to place among the top 10%, but the worst-case performance deteriorates as the elevation angle increases. In order not to reject any correct candidates in this

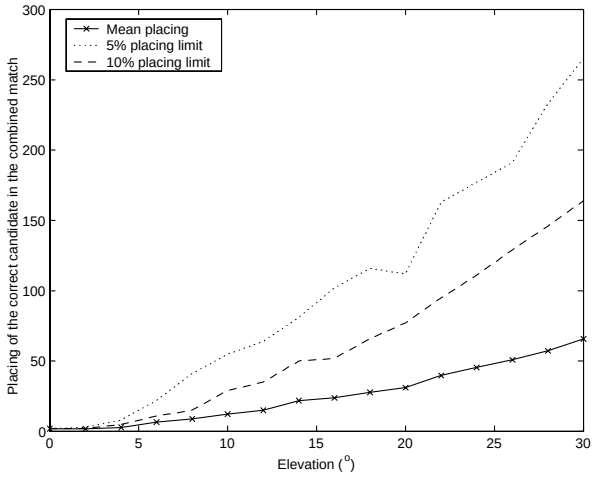


Figure 3.20: The sample mean, 5-percentile and 10-percentile of the placing of the correct candidate among the 792 candidates in the combined match.

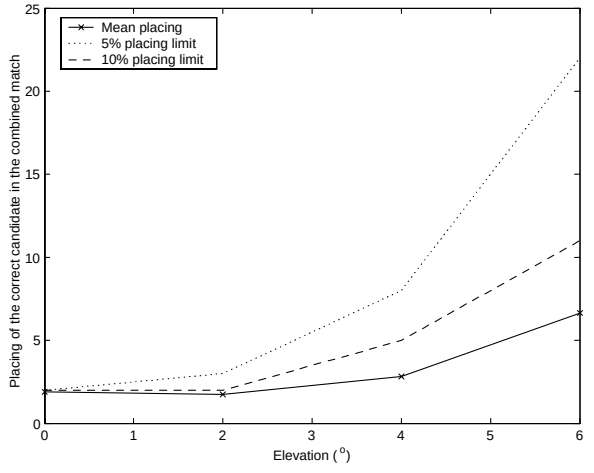


Figure 3.21: Enlarged version of the lower left part of figure 3.20.

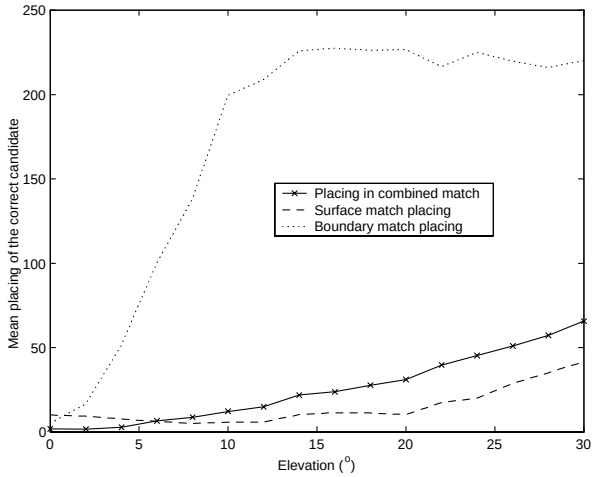


Figure 3.22: The mean placing of the correct candidate in the surface, boundary and combined matches.

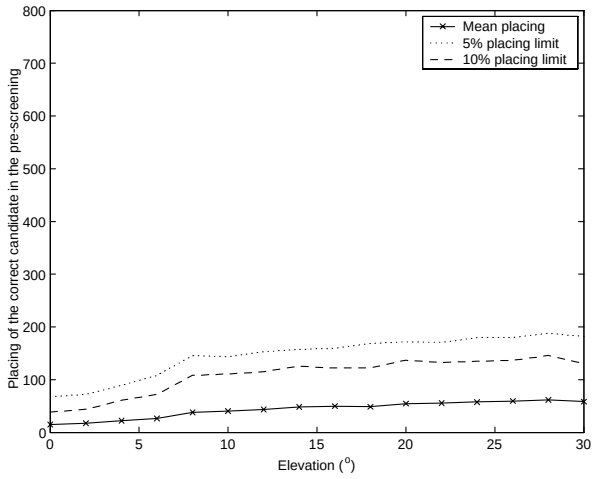


Figure 3.23: The mean placing of the correct candidate in the pre-screening, along with its 5- and 10-percentiles.

step, the percentage limit of the pre-screener must be set with some caution.

3.7.2 Sensitivity to Range Noise

The next set of tests concerned the robustness of the recognition algorithm to normally distributed range noise. Target objects were synthesized following the same path as for those of the previous test, but now always at zero angle of elevation. To the pixels of the range image was then added normally distributed noise with zero mean and specified variance, before the images were sent to the matching algorithm. The results from the noise-test are presented in figures 3.24 to 3.27. The results are presented in an analogous way to those of the elevation angle tests (figures 3.19, 3.20, 3.22 and 3.23), except that the variable is now the standard deviation of the gaussian noise, which is varied between zero and 0.45 m in steps of 0.03 m. Again, 1000 images were synthesized and tested at each noise standard deviation level.

From figures 3.24 and 3.25 we note that the accuracy and mean placing do not change significantly for noise levels of $\sigma \lesssim 0.1$m. At higher noise levels, the accuracy drops, and the mean placing of the correct candidate also deteriorates somewhat. Even so, the correct candidate on average still places in the top 1-2% or so of all the 792 candidates for all the tested noise levels, and at the highest tested level 95% of the correct candidates still place among the 3% with the highest scores.

This time it is, not surprisingly, the surface matching scores that will do worse than the boundary matching scores at giving the correct candidate a good placing as the noise level increases, as can be seen in figure 3.26. This indicates, again, that the weighting of the boundary and surface scores should be based on the relative resolutions and expected noise levels in the range and cross-range directions (section 3.5.2).

No clear trend can be identified from figure 3.27, which displays the ranking of the correct candidate by the pre-screener. The reason for this can be that the effect of the white noise decreases in the averaging performed when forming the horizontal and vertical projections, which diminishes the impact of this type of perturbation. At all the tested noise levels, more than 95% of the correct candidates are ranked among the top 10% by the pre-screener.

3.8 Discussion

This section has covered some major parts of the process of automatic target recognition. We started after data acquisition by carrying out image segmentation and transformation of the data into a suitable object to perform one-, two- and three-dimensional matching against a library of candidate objects (an example of a generative, as opposed to discriminative, method, see [8]).

Segmentation of the target in the gated sequence was complicated by non-uniform illumination. The approach used to carry out the segmentation involved the use of images of both the target and its background. This is really a special case of gated viewing measurements, since such a background may not always be present in the landscape. In any case, the method of segmentation used here was in some ways developed ad hoc, and a more robust and efficient method will be needed for performing under a wider variety of conditions. It is probable that

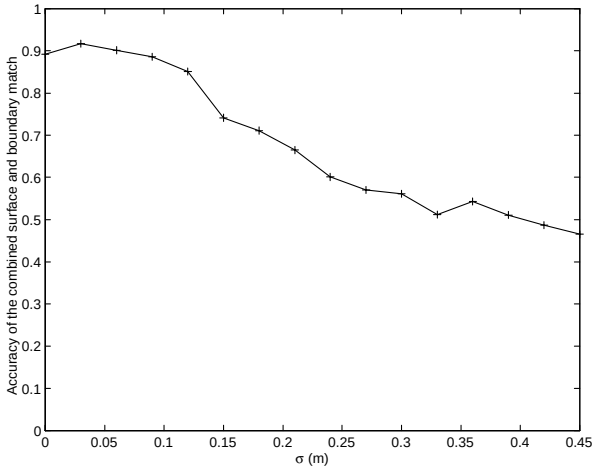


Figure 3.24: The variation of the accuracy of the combined surface and boundary match with the standard deviation of the added noise.

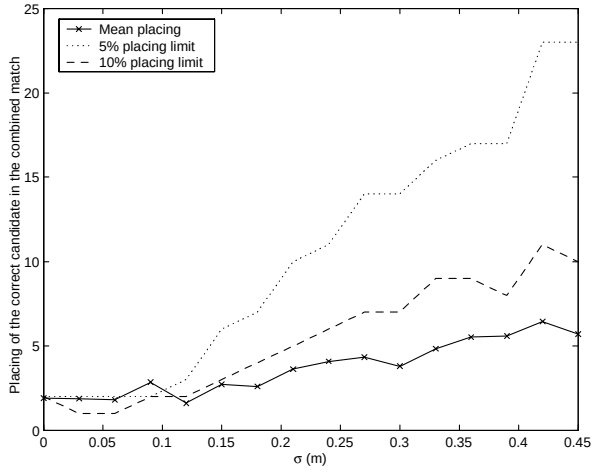


Figure 3.25: The sample mean, 5-percentile and 10-percentile of the placing of the correct candidate among the 792 candidates in the combined match, under the influence of normally distributed noise of increasing standard deviation.

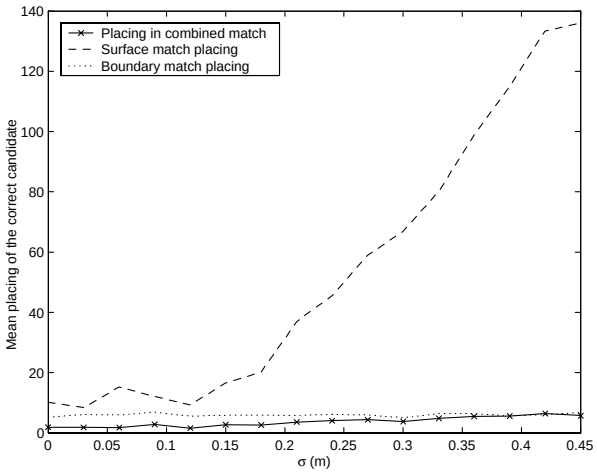


Figure 3.26: The mean placing of the correct candidate in the surface, boundary and combined matches under the influence of normally distributed noise of increasing standard deviation).

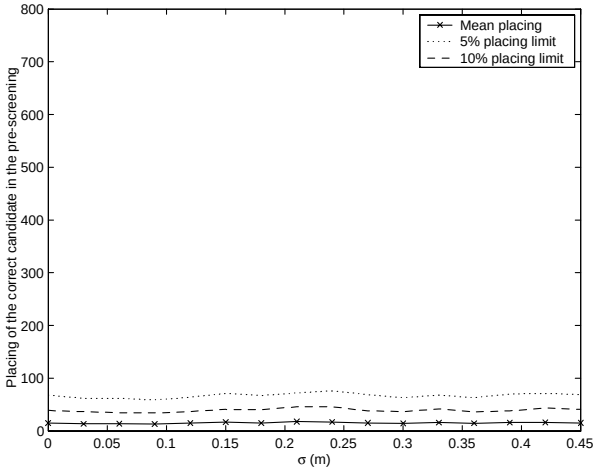


Figure 3.27: The mean placing of the correct candidate in the pre-screening, along with its 5- and 10-percentiles.

a better method could be devised, especially considering the limited amount of time put into solving this problem in the course of this work, and the extensive amounts of attention that the subject of segmentation has been given in the literature on image analysis.

The candidate library is the library of objects with which we compare our object that we have constructed from the gated viewing sequence, and the properties of these candidate objects are thus of importance to the results that we will get. One thing that has not been commented on so far in this work is that some target types may not only vary in aspect angle, but can also have different articulations. An example of such articulations is that a tank, such as the T72 or the Centurion of the model library, can rotate its turret and also raise or lower its barrel, thus changing its appearance. A way of dealing with different articulations would be to modify the parts of each CAD-model to form a number of new models from the original one, each new model having a different articulation. The candidate library would then be generated from these articulated models. This would increase the number of candidates and thus the computational load required to perform the recognition by matching, but it may not be necessary to make a large number of articulated models from each original. Considering that the recognition algorithm sometimes confuses the two T72 models (as mentioned in section 3.7) which differ in turret rotation angle by about 15^o, it may be sufficient to generate new articulations for each original model in steps of perhaps 5^o or 10^o within some interval of interest.

Another problem with the candidate library is the way it is generated. In order to save time, an already existing function (courtesy of Tomas Carlsson, FOI) was used to sample the mesh of the models in a uniform square grid of points. The resolution of this grid (i.e. the distance between a grid-point and each of its 4-neighbours) was set to 0.2 m, which means that narrow protruding objects, such as the barrel of a tank, may be completely lost in the sampling step, especially if their orientations lie near the horizontal or vertical. This has been checked to not happen in the particular candidate library used in this work, by may have been responsible for some of the misclassifications in the robustness tests, since the synthetic images were generated from a displaced sampling grid. In the case of the objects generated from real gated sequences, the range images are sampled by taking the median value of each 0.2m $\times$ 0.2m square, which leads to a similar situation.

A related problem is the one of aligning the objects-to-be-compared, so that the corresponding parts of the surfaces and boundaries of both are matched. As explained earlier, they are simply aligned on basis of the topmost, rightmost and least distant parts of the object. This method will in some ways have an injurious effect on the robustness of the algorithm, e.g. if the barrel of a tank (pointing, for example, to the right or towards the observer) is lost during sampling it would cause a severe misalignment when matching against the correct candidate. This would most probably lead to a misclassification even though the surfaces of both objects would have fitted nicely, had they been properly adjusted. This problem can also arise when mis-sampling is not an issue. The roof of a truck, although less likely to be occluded by the foreground, may have additions such as a roof-rack carrying some bulky load, persons or antennas, which would also cause misalignment. Similarly, this would concern the possible tilt of the target in the cross-range plane. These are problems which are not addressed in, for example, [5]. Some work would be required to find a method which produces a good

object alignment without increasing the computational loads to unacceptable levels.

The rough comparison of the execution times of the pre-screener and the surface and boundary matching algorithms indicates that the pre-screener will play the role of a timesaving device if a larger fraction than about 60% of the candidates can be safely rejected.

If the capability of the pre-screener to give high enough scores to the correct candidates reaches the levels seen in the robustness tests also with real data, the share of candidates which may be safely discarded can be great enough to justify its use. Possible failure modes of the pre-screener are the same as those just mentioned for the 2- and 3-D matches, namely the sampling and alignment problems. In this case, though, we have only one degree of freedom for each projection vector, instead of two and three, respectively. The first non-zero element on the target projection vector is aligned to that of each candidate projection vector before correlating them. Since the projection vectors are created from the sampled range images of the target and models, they are also likely to suffer sampling and alignment problems.

In section 3.7, we saw some quantitative measures of the accuracy and robustness of the recognition algorithm. Such figures are not to be seen as exact measures, since they will ultimately depend on the nature and size of the candidate library used, and since the one used in this work was quite small. In [5], the accuracy of the algorithm is boosted through the implementation of a "second look" strategy, where a second image of a part of the target is acquired at a higher resolution to rule in ambiguous cases. Such a second look might generally not be possible in the gated viewing case. Instead, we might increase the accuracy by some third matching step. One thinkable way of doing this is by using a proper gated image sequence for building a 3D mesh structure and comparing it to some of the most likely of the CAD-models directly (see chapter 4 for an elaboration on 3D meshes or section 4.4.3, where a variation of such an approach is considered). This approach might be applicable under the assumption that more information can be extracted about the 3D structure of the target from the gated sequence, than is contained in the range resolved image (figure 3.16). The implementation of such a step would of course turn the entire algorithm implemented in this work into a pre-screener in itself.

If the accuracy of the implemented algorithm is not high enough for it to perform autonomously, its property to at least give good placings to the correct candidates renders it possible that it could serve successfully as a pre-screener in the way just described, or it may serve in multiple sensor systems where additional target recognition information is available (see [8]) from other systems.

3.9 Conclusions

This chapter has presented the outlines of some parts of a viable strategy for automatic target recognition. This method can function as a pre-screener for a more thorough algorithm in a completely autonomous target recognition algorithm, or as an additional information source in a multiple sensor system. In order to reach an implementation that will function in a realistic scenario, some aspects of the process must be subjected to further studies.

- The segmentation of the target from the gated sequence images should be made more robust and efficient.
- The problem of range resolving the object from proper gated sequences, and the properties of the generated object should be studied.
- Data from the elevation angle tests suggest that the candidate library be expanded by the addition of candidates at positive elevation angles in increments of 5^o.
- The model library should be completed with different articulations of the basic models.
- A new, more robust, way of solving the related problems of object alignment and mis-sampling should be considered.

Chapter 4

Adaptive 3D Mesh Structures

In the previous chapter, which treated target recognition from gated viewing data, the matching was carried out after making the models and data meet halfway by constructing range-resolved objects from both. Another option would have been to process the original gated viewing data into a mesh of the same form as the CAD-models in the model library, and to match them directly. As one might expect the matching of entire meshes to be a more computationally demanding approach than the matching of images, this method is probably likely to be of use in a final matching step after applying the method of chapter 3, to choose only between a small number of likely candidates. The concept of an extra, even more thourough, matching step, requires that more information about the 3D structure of the target can be extracted from the raw data, than what is already contained in the target object (figure 3.16). Whether this is the case or not remains to be investigated.

Since the gated image sequences from October 2002 did not give the desired range-resolution, no mesh structures were constructed from these. Rather, before the gated sequences became available, some investigations of the construction of mesh structures from fully range-resolved scanning laser radar data were conducted. These investigations led to the development of an algorithm for measuring the distance between a set of points and a surface. This algorithm proved to fit into the tryouts of a method for automatic target recognition from airborne 3D laser radar data, which is being developed at FOI. Thus, section 4.4 provides an overview of this method, while the basis for the MATLAB function developed in this work is presented in section 4.3. The sections 4.1 and 4.2 deal with the construction and reduction of triangular 3D mesh structures.

4.1 Definition of a Mesh

To somewhat formalise the discussion that will follow, a definition of what constitutes a mesh will now be given. Several definitions exist, see e.g. [9]. A definition applicable to this work is presented here. A mesh M is a piecewise linear surface. It is defined by a set of vertices $V = \{v_1, ..., v_m\}, v_i \in R^3$, and a set K, called a simplicial complex, which specifies the connectivity of the ver-

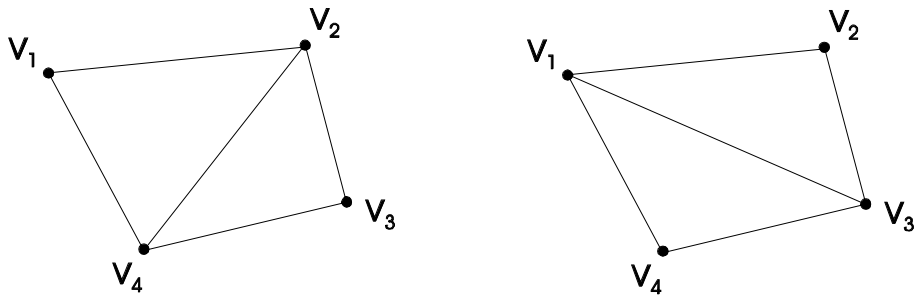


Figure 4.1: An example of a two triangular meshes with four vertices and two faces each. Face lists: left mesh $\{(v_1, v_4, v_2), (v_4, v_3, v_2)\}$, right mesh $\{(v_1, v_3, v_2), (v_1, v_4, v_3)\}$.

tices. The simplicial complex is further described in [9]. The connectivity of the vertices will most often be represented by a face list, which is best explained through an example. In figure 4.1 we have an example of two meshes which have identical vertices but differ in connectivity. Each of the two meshes has four vertices, which are points in R^3, but the face list for the leftmost mesh will be $\{(v_1, v_4, v_2), (v_4, v_3, v_2)\}$ (or any cyclic permutation within each face) and the face list for the right mesh of the figure is $\{(v_1, v_3, v_2), (v_1, v_4, v_3)\}$.

4.2 Mesh Construction from 3D Point Clouds

There are two desirable properties of a triangular mesh representation of a surface when used for target recognition. Firstly, the mesh should represent the true surface with as little error as possible. Secondly, the mesh should be as sparse as possible to reduce the computational loads when matching. In order to meet both of these demands at the same time, we need a structure that adapts to the shape of the target, making the mesh dense in areas of high detail and sparser in those of less detail. Basically, this goal can be achieved in either of two ways: The adaptive mesh can be constructed by starting from scratch and adding vertices to the mesh locally on the basis of level of detail. It can also be created by initialising a dense structure in order to catch the finer details of the surface, and then reducing this dense mesh in the areas where where the high density is not needed. After a literature study of [9], [10] and [11], the latter approach was chosen for implementation.

Specifically, the problem of building a mesh from a data-point cloud was studied. Figure 4.2 portrays a set of data points distributed in 3D space, as seen in the direction of measurement, colour-coded by range from the observer. The points in the figure were acquired by randomly sampling a larger set of points from a high-resolution laser radar scan by a factor of ten. The target was the same "T72 Kvarn" tank that was also used for constructing the CAD-model of table 3.1.

The first step in the process is to construct a dense mesh from the point set. Since most of the effort was put into solving the problem of collapsing an existing mesh, the method of constructing the starting mesh is, in its present

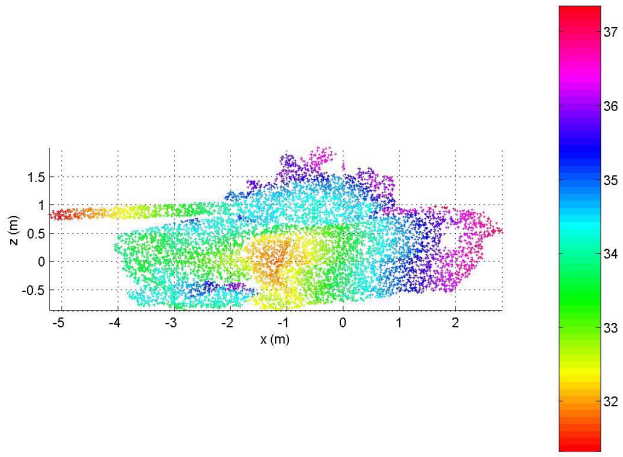


Figure 4.2: 8798 data points from a laser radar scan of a T72.

form, somewhat inefficient, albeit well-working. This part of the method will also have to be adapted to the type of input data, be it from gated viewing or 3D FPA or the like.

To the data points of figure 4.2, an artificial background is first added. The artificial background consists of new points which are added to the regions where no data exist in figure 4.2. The purpose of the artificial background is to enable us to interpolate the data into a range-coded image, with the aid of existing MATLAB functions. Onto this image, a dense rectangular (in the cross-range direction) grid of mesh vertex points is placed, and the range value of each is set to the value of the corresponding pixel in the generated range image. Around the edges of the target we will, as an artefact of the interpolation, get a number of vertex points which belong neither to the target nor to the artificial background. After removing all vertices which clearly are on the background, it is therefore necessary to make a search through all the remaining vertices and remove those which are not in the vicinity of any of the original data points. The remaining vertices are then connected to their neighbours so as to form a triangulated three-dimensional surface. A coloured and shaded version of a mesh constructed from the data points of figure 4.2 is shown in figure 4.4. We have thus constructed a dense mesh surface from a data point cloud. A different approach to surface reconstruction can be found in [11].

The mesh of figure 4.4 has 3441 vertices and 6400 faces, and the next step is to reduce this into a more manageable size for the future computations. In the sense of [9], an optimal mesh is a mesh which minimises an energy function that penalises both large numbers of vertices and errors in the surface representation (and an additional "spring" term whose purpose is to guarantee that a minimum exists). To produce such a mesh, the method described in [9] uses a combination of three types of transformation, namely edge collapse, edge split and edge swap operations. In [10], which deals with mesh simplification rather than

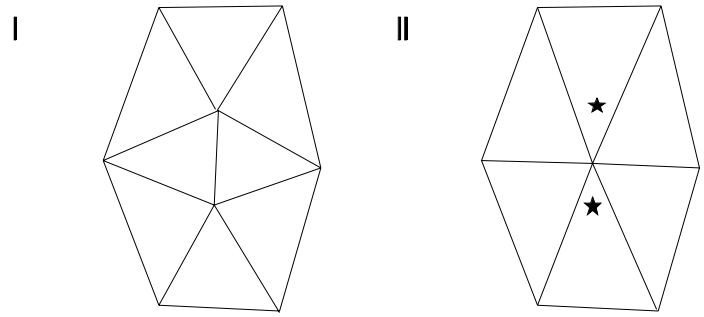


Figure 4.3: The edge collapse transformation. An edge in the left mesh is collapsed to produce the simplified mesh to the right. The positions of the removed vertices are marked with star-shaped polygons. If these vertices were part of the original mesh, they will be added to the set of free sample points.

optimisation, it is found that a single type of transformation, the edge collapse, is sufficient to produce adequate results. The principle of the edge collapse transformation is shown in figure 4.3.

A simplified version of the algorithm described in [10] was implemented in this work. The main principle of this mesh simplification algorithm is that edges of the mesh are collapsed in the order given by a priority queue, which is sorted so that the operation which would inflict the smallest error on the mesh is carried out first. The priority que is then updated before another edge is collapsed, and so on. Simplifications to the algorithm were possible to make because it was not necessary in this work to consider scalar attributes or discontinuity curves (such as texture changes) of the mesh, as is done in [10]. Also, in this work, no optimisation of the position of the new vertex after each edge collapse is made. The latter simplification also makes the regularising "spring" term redundant. The simplifications were made in order to produce a faster algorithm, and also for ease of implementation.

The error is measured as follows. At the start, the mesh is sampled in a number of points, to record the position of the original mesh. These sample points can be chosen in any number of ways. Here, the original vertex points have been used for this purpose. The total error of a certain mesh M is measured as

$$E_{dist} = \sum_i \min_{p_i \in M} (\|x_i - p_i\|^2) \tag{4.1}$$

where x_i are the sample points just mentioned, and $p_i \in M$ is a point on the mesh for each x_i. The position of p_i that satisfies the minimisation problem will be either that of the orthogonal projection of x_i onto one of the triangular faces of M or it will be on one of the edges of such a face.

Of course, before the mesh simplification starts, all sample points x_i will be on the mesh and the error will be zero. Before any edge collapse can take place, we need to initialise the priority queue. This is done by considering all edges in the mesh as candidates for collapse, and calculating the error that would be induced by each collapse. Since the error is zero from the start when all sample

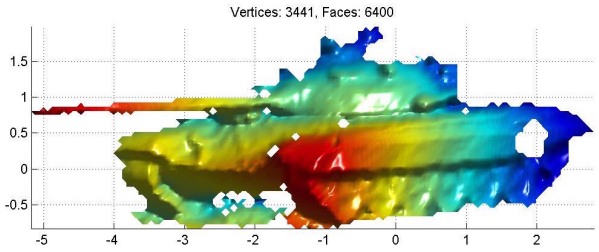


Figure 4.4: The dense mesh formed from the point cloud of figure 4.2. The mesh has 3441 vertices forming 6400 faces. The mesh is displayed shaded and in range colouring.

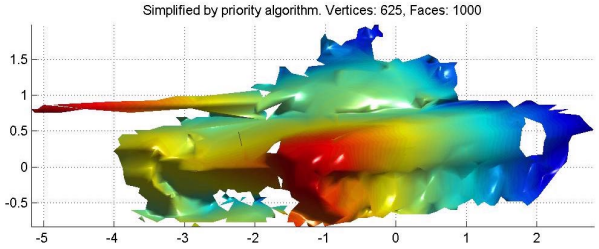


Figure 4.5: The mesh of figure 4.4 after simplification by the priority queue algorithm. The simplified mesh has 625 vertices, forming 1000 faces.

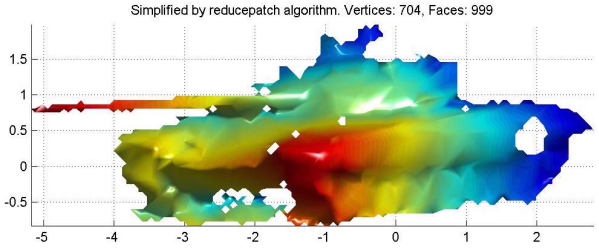


Figure 4.6: The mesh of figure 4.4 after simplification by the reducepatch algorithm of MATLAB. The simplified mesh has 704 vertices, forming 999 faces.

points are on the mesh, the sum of equation 4.1 will be made up of only two non-zero terms for each candidate edge collapse. These two terms correspond, of course, to the two vertices that would be lost from the mesh in the operation (see the right part of figure 4.3). The priority queue is formed by sorting the edges by their corresponding error. Actually, we never need to sort the queue because we are only interested in the lowest value, but the sorted queue is nevertheless a useful construction for understanding the method.

Once the initial priority queue has been constructed, edge collapse starts to take place. After each collapse, the priority queue must be updated with new error values for all the edges that were changed as a result of the collapse. As more and more edges are collapsed, the number of free sample points (sample points that no longer constitute vertices of the mesh) will increase and so would the number of non-zero terms in the sum of equation 4.1. After a while, this would slow down the updating of the priority queue considerably, since the error value for a number of edges has to be re-calculated after each collapse.

To save time, the error value for each possible edge collapse is recalculated using only the sample points in a local neighbourhood. This is possible because the priority queue incorporates only error differences $\Delta E = E_{dist} - E'_{dist}$, rather than total error of the entire mesh. The sample points in a local neighbourhood of a collapse are found by means of an association list. When new sample points (vertices of the original mesh that have been set free by a collapse, see figure 4.3) are added to the set of all free sample points, an association will be created between each new point and the nearest face of the mesh. When a face disappears from the mesh in a collapse, we also need to reassociate all of its associated points with faces of the new mesh. When we recalculate the error value for a certain edge collapse, the local neighbourhood is defined as all faces that that have one or two vertices in common with the edge in question, and all sample points associated with any of these faces (plus any new sample points, created by the collapse of an edge of the original mesh).

When the priority queue has been updated, the best edge collapse is performed and the process is repeated until a predefined number of faces remain in the mesh. In figure 4.5, we can see the result of the mesh of figure 4.4 having been simplified from 6400 to 1000 faces (and from 3441 to 625 vertices) by the priority-queue algorithm. The priority-queue approach to mesh simplification is one of several possible approaches. In MATLAB, the "reducepatch" function works as to reduce "the number of faces of the patch ... while attempting to preserve the overall shape of the original object". The algorithm behind the "reducepatch" function is not specified in the MATLAB documentation, and the source code of the .mex-file has not been available to the author of this report. Also, experience from using the "reducepatch" function in this work has shown that it does not always produce the wanted results for certain types of meshes, which motivates the implementation of the more stable priority-queue algorithm. A benefit of the "reducepatch" function is, on the other hand, that it executes much faster than the priority-queue function. This difference is partly due to the "reducepatch" function using compiled code, but this probably does not account for the entire difference. In figure 4.6, the mesh of figure 4.4 has been simplified from 6400 to 999 faces (and from 3441 to 704 vertices) by the "reducepatch" function.

4.3 Surface Error Measurements

A spin-off from the implementation of the priority-queue algorithm is that we now have a function that measures the shortest distance from a point (or many points) to a mesh in R^3. Since this function has been used in other work than in the mesh simplification algorithm (see section 4.4.3), it is given a more detailed description here.

A triangular face is defined by its three vertex points. Alternatively, we can represent it by one vertex point and two vectors. This is shown in figure 4.7, where the vertex is denoted P and the vectors $\overline{v}_1$ and $\overline{v}_2$. The two vectors span a plane in R^3. The equation of this plane can be written in normal form $Ax + By + Cz + D = 0$, where the normal of the plane $\overline{n} = (n_x, n_y, n_z) = (A, B, C)$, and $D = -\overline{n} \cdot P$. When a set of points and a mesh are sent to the function for distance measurements, the plane coefficients A, B, C and D are calculated once and for all for all the triangles of the mesh, because these will be used many times in the process. A normal vector of unit length is obtained through (in the notation of figure 4.7) $\overline{n} = (\overline{v}_1 \times \overline{v}_2)/\left\|\overline{v}_1 \times \overline{v}_2\right\|$.

The calculation of the distance between a point S and a triangular face starts with the orthogonal distance from the point to the plane, which is given by $d_{plane} = \overline{n} \cdot S + D$. Once d_{plane} has been determined, it is easy to find the position of the projection S' of S onto the plane. In figure 4.7, the projection is inside the triangle and the point-to-triangle distance is simply d_{plane}. This will not always be the case for all point/triangle pairs, so we will need a way of distinguishing between different situations. Since the vectors $\overline{v}_1$ and $\overline{v}_2$ span the plane, it is possible to write the projection of S as a linear combination of these: $S' = P + s\overline{v}_1 + t\overline{v}_2$. Since S', P, $\overline{v}_1$ and $\overline{v}_2$ are known, we solve the equation system to obtain the values of the parameters s and t. Now, if $s, t > 0$ and $s \leq 1 - t$, we know that the projection is inside the triangle. If it is not, we use the signs of s and t to determine if the projection is in area 1, 2 or 3 of the plane as defined in figure 4.7. The point-to-triangle distance will now be the distance between S and the triangle edge which is on the border of the area that the projection S' is in. One or two scalar products are necessary to determine if the point S is closer to any of the two edge endpoints than to any point on the edge. The point-to-triangle distance is then evaluated as either the point-to-point or point-to-line distance.

One possible application of the points-to-mesh distance measuring function is the measurement of the distance, or error, between two meshes. Figure 4.8 shows the result from supplying the vertex points of the mesh in figure 4.5 (the mesh simplified by the priority-queue algorithm) and the entire mesh of figure 4.6 (the mesh simplified by the "reducepatch" algorithm) as input to the function. We see that the meshes simplified by the two methods differ by a few cm in most places, but by up to more than 8 cm in some. A more robust measure of the difference between meshes can be obtained by considering both the backward and forward distances, as is done in [12].

4.4 ATR from Airborne 3D Laser Radar Data

Scanning 3D laser radar data differ from gated viewing data in that we are no longer dealing with images, but rather a 3D data-point cloud that is range-

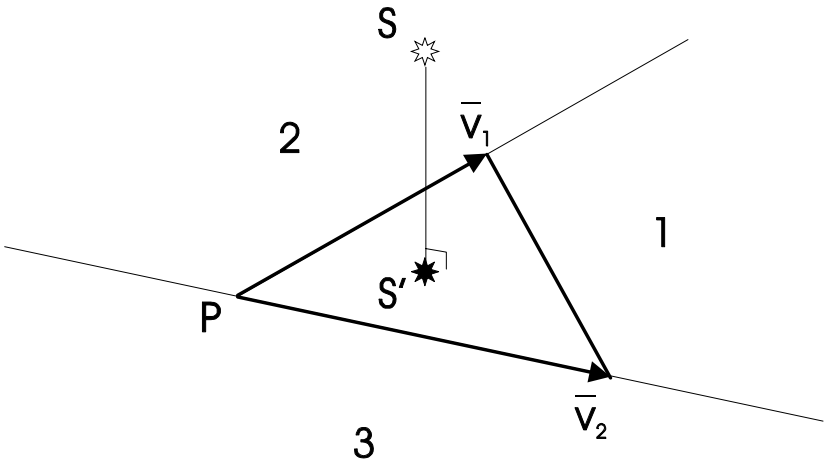


Figure 4.7: Measuring the distance from a point to a triangular face.

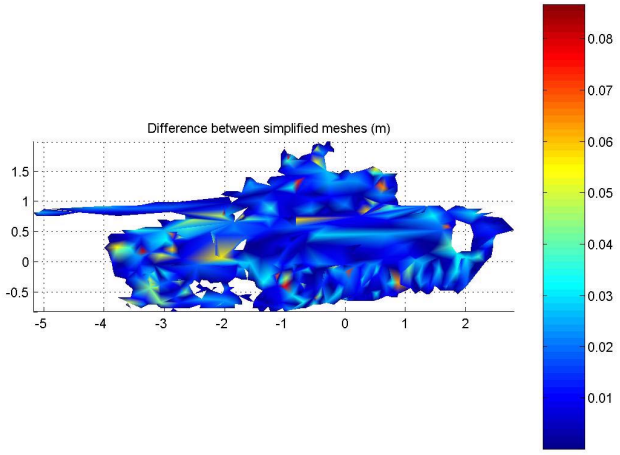


Figure 4.8: The measured difference between the meshes of figure 4.5 and figure 4.6.

resolved already from the acquisition. We might also expect aspect angles to differ from those of the gated viewing case. In this section, the algorithm described in section 4.3 will be used in one of the steps of a possible scheme for automatic target recognition from such data. This step is presented in section 4.4.3. Some of the work done on airborne 3D laser radar data analysis by Christina Grönwall (earlier Christina Carlsson) at FOI [3][8][13], leading to the area of contribution of this work, is briefly presented below. It should be noted that the method is still under development. The basic idea of the method is to first estimate the orientation and articulations of the target by means of geometric fitting, and then making direct measurements of the distance between the point cloud and the CAD-models of a model library.

4.4.1 Data Acquisition

The principles of the ATR method will be illustrated using scanning 3D laser radar data, acquired with the helicopter-based TopEye system that was described in section 2.2. In figure 4.9, a visual image of an area at the Kvarn artillery range, where TopEye measurements were made in August of 2000, can be seen. Three vehicles were positioned at the site; one BTR 70, one TLB and one T72. More information about the circumstances of the measurements can be found in [13].

The resulting data from an overhead pass with the TopEye system are presented in figure 4.10. The 3D data are here projected onto a 2D image as seen from above, with each data point greyscale-coded by its height-value (z-value). Intensity information was also collected by the measurement system but is not shown here. Typically, around 100 data points are acquired from each target at this particular combination of flight altitude, speed and vehicle size.

Figure 4.9: Visual image of part of the area covered by the TopEye measurements.

4.4.2 Orientation Estimation

The target data points are separated from those of the surroundings on the basis of their z-values. The orientation of the target is then estimated through geometric fitting of rectangles to the data. An efficient method of doing this,

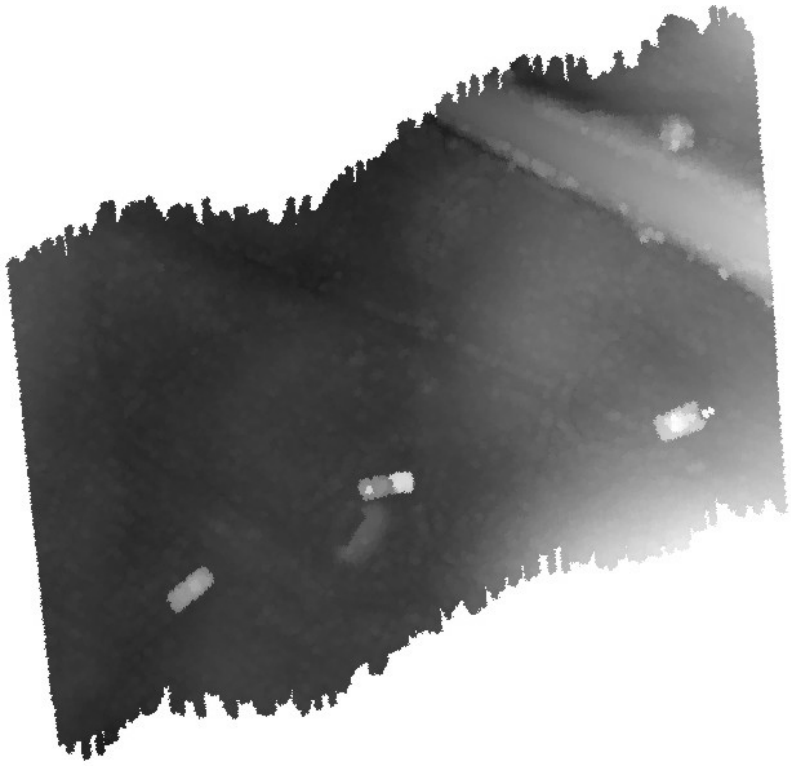


Figure 4.10: Height data from TopEye scan of the vehicles in figure 4.9. Data are scaled from black (ground) to white (height).

requiring only a limited number of rectangle orientation hypotheses to be tested, is presented in [3]. An illustration of such a minimum area rectangle fit is illustrated in figure 4.11. Rectangle fitting to target data points alone will tend to underestimate the size of the target, which is compensated for using least-squares methods to adjust the rectangle to both target and ground data. In figure 4.11 the rectangle is fitted to the data from a top view, and rectangles are also fitted in a similar fashion from the side and frontal views, creating the smallest bounding box around the target data.

Now, the basis of the next step is the notion that a target, e.g. a tank, can be segmented into a few basic constituent parts. In the case of a tank, these can be its body, turret and barrel. The articulation of the tank is estimated by segmenting the target and considering the orientation of each of its parts separately.

The segmentation is accomplished by calculating the shortest distance from each data point to the bounding rectangle. Data points that lie far away from the rectangle sides are considered as possible candidate locations for a partitioning of the data. Figure 4.12 shows the use of this principle for segmenting the T72 data into three data sets corresponding to its body, turret and barrel.

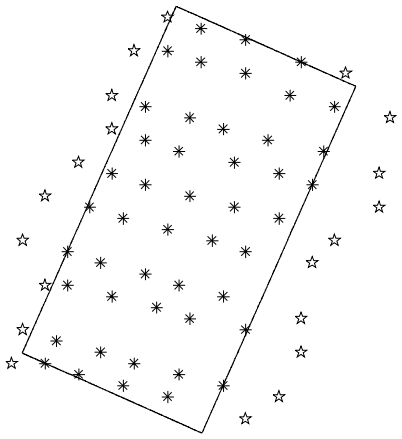


Figure 4.11: Example of rectangle fitting to estimate the orientation of a terrain vehicle.

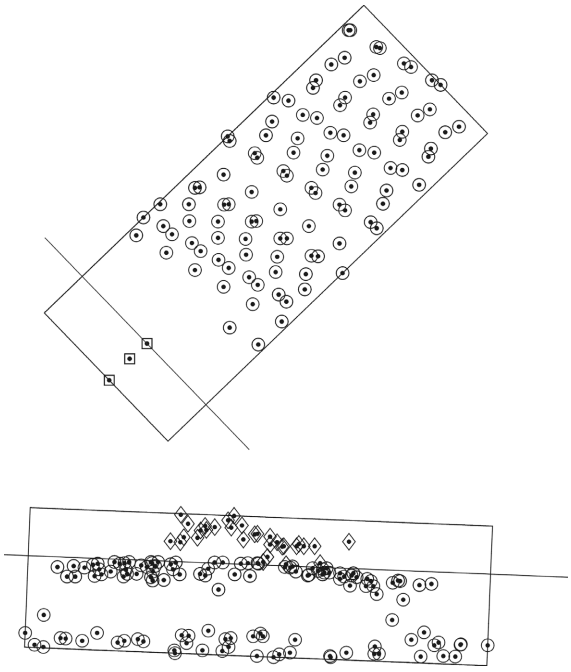


Figure 4.12: Segmentation of the T72 data into barrel, turret and body by considering data points that are distant from the bounding rectangle as candidate locations for partitioning. Top: top view. Bottom: side view.

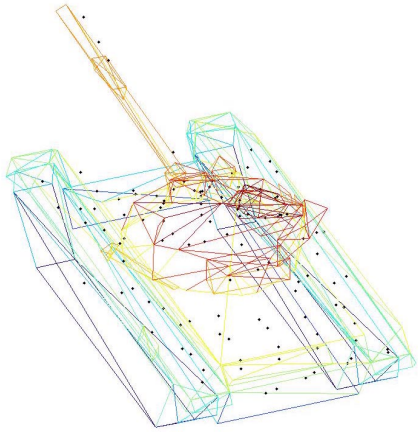


Figure 4.13: Matching by measuring the distance between TopEye data and the mesh of the CAD-model directly. A reduced CAD-model, compared to the one in 3.1, is displayed.

4.4.3 Surface Matching

Once the orientation and, in applicable cases, articulation of the target have been estimated, the CAD-models of the library are rotated and adjusted so as to fit the estimated parameters. The matching score for each model is then obtained by directly measuring the distance between the data points and the mesh of the model. (The distance measure might be, for example, the mean of all point-to-mesh distances). The principle is illustrated in figure 4.13, where the TopEye data from the T72 tank is displayed together with the corresponding CAD-model. The CAD-model shown in this figure is a coarse model with a reduced number of faces compared to the one in table 3.1. The contribution of this work is the MATLAB function that measures the distance between each data point and the triangular mesh surface of the model. This function is given a more thourough presentation in section 4.3.

4.5 Discussion

This chapter has presented two methods for the purposes of mesh construction and surface error measurements. Both methods suffer from a common problem, which is the lack of computational speed for real-time applications. This especially applies to the adaptive mesh construction, and mostly to the mesh simplification part of this process.

While giving satisfying results, the mesh simplification algorithm was based on the algorithm of [10], which was designed for off-line use in computer graphics and not for real-time applications. The execution times of this model algorithm, as presented in the mentioned article, were to long to be acceptable, which is why simplifications were made before it was used here. Despite these simplifications, the speed of the algorithm implemented in this work did not reach that of its

more complicated predecessor, but was in fact several times slower. This may be due to the different programming languages of the implementations. The author of [10] used C/C++, while the programming work that this report is based on was carried out in MATLAB.

The "reducepatch" function of MATLAB proved to be fast, but was unfortunately not robust to meshes with certain features, such as those of a type of stair-shaped meshes that can be obtained from gated viewing sequences. As to the first part of the adaptive mesh scheme, which is the construction of the dense mesh, which precedes the simplification, was never particularly optimised, and it should be possible to bring its execution time down to acceptable levels.

The surface error measuring function does not execute as slowly as the simplification algorithm, but when comparing two meshes of the sizes of those in figures 4.5 and 4.6 (about 1000 faces each) it is still a matter of minutes on a normal PC. Again, the function could probably be speeded up through implementation in C/C++. Also, the surface error function was constructed to deal with the small local neighbourhoods of edge collapses. When applied to larger meshes and point sets, it has the drawback of having to test all faces of the mesh for every sample point in order to find the minimum. A remedy for this is suggested in [12] where a partitioning of the point and mesh data in R^3 was made in such a way as to reduce the amount of testing. Still, the version of the surface error function implemented here may be of use when dealing with fewer points, such as when dealing with TopEye data in the way discussed in section 4.4.

The basic outlines of a method for automatic target recognition from airborne 3D laser radar data were also presented in this chapter. The method is under development, and the surface error measuring method developed in this work has been useful in the tryouts. The relatively small number of target data points from a typical TopEye scan of a vehicle has a positive effect on the execution speed of the surface matching algorithm. For real-time applications with a large model library, a faster implementation will nevertheless be required. Compared to the gated viewing case, the number of candidates will be significantly reduced. Since the approximate orientation of the target will be known from the previous step in the algorithm, we only need one or a few comparisons per model, instead of matching the data with all possible combinations of articulation and orientation for every model. Some information concerning the target type can also be provided already by the articulation estimator, e.g. it might determine if the target has a turret or not, which would reduce the number of candidates, and thus the computational load, even further.

Chapter 5

Discussions

5.1 Conclusions

The target recognition algorithm presented in chapter 3 shows promising results. The algorithm can function stand-alone, or as a part of a multi-sensor system for increased accuracy. All steps of the implemented algorithm can be improved. The robustness tests should be considered as a first attempt at evaluating the algorithm, and more realistic tests are needed to properly quantify the results.

The mesh construction and reduction algorithm introduced in chapter 4 gives satisfying results, but executes to slowly for real-time application. For sufficiently small data sets, the surface error measuring function is useful in testing a method for automatic target recognition from airborne scanning laser radar data.

5.2 Future Work

The suggested future work concerning the ATR method in chapter 3 can be divided into the two categories of improving the existing algorithm, and the investigation of related subjects. Concerning the direct development of the implemented algorithm, it is suggested that:

- A transformation-based motion compensation algoritm [6], TBMA, be incorporated in the data pre-processing step.
- The target segmentation method be made more robust and efficient.
- The model library be expanded to contain a representative selection of vehicles, as well as different articulations of these models.
- The problem of object alignment be given a more robust solution, while maintaining computational efficiency.
- Large-scale robustness tests using realistic data, e.g. from a gated viewing simulation program, be performed.

Concerning related investigations, work should be put into:

- Testing the range-resolution of the gated viewing system in field trials, and developing theoretical models for the achievable resolution.
- Acquiring gated image sequences of vehicle targets using the capability of the gated viewing system of gate stepping down to 1 ns, and examining the possibilities of constructing 3D mesh structures from these data.
- Examining the order of range and cross-range resolutions required for robust target recognition.
- Following the development of 3D FPA technology and optimising ATR methods for these data.

The surface error measuring function presented in chapter 4 can be made more efficient, for example by considering data partitioning to reduce the number of point-to-triangle distance calculations for each sample point..

Appendix A

Source Code Documentation

This work has primarily been concerned with evaluating the performances of different algorithms, and not with the development of software for use in a real system. However, some MATLAB code has been produced that may provide a basis for further studies. The most important ones among these files are commented on in this appendix.

Many of the files are in the form of MATLAB scripts. This form was chosen for increased transparency during the development, but in order to avoid unwanted side effects in the workspace when using them together with new matlab code, it may be appropriate to convert these into matlab functions. This will be easy to accomplish in most cases.

A.1 Pre-Processing Files

In order to perform target recognition from a gated sequence, we run three matlab scripts in succession. These scripts are presented below, along with a script for generating a candidate library from a set of CAD-models.

gv2range.m

This matlab script converts the data in a gated sequence to a range-resolved object, which can then be sent to the matching algorithm. At the head of the script the user specifies the path of the gated images, and the relevant parts of the filenames. The image filenames should follow the same convention as those saved from the Älvdalen 2002 series of sequences. The user must also specify which images of the sequence contain the object and which ones contain the background. The cross-range resolution of the wanted object image must be set to match that of the candidate library (0.2 m/pixel during all of this work). To obtain a correctly scaled object, the range to the target, the range step of the sequence, the focal length of the camera and the size of the sensor element should also be entered.

The script first produces mean images of the target and its background. These images are thresholded according to the method described in section 3.4,

to get a silhouette image. The script then steps through the image sequence containing the target again, to obtain the time/range-variations of all the pixels in the silhouette image. This information is padded and low-pass filtered, and the sequence is then stepped through a third time to find the first frame of the sequence where the intensity of each pixel exceeds its threshold value. The resulting range image is then sampled to the specified cross-range resolution, and finally the range-coordinate axis is transformed. The object image for input to the matching algorithm is now contained in the variable "dist_image".

When performing automatic target recognition from a sequence of gated images, this will be the first script to run.

A.2 Recognition Files

librarybuild.m

This file creates a candidate library containing 3D-images and vertical and horizontal projections from the library of all models with subdirectories in the directory specified by 'librarypath'. Each model subdirectory must contain a 'vertices.dat' ascii vertex list, a 'faces.dat' ascii face list and a 'rotations.dat' ascii file specifying the rotations needed to transform the coordinate system into x-right, y-forward and z-up. The directory 'librarypath' must not contain other subdirectories than those specified above. This script will create a 'features.mat' and a 'dimages.mat' file in each model subdirectory. These data files will then be used by the pre-screener and surface and boundary matching scripts, so it is necessary to run this file if the model library is updated. Required files are projpatch.m, point_vect.m, face_check_multi.m, which_face_check.m and which_face_check.dll written by Tomas Carlsson.

featurescreen2.m

In the ATR process, this will be the file to run after gv2range.m. This is a script that may easily be converted into a function, but the present implementation requires the target object image to be contained in the variable "dist_image". The script creates vertical and horisontal projection vectors from the target object, and correlates these against those of the candidate library. After executing this script, the three columns of the matrix "Mc" will contain correlation score, model number and azimuth angle number (1-72 corresponds to 0^o-355^o in 5^o increments) for each candidate, sorted in order of decreasing scores.

In the file head, the user specifies the path of the candidate library, and the weight of the horizontal projection score in relation to the weight of the total score of 1 (0.2 has been used in this work).

sbmatch.m

After the pre-screening of featurescreen2.m, this file should be run in order to perform the surface and boundary match. This is again a MATLAB script that can be converted into a function. The present implementation requires the sorted list from the pre-screener in the matrix "Mc" and the target object in "dist_image". In the header of the script, the user must specify the number of candidates to keep after the pre-screening. The surface match threshold τ_{srf}

can also be varied here (0.2 m was used in this work). Each row of the resulting matrix "Fit" will contain the weighted surface and boundary score, the model number, the azimuth angle (in degrees), the surface score and the boundary score, respectively. The rows are sorted in order of decreasing weighted score.

A.3 Mesh Files

gv2cloud.m

Esentially the same script as gv2range.m, but will also create a dense point cloud from the (unsampled) range image. In addition, a second part of this file can be activated by removal of the "return" command. This second part will triangulate a mesh surface directly from the range image (see make_mesh2.m).

build_mesh.m, make_mesh2.m

function[Vertex_list,Face_list]=build_mesh(X,Y,Z,I,Bsize)

The function build_mesh.m does the same job as the script make_mesh2.m but is in the form of a MATLAB function instead of a script.

Both files require the functions background_pad.m and make_image.m.

The file constructs a dense mesh from a data point cloud. In the script make_mesh2.m, the user specifies the path and name of the .mat-file containing X, Y and Z position vectors and intensity vector I. The value of the integer "Block_size" (argument Bsize for build_mesh.m) gives the spacing of vertex points as measured in pixels of the range image (see make_image.m). A small value thus gives a dense mesh.

background_pad.m

function[X,Y,Z,I]=background_pad(X,Y,Z,I,NX,NZ,edgebuffer)

Pads empty spaces in the 2D projection of a point cloud (represented by coordinate and intensity vectors X,Y,Z and I) with artificial background points. The default depth of the background is 1.75 times the depth of the data point set. The NX and NZ arguments specify the resolutions along the two image axes. A larger value gives a higher resolution. The resolution should not be set too high compared to the density of the projected data point set, as this may create false "holes" in the set.

make_image.m

functionImagematrix=make_image(X,Y,Z,Nnewgrid_x)

This function interpolates a range image from a set of data points.

meshsimplify.m

function[Vlist_out,Flist_out,Delta_E]=meshsimplify(V_list,F_list,Nc)

Requires mesherror.m and planecoefficients4.m.

The function meshsimplify.m simplifies the triangular mesh specified by vertex and face lists V_list and F_list to Nc faces by means of the priority-queue algorithm described in chapter 4. The vector Delta_E contains the error induced by each operation.

mesherror.m, mesherror2.m

function[Evector,Fvector]=mesherror(samples,faces,vertices)

mesherror2.m is the same function but also displays a waitbar.

Requires planecoefficients4.m.

The function measures the distance from each samplepoint (with the coordinates specified in the in the $N_{points} \times 3$ matrix "samples") to the mesh defined by the face and vertex lists "faces" and "vertices". The function returns the vector "Evector", which contains the squared distance from each sample point to the mesh, and the matrix "Fvector" which is a face list containing the closest face of the mesh corresponding to each sample point.

planecoefficients4.m

function[A,B,C,D]=planecoefficients4(v1,v2,v3)

The function returns normalised normal form coefficients for the plane spanned by the three points v1, v2 and v3.

Bibliography

[1] Lena Klasén, Ove Steinvall, Göran Bolander, and Magnus Elmqvist, "Grindad avbildning-inledande prov vid långa avstånd," Tech. Rep. FOI-R–0302–SE, Defence Research Agency, Sweden, 2001.

[2] Ove Steinvall, "Laser based (flash) 3-d imaging for target recognition and terminal guidance: Technical overview and research problems.," in manuscript.

[3] Christina Carlsson, *Vehicle Size And Orientation Estimation Using Geometric Fitting*, Ph.D. thesis, 2000, Linköping Studies in Science and Technology, Lic. Thesis No 840, 2000.

[4] "http://www.3dlasermapping.com/topeye.htm," cited 2003-01-17.

[5] Q. Zheng, S.Z. Der, and H.I. Mahmoud, "Model-based target recognition in pulsed ladar imagery," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 565–572, April 2001.

[6] Lena Klasén, *Image Sequence Analysis of Complex Objects - Law Enforcement and Defence Applications*, Ph.D. thesis, 2002, Linköping Studies in Science and Technology, Thesis No 762, 2002.

[7] Tomas Carlsson, Ove Steinvall, and Dietmar Letalick, "Signature simulation and signal analysis for 3-d laser radar," Tech. Rep. FOI-R–0163–SE, Sensor Technology, 2001.

[8] Erland Jungert and Christina Grönwall, "ΣQL ett beslutsstöd för måligenkänning i en multisensormiljö," Tech. Rep. FOI-R–0692–SE, Avdelningen för ledningssystem, 2002.

[9] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, "Mesh optimization," in *ACM Computer Graphics Proceedings, Annual Conference Series, Siggraph93*, 1993, pp. 19–26.

[10] Hugues Hoppe, "Progressive meshes," in *ACM Computer Graphics Proceedings, Annual Conference Series, Siggraph96*, 1996, pp. 99–108.

[11] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, "Surface reconstruction from unorganized points," in *ACM Computer Graphics Proceedings, Annual Conference Series*, 1992, vol. 26, pp. 71–78.

[12] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the Hausdorff distance," in *Proceedings of the IEEE International Conference on Multimedia and Expo*. 2002, vol. 1, pp. 705–708, IEEE Computer Society.

[13] Christina Grönwall, "Mätningar med flygburet multisensorsystem, mätrapport från fordonsplatser i kvarn och tullbron," Tech. Rep. FOI-D–0060–SE, Sensorteknik, 2002.