Universidade de Aveiro Departamento de Engenharia Mecânica
2013

Daniel Torres Couto Coimbra e Silva

Deteção e Segmentação de Alvos em Ambiente de Estrada usando LIDAR

LIDAR Target Detection and Segmentation in Road Environment

Daniel Torres Couto Coimbra e Silva

Deteção e Segmentação de Alvos em Ambiente de Estrada usando LIDAR

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President

Prof. Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Doutor Cristiano Premebida
Investigador da Universidade de Coimbra - Faculdade de Ciência e Tecnologia

Prof. Doutor Vítor Manuel Ferreira dos Santos
Professor Associado da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Aqui deixo o meus mais sinceros agradecimentos a todas as pessoas que me ajudaram na realização deste projeto. Ao professor Doutor Vítor Santos pelo seu acompanhamento, disponibilidade e motivação transmitida. Ao Jorge Almeida pelo seu apoio e conhecimentos transmitidos essênciais para a concretização deste trabalho. Ao Pedro Cruz pelas dicas preciosas e pela sua alegria contagiante. Aos meus colegas de curso pelos momentos partilhados, gargalhadas, grande companheirismo, entre-ajuda e boa disposição nestes últimos cinco anos. Aos meu grupo de amigos do BLC pelos longos anos de amizade. Aos meus pais, avós e irmão pelo seu apoio incondicional e por estarem sempre presentes. Um obrigado especial à Mariana Poço, por estar sempre do meu lado principalmente nas alturas menos boas. Obrigado por tudo Mariana.

Palavras-chave AtlasCar, Segmentação, Sensor Laser 2D, Medidas de desempenho, ROS

Resumo Neste trabalho é feita uma avaliação exaustiva e comparativa de diversos algoritmos de segmentação de dados laser 2D em ambiente de estrada. Numa primeira fase, os algoritmos de segmentação são implementados usando o ambiente ROS; estes algoritmos têm a função de juntar pontos adquiridos pelo laser e agrupar os pontos de acordo com as suas propriedades espaciais e que idealmente irão pertencer ao mesmo objeto. Tendo cada algoritmo pelo menos um parâmetro variável na sua condição de separação, um dos objetivos do projeto é determinar o seu valor optimo em várias situações de estrada. A etapa seguinte foi definir um Ground-truth: diversos laser scans foram manualmente segmentados. Por fim, é feita a comparação entre os resultados dos algoritmos com o Ground-truth, testando assim a validade de cada algoritmo. Com o intuito de se realizar uma avaliação quantitativa, foram criadas seis medidas de desempenho da segmentação dos algoritmos que penalizam casos de má segmentação.

Keywords AtlasCar, Segmentation, Laser Rangefinder, Performance measures, ROS

Abstract In this project a comparative and exhaustive evaluation of several 2D laser data segmentation algorithms in road scenarios is performed. In a first stage, the segmentation algorithms are implemented using the ROS programming environment; the algorithms are applied to the raw laser scan data in order to extract groups of measurement points which share similar spatial properties and that probably will belong to one single object. Each algorithm has at least one threshold condition parameter that is configurable, and one of the goals is to try to determine the best value of that parameter for road environments. The following stage was the definition of the Ground-truth where multiple laser scans were hand-labelled. The next step was the comparison between the Ground-truth and the segmentation algorithms in order to test their validity. With the purpose of having a quantitative evaluation of the methods' performance, six performance measures were created and compared.

Contents

List of Tables

List of Figures

Nomenclature

ABD *Adaptive Breakpoint Detector*

BT *Best Threshold*

DS *Dietmayer Segmentation*

GT *Ground-Truth*

KFBD *Kalman Filtering Breakpoint Detector*

LIDAR *Light Detection and Ranging*

MS *Multivariable Segmentation*

PM *Performance measure*

ROS *Robot Operating System*

SA *Santos Approach*

SNN *Spatial Nearest Neighbour*

SS *Simple Segmentation*

TOF *Time of Flight*

Chapter 1

Introduction

1.1 The ATLAS Project

ATLAS is a project created and developed by the Group of Automation and Robotics from the Department of Mechanical Engineering of the University of Aveiro, Portugal. This project took its first steps in 2003 with the purpose of competing in autonomous driving competitions taking place at the Portuguese National Robotics Festival. Since then, several robots of the ATLAS series were developed (figure 1.1) and had a continuous success in those competitions and won many prizes.

Figure 1.1: ATLAS robots used in the autonomous driving competitions. [1]

With the experience in autonomous navigation in controlled environments acquired in robot competitions, the project team decided to go to the next level: Real road environment with a real scale model, and so the *AtlasCar* was born.

The *AtlasCar* is a prototype whose purposes are the research and the development of new advanced driver assistance systems; improve the safety of those systems so they can be ultimately incorporated lately in real vehicles and the study of new challenges in the autonomous driving domain. To achieve that, a *Ford Escort* model was equipped with several sensors which are meant to perceive the surrounding environment and the interior of the car and it was mechanically modified to be able to perform some operations (parking for example) without requiring the presence of a driver (figure 1.2).

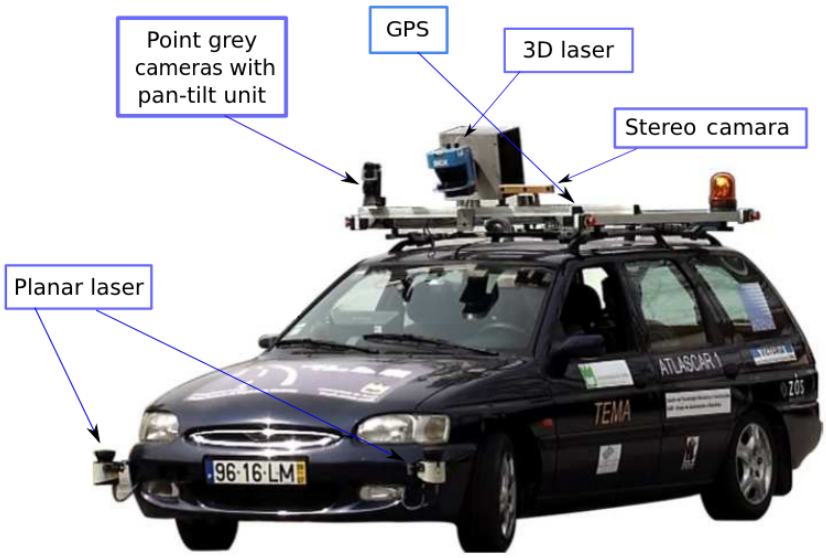


Figure 1.2: AtlasCar, a Ford Escort SW equipped with the sensors. [1]

The Sensory units mentioned above include:

- **Stereo camera** - Capable of transmit RGB and disparity images with a maximum rate of 16 fps rate. The images are then processed by a computer and 3D representations can be generated.
- **Foveated vision** - This vision system is composed by two similar cameras but with different viewing angles and focal distances. One camera has a wide angle to provide a full vision of the surrounding environment and the other is used to provide greater detail about a certain obstacle or region of interest. This system is very useful to test vision based attention mechanisms.
- **3D laser** - A custom sensor based on a commercial 2D range finder mounted on a rotative platform. This way it is possible to generate 3D laser scans that can be used to determine geometrical properties of the environment.
- **2D laser** - The car has 2 planar lasers in the front bumper and one on the roof. The primary function of the lasers on the front bumper is to detect obstacles in front or on the sides of the vehicle if the obstacles stand within the lasers detection angle. The laser on the roof of the car is directed towards the road in order to make a thee-dimensional reconstruction of the environment on the fly. More information of these lasers is given in the section 1.4.
- **IMU** - The IMU (*Inertial Measurement Unit*) uses a combination of accelerometers and gyroscopes to measure and report the car's motion state: velocity, orientation and gravitational forces.

For more information about the *AtlasCar* project, feel free to check the official site: http://atlas.web.ua.pt/atlascar.html

1.2 Project context

This work is inserted into the ATLAS project of the Department of Mechanical Engineering of the University of Aveiro in the context of developing advanced perception systems. This kind of systems is very common and has a important role in the control of autonomous vehicles.

The matter relates to the mobile object tracking more specifically in the segmentation process which is the primary stage of any tracking algorithm. Multi target tracking presents great importance in several areas: In mobile robotics, with the application of this kind of algorithm it is possible to estimate the time to collision with surrounding objects and avoid dangerous situations; the estimation of the time to collision is crucial for autonomous navigation (path planning) and can be used for driving assistance systems. The tracking systems can also be applied in buildings for personal access control applications or even monitoring the movement of people in crowded spaces like shopping centres or airports.

A tracking algorithm has the following stages:

- Segmentation - In this stage, the scanner points are grouped according to which object they are part of; this is the primary and also a decisive stage as failures during this stage will strongly affect all the subsequent phases; furthermore, errors in this stage do not have a easy retrofit correction solution.
- Data association - In this stage it is determined which current measurements correspond to a tracked object.
- Motion Estimation - Tracked objects motion is calculated: position, velocity, acceleration or even turn rate.

In conclusion, the segmentation stage is a really important stage in tracking purposes and it will be the "core" of this project.

1.3 Objectives

This project aims to perform a comparative and exhaustive evaluation of different segmentation algorithms in several road scenarios. The comparison is made through several developed performance measures in order to inquire which method performs the best segmentation possible.

1.4 LIDAR general description

LIDAR is the acronym for Light Detection and Ranging (also know as LADAR) is an optical position measurement sensor commonly using infra-red light to measure distances to objects. Using the Time Of Flight (TOF) measurement principal: A laser pulse is sent out by the sensor and is reflected by objects in the environment; the pulses that are reflected are detected by the sensor; the time delay between the laser pulse transmission

and its detection by the sensor (after being reflected) is used to calculate the distance of the object from the sensor (object's range). The LIDAR system is basically constituted by a laser scanner and a rotating mirror (figure 1.3), the mechanical rotation of the mirror causes the laser beam to disperse in different directions allowing the construction of a 2D map of the environment. By adding an additional rotating axis, it is possible to obtain a 3D representation of the surrounding environment.

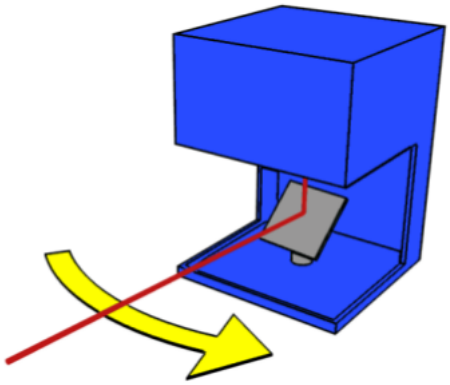


Figure 1.3: LIDAR operation: laser beam reflected by the rotating mirror. [2]

Although the rotating mirror moves continuously, the measurements are made at a discrete angle increment which is called angular resolution. The figure 1.4 shows the beams send out by the LIDAR in sequential pulses in different directions, the beams are angular distanced by the angular resolution value; the laser beams intercepted by an object are reflected back to the sensor, the beams not interrupted will be lost.

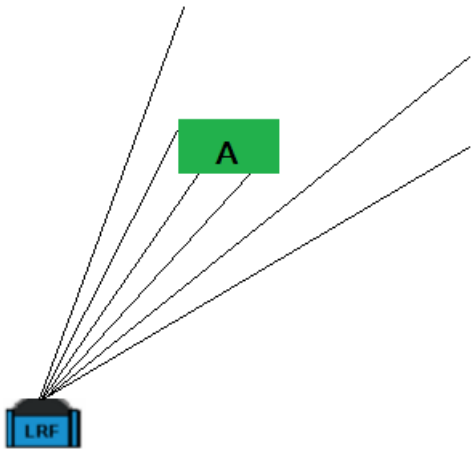


Figure 1.4: Angular resolution effect. [3]

A higher laser power and lower beam diverge is desirable because it can result in a higher detection range and better outline definition of the detected objects; however the laser cannot be harmful to the human eye so, in robotic applications with direct contact with persons, the laser beam must have a high divergence and low power.

Table 1.1: Sick LMS111 specifications. [4]

Max. range	50 m
Range at 10% reflectivity	10 m
Scanning angle	270^o
Angular resolution	0.5^o
Statistical error	± 40 mm
Systematic error	± 30 mm
Data interface	Ethernet 100 Mbit TCP/IP, UDP; RS-232; CAN
Supply voltage	10.8 V to 30 V DC
Dimensions (W x H x D)	102 mm x 162 mm x 105 mm
Weight (without cables)	¡ approx. 1.1 kg
Operating temperature	-30^oC to + 50^oC

The laser used in this work was a Sick LMS111 (figure 1.5) positioned on the left side of the front bumper of the car (see figure 1.2). It is a scanning rangefinder with a single scan axis. The Sick LMS111 specifications are presented in table 1.1. In the context of this project, the most relevant properties are the LIDAR's maximum range (50 m) and the angular resolution (0.5^o).

Figure 1.5: Sick LMS111 [4]

1.5 Perception of the external environment - LIDAR vs. Vision systems

Since its emergence, autonomous vehicles make use of perception systems. The most common solutions are the LIDAR and the Vision systems. For tracking purposes, LIDAR has two main advantages over the vison systems:

- **Position** - Determining the position of the objects with LIDAR is a trivial process as the LIDAR outputs objects ranges. However in vision this problem can be overcome using stereo vision (*"knowing the position of two cameras and their optical properties, it is possible to estimate the location of objects in depth in a given scenario."* [5]) or assuming a particular object size which is an unreliable technique.

- **Segmentation** - Segmenting objects from the background is also a simpler task when using LIDAR as objects are clearly separated from the background (note: Objects that are close together may be hard to segment). On the other hand, in vision systems when two objects appear superimposed by our perspective it is hard to determine when an object ends and the next begins. [3]

The main disadvantage of the LIDAR over vision is that the laser data contains little information: range and angle. In computer vision, because we have an extended nature of the objects, it makes it more easy to classify objects of interest (pedestrians, vehicles, bushes, etc.)

1.6 Tracking problems using LIDAR

The use of a laser sensor can bring several issues; especially in more challenging situations in urban environment that occur frequently and should be overcome to perform a successful tracking which is the ultimate goal. In the next subsections, a number of problematic situations are enumerated and discussed.

1.6.1 Shape Change

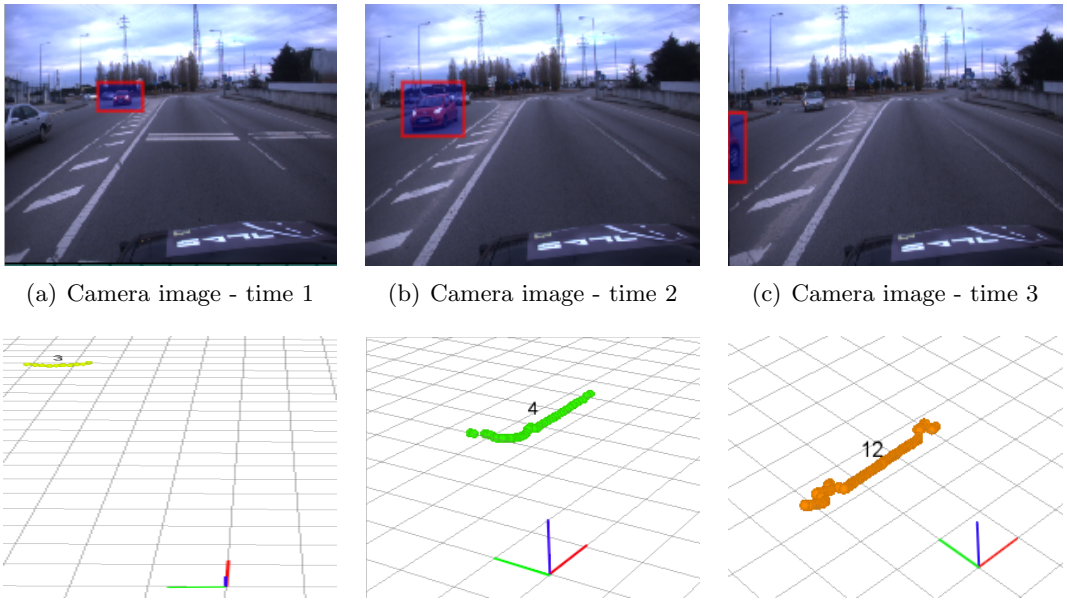
Object apparent shape change is a very common problem in tracking and is described as changes in the objects appearance due to the changing of the scanner perspective. The reason for this to happen is that the scanner only sees the part of the object surface currently facing the scanner; so, as the LIDAR moves around the object it perceives different contours of the object's surface. This problem is further emphasized for situations where the LIDAR itself is also in motion.

Although the shape change does not cause major difficulty in associating the scan data to the same object from one scan to the next because change is very small, the main difficulty is determining if the shape change is due to the change of perspective or the actual motion of the object.

In figure 1.6 is presented a situation during the car's ride where another vehicle (marked with the red rectangle) is coming in the opposite direction. At the time 1 it is only possible to see the front part of the car; at the time 2 we can see both front and side of the car; whereas at the time 3 when the detected car is going beside our car the LIDAR only scans the car's side.

1.6.2 Occlusion

Occlusion is one of the biggest problems when using a line of sight sensor, especially in a dynamic environment the urban scenarios when objects like trees, lampposts, columns and several others create occlusion areas.



(a) Camera image - time 1 (b) Camera image - time 2 (c) Camera image - time 3

(d) Marked car's correspondent scan representation - time 1 (e) Marked car's correspondent scan representation - time 2 (f) Marked car's correspondent scan representation - time 3

Figure 1.6: A car's shape change example.

An example of a situation that often happens when a moving object (object A) occludes a stationary object (object B) creating a false moving boundary (figure 1.7): Due to the fact that object B is being occluded by the object A the centroid of the object B seen in the LIDAR's perspective is moving in the direction of the remaining visible object.

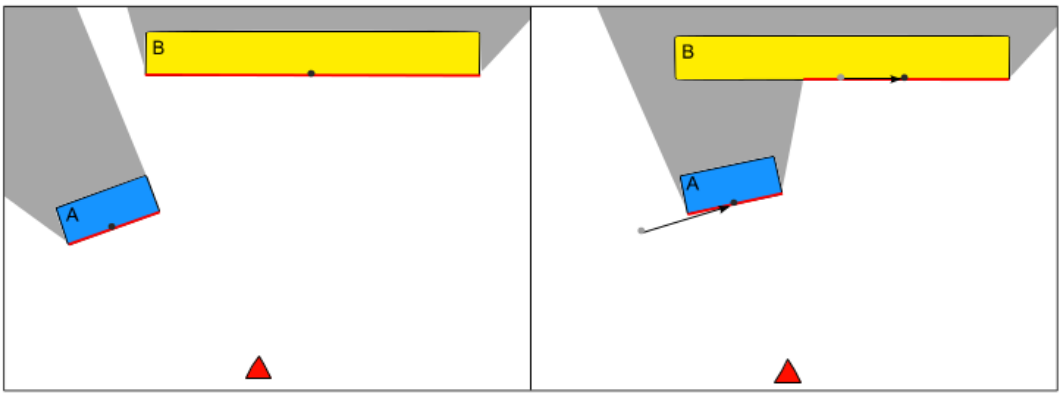


Figure 1.7: Artificial movement caused by partial occlusions. Object B seems to move as it is ocluded by the object A. The LIDAR sensor is represented as the red triangle, occlusion zones in gray, the visible part of the object is draw using a red line and the apparent perceived centre of each object is marked by a black dot. [6]

Occlusion particularly influences the segmentation process in cases where a small object is in front of a larger background object as shown in the figure 1.8 where partially occluded objects tend to create more segments than the number of objects: In the reported case, the object B was segmented as two objects and not just one, which was the real situation.

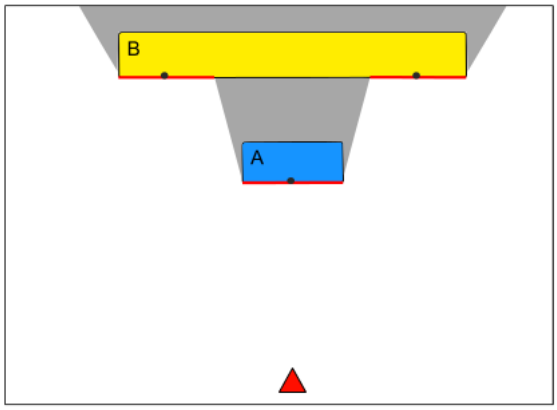


Figure 1.8: Poor segmentation of objects: object B is segmented as two distinct objects instead of just one. [6]

1.6.3 Cluttered situations

According to [13], cluttered situation are *"The drama of any segmentation method"*. A clutter situation is when objects are really close together, therefore is not clear if wee should segment the data as one or multiple objects, for example a group of people close to each other appears to be one single object or a person walking near a wall appears to merge with the wall.

1.6.4 Discontinuity in range

Discontinuity in range can be a real problem when working with high noise level laser Scanner; it is a measurement error on the transition from the nearer object to the farther object in which appear intermediate readings (figure 1.9). This causes error in the detection of the objects boundaries and can harm the segmentation results: Due to those *"drippy points"* [3], the two objects might be grouped as one single object. The noise level is due mainly to the LIDAR sensor's scanning patterns.

1.6.5 The horizontal laser scan data issue

When using a single axis scanner, only the 2D planar cross section of the targets is visible. Therefore when the scanner pitches (rotation around the side-to-side axis) or rolls (rotation around the front-to-back axis), we see a different contour of each object. Also when the ground is not flat, the scanner laser beam may hit the ground, resulting in seeing the ground itself as an obstacle.

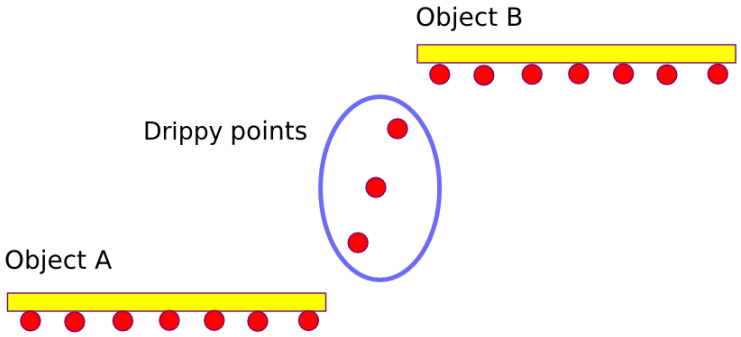


Figure 1.9: Example of a discontinuity in range where drippy points were created between two objects. The red circles represent the range points, the points surrounded by the blue ellipse are the ones "created" by the LIDAR.

1.6.6 Weak returns

The LIDAR is also sensible to weak reflectivity surfaces at the infrared frequency where the scanner operates like glass-built materials for example. This causes sparse or even incomplete measurements; The problem is even worse as the distance of the object form the LIDAR grows. Also LIDAR is sensitive to weather conditions as the laser beams can be absorved by rain or fog which results in more information loss. [3]

1.6.7 High angles of incidence

Another issue when using a LIDAR relates to high angles on incidence of the laser beam with surrounding environment objects as the return light is very small and a high angles on incidence the laser light can be completely reflected away from the sensor making the object invisible for the scanner (figure 1.10.)

1.6.8 Vegetation

Correctly segmenting vegetation like groups of trees and bushes can be a problem as shown in figure 1.11: *"The outline seen by the scanner appears to fluctuate in a random way as the scanner moves" and "clearly there is a great deal of noisy fluctuation of the range measurements"* [3]. Therefore, performing segmentation operations to vegetation is a hard process, it is also a very difficult task to define a simple geometric model for it.

In conclusion, these are problems that make laser detection a very difficult task, and a challenge for the segmentation algorithm. This is the main reason for the study and comparison of several segmentation algorithms capabilities made in the project.

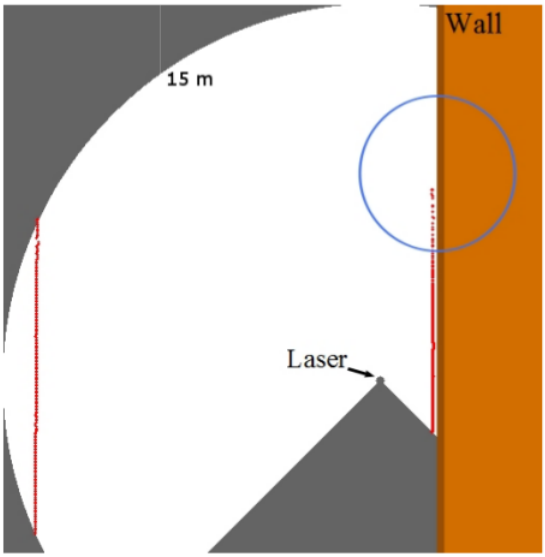


Figure 1.10: The Laser stops detecting the wall in the blue marked zone. [6]

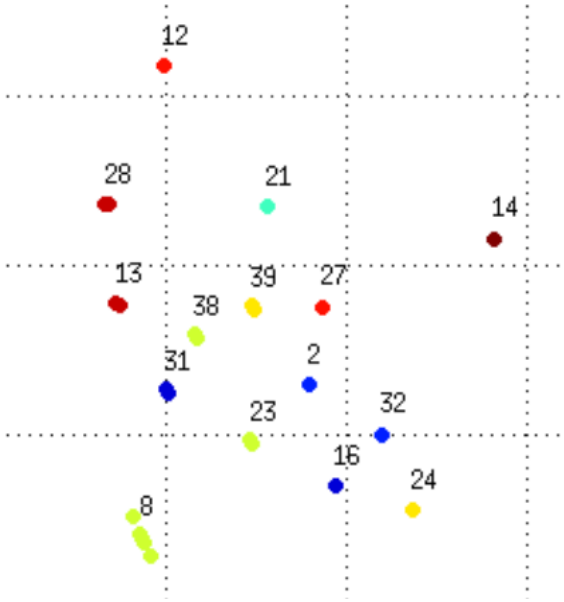


Figure 1.11: Scanned vegetation.

1.7 *Robot Operation System*

"“ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications.” [7].

ROS is a development environment specifically for applications in robotics. This environment was developed for large projects; due to its modular architecture it allows to reduce the projects complexity by spiting a large project onto smaller modules each with a specific application which eases its debugging and comprehension. Also those

modules can be used for other applications and not just in a specific project.

ROS was created in 2007 for the STAIR project (Stanford Artificial Intelligence Robot) ([8]). Since then, the development team has been ensuring the project's continuity. The Robot Operating System is intended to meet the following goals [9]:

1. Facilitate the development of modular software - Perform modular builds inside the source code tree;
2. Free and Open-Source platform;
3. Language Neutral - *ROS* currently supports four different languages: C++, Python, Octave and LISP;
4. Light and robust operating system;
5. Large number of small tools to build and run various *ROS* components - Message viewer, get and set configuration parameters, auto-generation of documentation etc.

The communication between processes (nodes) is performed trough *messages*; these *messages* are a simple data structure already predefined by *ROS*. A node sends a *message* through a publication of a topic which is a string used to identify the concept of the *message*; this is important for large projects because we can have several topics being published and subscribed for multiple nodes.

Most of the projects developed in the Laboratory of Automation use the *ROS* environment because, thanks to the mentioned modular architecture, a module can satisfy multiple tasks and can be easily incorporated in other researchers' projects.

Chapter 2

Segmentation of 2D laser data

In this chapter are summarised the main categories of 2D laser data segmentation methods, and a short survey on 2D laser data segmentation methods is given.

2.1 Segmentation categories

Segmentation constitutes a very important stage that can be performed by several methods. The main goal of this stage is to extract groups of points from the incoming $Scan_s$ that share similar spatial properties allowing to identity the limits of surrounding objects. Therefore, the "key step" is to identify the discontinuities in the laser data sequence which are called break-points and ideally will correspond to a real object boundary.

The several methods can be grouped in two main categories: 1) Distance-based method 2) Stochastic distance-based

2.1.1 Distance-based segmentation methods

Distance-based methods are the most common approach for break-point detection in 2D data mainly due to their simplicity and processing speed; however they also present considerable efficiency.

In order to find the break-points, the methods included in this category compute the Euclidean distance between two consecutive points: $D_{eu}(p_i$ and $p_{i-1})$ and if that distance is greater than a threshold value D_{thd}, a break-point is detected. The algorithm 1 summaries the general form of the distance-based segmentation methods [13].

Algorithm 1: Distance-based segmentation methods
Inputs: $Scan_s$: Sequence of n_s measurement points; D_{thd}: The specific threshold for the chosen method;
Outputs: S: Set of segments;

1: n_s: Number of points in the scan $Scan_s$;
2: $k \leftarrow 0$: Number of segments;

3: **for** i=1; $i < n_s$; i+1 **do**
4: Calculate the Euclidean distance D_{eu} between p_i and p_{i-1};
5: **if** $D_{eu} > D_{thd}$ **then**
6: A break-point is detected;
7: $k \leftarrow k + 1$ Increase the number of segments;
8: $S \leftarrow S_k$ Form the segment;
9: **else**
10: continue;
11: **end if**
12: **end for**

The only difference between these methods lie on the expression used to compute the threshold condition D_{thd}

2.1.2 Stochastic distance-based segmentation methods

Stochastic distance-based methods use a stochastic filter, namely Kalman filter approaches, to detect the break-points: A validation region (VR) is created around the estimated measurement and determines if a new laser-point should continue in the process or if it is discarded (break-point detected). The Algorithm 2 summaries the general form of the stochastic distance-based segmentation methods [13].

Algorithm 2: Stochastic distance-based segmentation methods
Inputs: $Scan_s$: Sequence of n_s measurement points; M: stochastic model of the filter
Outputs: S: Set of segments;

1: n_s: Number of points in the scan $Scan_s$;
2: $k \leftarrow 0$: Number of segments; Initialization of the filter model;
3: **for** i=1; $i < n_s$; i+1 **do**
4: State and covariance prediction;
5: Measurement prediction: $\hat{p_i}$
6: Generate the validation region (VR) conditioned on a threshold;
7: Compute the innovation $\nu = \hat{p_i} - p_n$;
8: **if** $\nu \in$ VR **then**
9: Filter gain calculation;
10: Update the state vector and the covariance matrix;
11: **else**
12: A break-point is detected;
13: $k \leftarrow k + 1$ Increase the number of segments;
14: $S \leftarrow S_k$ Form the segment;
15: Initialize model M;
16: **end if**
17: **end for**

2.2 Related work

In ([10]) it is described a method for basic object classification in traffic scenes using data from a single Laser-Range-Finder that is able to distinguish between cars, trucks/buses, motorbikes/bikes and small moving objects, for example, pedestrians. In addition to this classification, object tracking is also performed allowing a prediction of object trajectories and their dynamic behaviour. In the segmentation stage, it is used a distance-based segmentation method in which the criterion of unity is the distance between two consecutive measurements. If that distance is less than the threshold condition the points belong to the same segment. The threshold condition is given in function of the minimum range value of the two laser points in test. In addition, the threshold value strongly depends on a configurable constant that is used to adjust the algorithm to overcome noise and strong overlapping of pulses in close range.

According to the authors, in order to form a segment, it is required a minimum number of measurement points depending on the distance: for measurements near the laser's maximum range, it could make sense to track even a single measurement if it occurs in the region of interest. In close range, a segment must have at least ten/eleven points close to each other to establish a segment. Due to occlusions and disturbances, it may occur that several segments don't automatically represent an object of interest. However in traffic scenes, the number of objects of interest is limited, therefore the authors utilize a static apriori knowledge about typical road users dimensions (width and length) to perform the classification.

In [11] is described a tracking system constituted by three modules. Segmentation, static LIDAR and moving LIDAR. The system estimates the position and velocity of existing objects in the environment thanks to the scans provided by the LIDAR. In their segmentation process, the authors use a distance-based segmentation method based on the [10]: The threshold condition has the same parameters as in the previous method but with the addition of a new parameter β angle; this angle is another parameter used for segmentation sensibility refinement; *"β was introduced to reduce the dependence of the segmentation with respect to the distance between the LIDAR and the object."* [11].

In [12] is presented a geometrical feature detection framework with 2D LIDAR. This framework has three main procedures: Data pre-processing, break-point detection and line extraction. The breakpoint detection allows to determine sequences of measurements which are not interrupted by scanning surface changing; in this paper were investigated two breakpoint detectors, one based on adaptive thresholding (ABD) and the other on Kalman filtering (KFBD). The breakpoint detection stage has a major importance; as with the range discontinuities detected, the line extraction is performed within the region of points between two discontinuities. In this article the authors have been focused on the line kernels namely the Split-and-Merge Fuzzy (SMF) line extractor.

In the ABD method is also a distance-based method between two consecutive laser points. In this methodology it is defined a virtual line passing the previous laser point in the sequence of measurements given by the LIDAR (p_{i-1}); that line represents the extremum case where an environment line can be reliably detected. That virtual line makes an angle λ with respect to the scanning direction α_{i-1} of the range point p_{i-1} and aims to extrapolate the worst acceptable range point p_i. The treshold condition also

depends on the LIDAR's residual variance. The residual variance is used to encompass the stochastic behaviour of the sequence scanned points and the related noise associated to the range of the tested point.

Also in ([12]) is presented a method in which a Kalman filter is used in conjunction with a statistical test to verify if two adjacent range measurements belong to the same region. In this algorithm, a discrete system model is used to describe the dynamic behaviour of every range measurement; the statistical test is used to find the measurements which do not satisfy, in the stochastic sense, the predicted model. If the test fails, a breakpoint is detected and the filter is re-initialize.

A multivariable segmentation method is presented on ([13]). In this method is calculated a vector of attributes from two consecutive laser points that form a pair; this vector is compared with the next vector calculated from the pair ahead. A break-point is detected if the cosine similarity is lower than a defined cosine distance threshold.

A classical segmentation technique is the K-means originally present on ([14]). K-means is an algorithm used to group data based on attributes into a K number of groups, where K is a positive integer number. The first step is defining the number of segments of K and randomly give the initial coordinates to the K segments centroids' localization; For each data point, the distance is to every centroid is calculated in order to find out which centroid it's closed to; assign each data point based on the minimum distance; determine the new centroids position based on the points it owns and "jumps there", this process is repeated until a convergence situation is found: The K segments' centroids have not changed their position.

An overview of the K-means method is presented on figure 2.1; and on figure 2.2 shown the K-means assignment in which every data point represented by the colour circles are assigned to the nearest Mean represented by the "*"; then the new Mean position is calculated and the Mean moves to center of the formed segment.

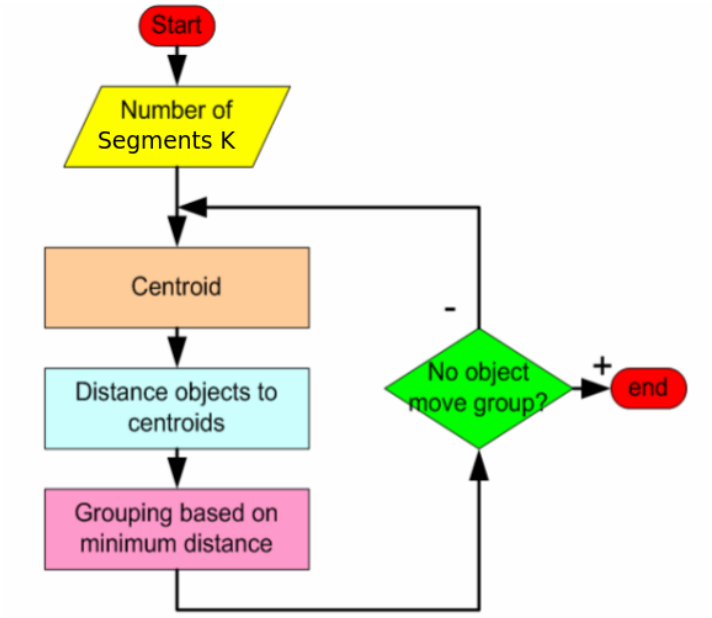


Figure 2.1: Overview of the K-means method. [15]

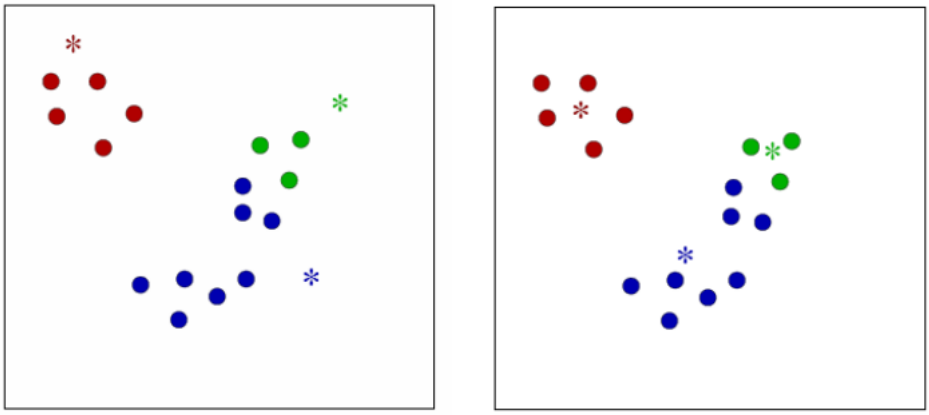


Figure 2.2: K-means assignment Step and Means update. a) Each point is assigned to the nearest mean. b) Mean is moved to the center of the segment. [16]

K-means algorithm can be used for multiple applications: Laser data, image processing, pattern recognitions, classification of plants or animals into distinct groups or species, clustering of landsat data, etc.

However, laser data segmentation with K-means present a significant disadvantage: The number of segments K must be determined before hand, that means which is not very convenient to use it in real time operations. $\underline{o}$

Chapter 3

Methods and Programming

3.1 Overview

This project is based on three main tasks: The implementation of the segmentation algorithms, the Ground-truth definition and the evaluation of the algorithms performance.

In the first task, several distance-based methods were implemented, namely: Simple Segmentation, Dietmayer Segmentation, Santos Approach, Adaptive Breakpoint Detector, Spatial Nearest Neighbour and the Multivariable Segmentation. The algorithms are applied to the raw laser scan data in order to extract groups of measurement points which share similar spatial properties and that probably will belong to one object. We will call these groups as segments; in the perfect case scenario each segment would correspond to one single object in the real world, however due to problems like the ones described on the section 1.6, this perfect scenario is hard to accomplish. Each algorithm has at least one threshold condition parameter that is configurable, and one of the goals is to ascertain the optimal value of that parameter for road environments. So, the next step is to compare the results of the segmentation given by the several algorithms with a defined Ground-truth and check the errors given by each method when the value of the threshold varies. Therefore a Ground-truth to make the comparison and test the validity of the segmentation methods is needed.

To obtain the Ground-truth, a Matlab program was used to hand-labelling the laser data. In short, the program reads the laser data from a text file and then the data is placed in a structure composed by the coordinates of the scanner points belonging to the labelled cluster, and the cluster's label. The next stage was to bring the Ground-truth results to the *ROS* platform and represent the results on *RVIZ* for a first visual comparison.

In order to evaluate the algorithms performance more effectively, and not just by visualizing, six performance measures were created. These performance measures penalize both under-segmentation and over-segmentation situations, they also have in consideration the segments sizes, the distance between centres or boundaries of the segments, and their correspondents in the Ground-truth. Both performance measures are described in the section 3.4.

The project's source code was developed in C++ programming language using Ubuntu 12.04 operating system, and as mentioned previously, this project is inserted in the advanced perception systems for the *AtlasCar* (section 1.1).

3.1.1 Integration with ROS and Matlab

In this subsection, a detailed overview of the project's stages and data flow is performed (figure 3.1). Initially the surrounding obstacles positions (raw data) are acquired by the LIDAR (LMS111) located on he left side of the front bumper of the *AtlasCar*. The raw data is written to a text file and them read by the software Matlab *R*2011*b* 7.13 in which the raw data will be manually labelled according to what appears to be reality the real world objects. The hand-labelling results are written to another text file which will be read in the ROS platform.

From the mentioned text file, the labelled points positions and labels are extracted to a created Point class containing laser point information: Cartesian and Polar coordinates and Ground-truth label. The next stage is to filter the invalid Points: Points with range values outside the LIDAR's range limits are deleted. An optional filter operation can be performed: Points that form small segments (e.g. segments with less than 3 Points) in the Ground-truth are also removed. Having the valid points laser Points, segmentation operations are performed using the different algorithms with several parameters variations. The segmentation algorithms will group the Points and form a set of segments; the segmentation results can be seen on the *Rviz* graphical user interface using a *visualization_msgs::MarkerArray*.

Lastly, the results of the segmentation stage (segment's centres and boundaries) are written to a new text file to be read in Matlab where the performance measures are applied.

3.2 Segmentation algorithms implementation

In tracking operations, the segmentation process is usually the first step after the acquisition stage. Nonetheless there can be a preprocessing stage which is intended to remove laser sensor noise using some techniques like median filters for example. Therefore, the segmentation stage is the primary stage to detect entities of interest.

The detected objects within the LIDAR's angle of view are described by a group of scan points which we call *segments* (also called *clusters*). To avoid misinterpretations, segmentation and *segments* will have the following definitions according to [17].

Definition 1. *Segmentation is the process of transforming the raw laser points into primal groups of segments (useful data).*

Definition 2. *Segment is a set of range measurements points in the plane close to each other and probably belonging to one single object. A given segment is defined by the set* $S_i : (i \leq n_{s)}$ *of laser points that respect a certain threshold condition where* n_s *is the total number of points in a scanner. The threshold condition depends on the chosen segmentation method.*

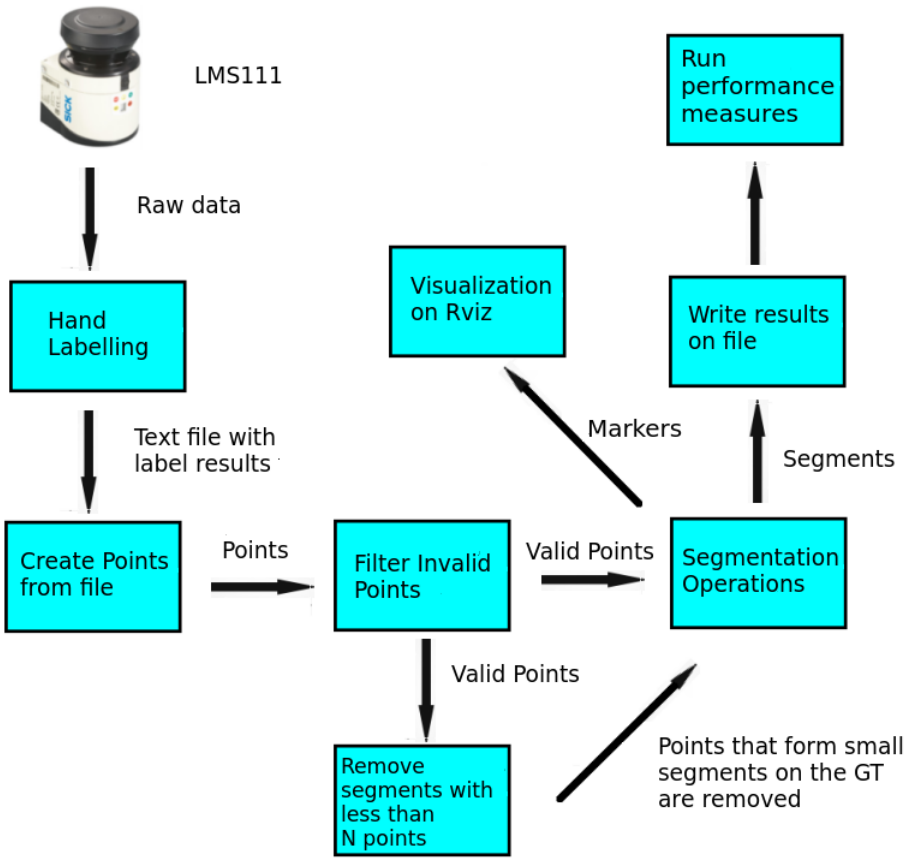


Figure 3.1: Overview of the project.

Consider a full scan $Scan_s$ as a ordered sequence of n_s measurement points ($\mathbf{p}$) in the form:

$$Scan_s = \{\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_{n_s}\}. \tag{3.1}$$

The raw data collected by the LIDAR is defined in a $\mathbb{R}^2$ space. These scanner points are commonly defined in polar coordinates (r_i, α_i) but also in Cartesian coordinates(x_i, y_i), meaning:

$$p_i = \left\{ \begin{pmatrix} r_i \\ \alpha_i \end{pmatrix}, \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right\}, \ i\epsilon[1, n_s] \tag{3.2}$$

Where:

$$\begin{cases} x_i = r_i cos(\alpha_i) \\ y_i = r_i sin(\alpha_i) \\ r_i = \sqrt{(x_i)^2 + (y_i)^2} \end{cases} \tag{3.3}$$

The two scan coordinates are illustrated more clearly in the Figure 3.2

During the acquisition stage, if the laser doesn't get a detectable return we have a case of a "missed scanned point". When this occurs, the LIDAR sets the value of point's

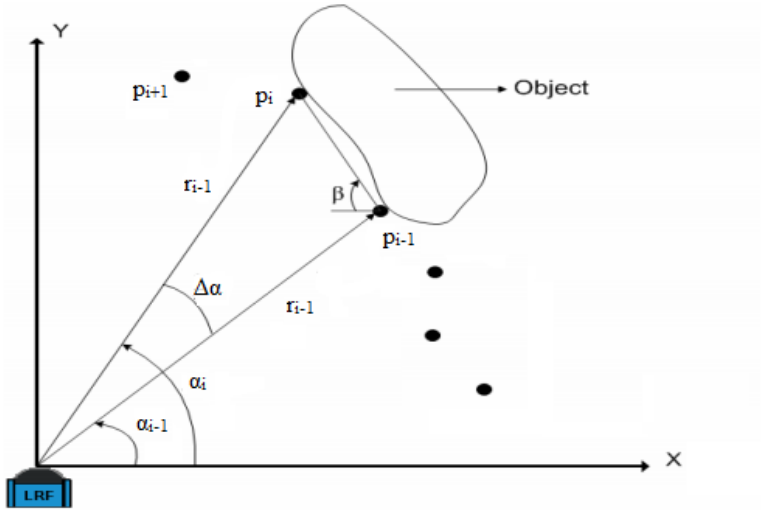


Figure 3.2: Geometrical representation of a hypothetical laser data.

range equals to 50.0 m; which corresponds to the Scanner's maximum operating range value, this misreading will be skipped during the Segmentation process.

During this preprocessing stage, it is a common practice to discard the isolated/spurious points called outliers; however, in this for project the outliers were kept because we believe that every single point is a piece of information: an outlier can represent an illumination post or a person, or even a car.

To conclude, the preprocessing module outputs the sets $Scan_s$ where the range-points (r_i, α_i) have a minimum range of r_{min} and a maximum range of r_{max} delimited by:

$$r_{min} < r_i < r_{max} \tag{3.4}$$

Thanks to 3.4, the number of points per scan will be less than the n_s range values output by the laser scanner.

In the following subsections, the theory behind the implemented algorithms will be detailed, and as it was mentioned before, the purpose of implementing the several Segmentation algorithms is to enquire which one resembles more with the Ground-truth (real objects). All these algorithms have a threshold parameter which for each algorithm depends on different causes. So the goal is to vary that parameter and find the optimal value which will correspond to the value where the segmentation results with that method will be closer to the Ground-truth.

Varying the threshold value can lead to two different situations: *under-segmentation* and *over-segmentation*.

Over-segmentation occurs when the objects being segmented are themselves segmented or fractured into subcomponents (smaller sets of scan points) which leads to an excessive number of segments. This situation increases the chances of some boundaries of importance like cars or persons being extracted and creating many insignificant boundaries. Over-segmentation is a common scenario when the threshold condition value is overly low.

Figure 3.3 shows a typical case of over-segmentation where the scan points relatively close to each other are separated into different objects, the segments are represented with different colors and marked with an id in the cluster's centroid position

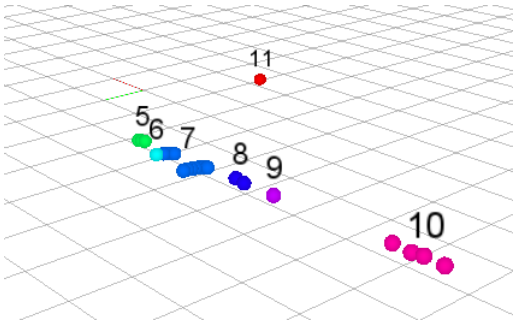


Figure 3.3: Over-segmentation of scan points situation. Too many segments.

Over-segmentation situations can be solved after the segmentation process by applying some kind of merging techniques which won't be discussed in this project.

On the other hand, when the threshold value for creating a new segment is excessively high, it can lead to an Under-segmentation situation. In this case, multiple objects can be grouped into one segment which in the end of the segmentation stage would result in a small number of segments. However, when using model fitting for object classification, this is not as critical as the over-segmentation as the detection and correction would be more difficult in that case: After the segmentation process there can be an object recognition stage which processes each segment individually and compare those segments with models of objects of interest (eg. by an overlapping process), so the stage can detect multiple objects in one segment. On the other hand if the object is split, the model fitting would be more difficult [18].

Figure 3.4 represents a case where multiple objects are grouped into a single segment. The two largest groups are two cars in the real road, but with an excessively high threshold value, they ended up being interpreted as one single object.

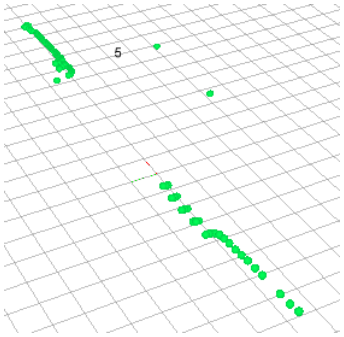


Figure 3.4: Under-segmentation of scan points situation. Too few segments.

3.2.1 Simple Segmentation (SS)

As the heading suggests, this is the simplest segmentation algorithm. In simple segmentation two consecutive points (neighbouring points) are considered part of the same *segment* if the Euclidean distance between them is lower than the threshold value:

if $D_{(r_i,r_{i-1})} > D_{thd}$ **then**
  Create a new segment
else
  Segments are not separated
end if

The Euclidean distance value between two consecutive scanner points can be calculated by:

$$D(r_i, r_{i-1}) = \sqrt{r_i^2 + r_{i-1}^2 - 2r_i r_{i-1} cos(\Delta\alpha)} \tag{3.5}$$

Where $\Delta\alpha$ is the laser's angular resolution.

As the laser's angular resolution $\Delta\alpha = 0.5^o$ is constant between two consecutive measurements, and has also a considerable small value, for the segmentation process it is common to use the simplified form:

$$D(r_i, r_{i-1}) \approx \mid r_i - r_{i-1} \mid \tag{3.6}$$

The Euclidean distance can as well be computed using the points' Cartesian coordinates:

$$D(r_i, r_{i-1}) = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{3.7}$$

In the Simple Segmentation algorithm the parameter that will vary is the D_{thd} itself, it is expected that when this values increases the number of segments will be lower and vice versa.

3.2.2 Dietmayer Segmentation (DS)

This method is a traditional break-point detector proposed on [10]. This is another example of a Distance-based method in which two consecutive points belong to the same object if the Euclidean distance between them (equation 3.5) is lower than a threshold value. For this method the threshold condition is given by:

$$D_{thd} = C_0 + C_1 min\{r_i, r_{i-1}\} \tag{3.8}$$

Where C_1 is defined by:

$$C_1 = \sqrt{2(1 - cos(\Delta\alpha))} \tag{3.9}$$

The C_0 is a constant that allows an adjustment of the algorithm to noise and strong overlapping of pulses in close range. Note that $\Delta\alpha$ is the angular distance between

the consecutive valid range points, it has the LIDAR's angular resolution value (0.5^o) if the points are consecutive in the raw data sequence. The following figure shows a geometrical representation of this method:

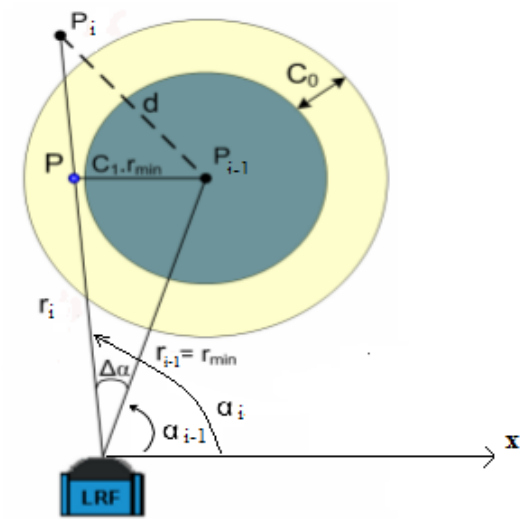


Figure 3.5: Geometrical representation of the variables in the Dietmayer Segmentation method. [17]

The $\min\{r_i, r_{-1}\}$ is the lowest ranging point value, meaning:
if $r_{i-1} < r_i$ **then**
$min\{r_i, r_{i-1}\} = r_{i-1}$
else
$min\{r_i, r_{i-1}\} = r_i$
end if

For this algorithm the variable parameter is the C_0 constant; increasing this constant will result in a final D_{thd} gain which will lead to a lower number of segments. On the other hand, a C_0 reduction will decrease the D_{thd} and result in a raise of the number of segments. It is also expected that the results for this algorithm would be similar to the Simple Segmentation because they are both based on the distance between two consecutive points.

3.2.3 Santos Approach (SA)

This segmentation criterion is based on the previous approach and it is described in [11]. Two consecutive valid scan points with ranges r_{i-1} and r_i, belong to the same segment as long as the distance between them fulfil the following expression:

$$D(r_i, r_{i-1}) \leqslant C_0 + \frac{C_1 min\{r_i, r_{i-1}\}}{cotg(\beta).[cos(\frac{\Delta\alpha}{2}) - sin(\frac{\Delta\alpha}{2})]} \tag{3.10}$$

The $\Delta\alpha$, C_0 and C_1 parameters are the same as in the previous method; this approach, however includes a new parameter β. This parameter is introduced to reduce

the dependency of the Segmentation regarding to the distance r_i between the LIDAR and the object.

The geometrical representation of this method can be found in the figure 3.6.

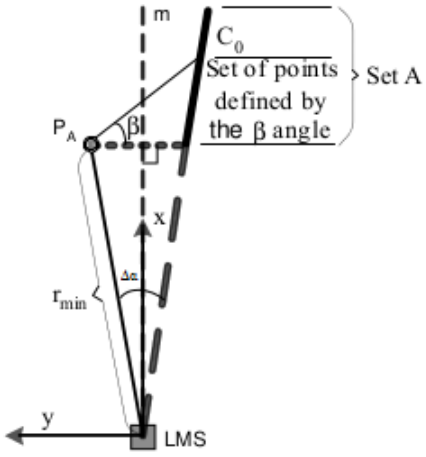


Figure 3.6: Geometrical representation of the variables onto the Santos Approach. [11]

Where $\Delta\alpha$ is the angular distance between two valid scan points and the set A represents the set of points defined by the β angle: The limit value for a point to belong to the same segment as the nearest point P_A.

As stated earlier, a dependency of the β angle is introduced. If the constant for noise reduction C_0 is removed, then β gives the maximum absolute inclination regarding to the perpendicular to the median line between two consecutive LIDAR beams (m in 3.6), that an object's surface can have to belong to a segment. A clear description of the β angle effect is shown by the 3.7.

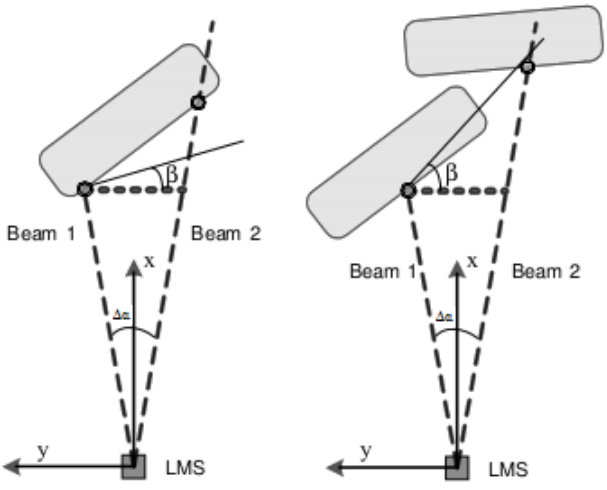


Figure 3.7: Different β angles effect. [11]

According to 3.7, if β is too small, the points of the object surface may not belong to the same segment; on the other hand, with an overly high β value, a segment can represent multiple objects.

This is the only algorithm in which two parameters will be tested (β and C_0 constants). In order to inquire their separate weight, when one parameter is tested the other one will have a fixed predefined value. As in the Dietmayer Segmentation, it is expected that the number of objects detected varies inversely with the C_0 value since its rise implies a decrease in the D_{thd}.

3.2.4 Adaptive Breakpoint Detector (ABD)

This is another distance-based on consecutive scanned points method and it is presented on [12]. In that article the authors have been more focused on their Split-and Merge Fuzzy (SMF) line extractor and other line kernels but before that they presented two break-point detectors: The Adaptive Break Point Detector and a Kalman filter-based break point detector (KFBD).

"The break-point detection allows to determine sequences of measurements which are not interrupted by scanning surface changing" [12]. With the intention of having a more effective line extraction, the break-point detection phase is of major importance. Without this task, the line extractor would tend to connect two neighbouring linear segments even if a large discontinuity exists between them.

As the name suggests, this method is based on adaptive thresholding. According to the authors, the threshold condition should be dependent of the scanned point range r_i since the LIDAR's angular increment is constant.

To determine the D_{thd} value, the following methodology is proposed according to the figure 3.8:

1. Define a virtual line passing on the scan point p_{i-1} that represents the extreme case where an "environment line can be reliably detected".
2. The virtual line should make an angle λ relatively to the scan direction α_{i-1} (line passes in p_{i-1}) which aims to extrapolate the worst acceptable range point p_i.
3. The angle λ is a free variable which the authors suggest to be defined about 10^o.
4. The detector generates a threshold circle centred at p_{i-1} with radius D_{thd} which adapts with the range r_i. If the next scan point p_i is outside that circle a new segment is created.

In conclusion, the ABD presents the following threshold condition:

$$D_{thd} = r_i \frac{sin(\Delta\alpha)}{sin(\lambda - \Delta\alpha)} + \sigma_r \tag{3.11}$$

The σ_r is the Laser's residual variance, although this value increases with the scanned distance r_i, $\sigma_r = 0.03$ m is a compatible value according to the laser LIDAR's datasheet

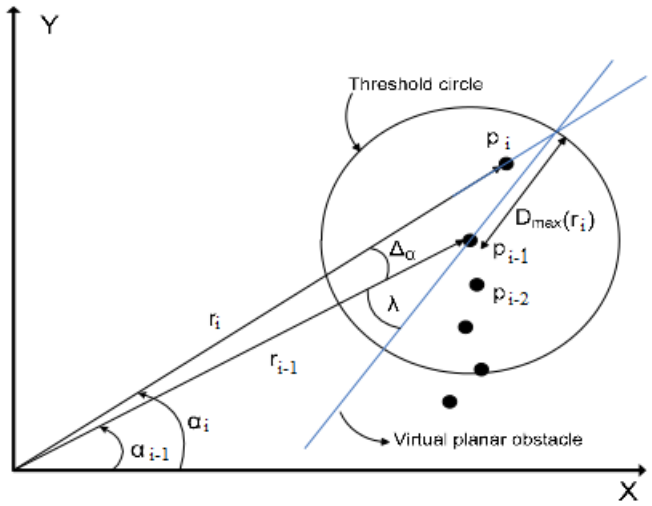


Figure 3.8: Geometrical representation of the variables onto ABD segmentation method. [17]

As mentioned in this subsection, the only free variable for this method is the angle λ, according to equation 3.11 it is expected that a higher λ value will result in the reduction of the D_{thd} and therefore in a higher number of segments.

3.2.5 Spatial Nearest Neighbour (SNN)

This method was used by Jorge Almeida in this implementation of the *Multiple Hypothesis Tracking algorithm* (check *http://lars.mec.ua.pt/lartk/doc/mtt/html/index.html* for more details). This is a recursive algorithm where the distance between a scanned point and all the other points that are not yet assigned to a segment is computed. If the distance is smaller than a certain clustering threshold, then the points are assigned to that cluster. We decided to call this method *Spatial Nearest Neighbour*.

The algorithm can be summarized as:

if p_i not yet assigned to a cluster **then**
  Calculate the Euclidean distance D_{eu} to all the other unsigned points
  if $D_{eu} < D_{thd}$ **then**
    Points will belong to that segment
  end if
  Go to the next unsigned point and repeat procedure until there are no more unsigned points.
end if

As in the Simple Segmentation algorithm, the tested free variable will be the threshold value itself. Therefore, it is expected that higher D_{thd} will result in bigger segments, and a smaller value for D_{thd} leads to a larger number of segments, but with a reduced number of measurements points per segment.

It is also expected that the result of segmentation process with this algorithm presents a lower number of segments when compared to the algorithms described previously.

While these other methods compute the euclidean distance between a point and its neighbouring, in this method that distance is calculated to all other scanned points (not yet assigned to a segment), so it is possible that the point p_i and the point p_{i+3} belong to the same segment while the points p_{i+1} and p_{i+2} belong to another. A clear example is represented in figure 3.9 where a large object can be segmented into smaller parts by a smaller object A that is occluding it.

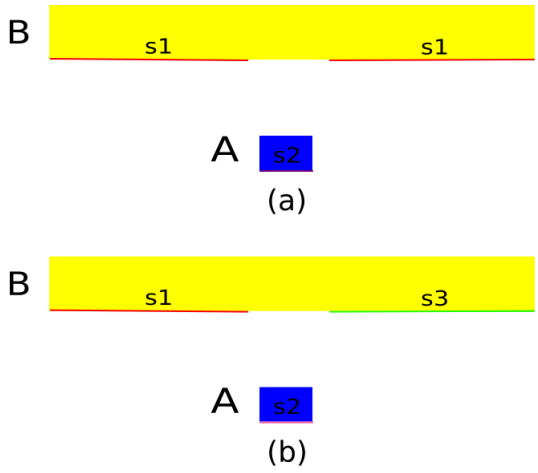


Figure 3.9: Hypothetical segmentations (a) Segmentation Using the SNN, non-consecutive background points are assigned to the same segment (s1) (b) With the consecutive distance-based methods, if a discontinuity in a object's surface is detected a new segment is created (s3). Inspired from [6]

In conclusion this method can partially solve some occlusion related problems and it is expected to be one of the algorithms to present better results in the algorithms evaluation performance.

3.2.6 Multivariable Segmentation (MS)

This method is presented in [13] and it is the most complex method implemented.

For this method, a set of attributes is calculated, a multivarible array of features that are extracted from pairs of consecutive laser points, for instance $P_i = [p_i, p_{i+1}]$ which is compared to the next pair $P_{i+1} = [p_{i+1}, p_{i+2}]$ vector of attributes. P_{i+2} is considered a breakpoint if the *cosine distance* between the pairs P_i and P_{i+1} is lower than the specified threshold value.

So, for each pair of laser points P_i and P_{i+1}, is calculated the following set of features:

- $f_1 = \sqrt{\Delta P_x^2 + \Delta P_y^2}$: where $\Delta P_X = x_i - x_{i+1}$ and $\Delta P_Y = y_i - y_{i+1}$ are the difference in the Cartesian directions between between p_i and p_{i-1};
- $f_2 = \bar{P}_i$: Is the average value of p_i and p_{i-1}, that is $f_2 = (r_i + r_{i+1})/2$;
- $f_3 = \bar{P}_i.\Delta P_X$: scalar multiplication between $\bar{P}_i$ and ΔP_X;

- $f_4 = \bar{P}_i.\Delta P_Y$: scalar multiplication between $\bar{P}_i$ and ΔP_Y;
- $f_5 = \sqrt{(r_i^2 - \bar{P}_i) + (r_{i-1}^2 - \bar{P}_i)}$: standard deviation of $\bar{P}_i$;
- $f_6 = f_5^2$: second order moment of $\bar{P}_i$;

This segmentation method is a sequential process in terms of spatial ordering performed for two pairs of points for instance P_i and P_{i+1}. In the author's project spurious/isolated points are discarded in the preprocessing stage unlike this work. So it was needed to slightly modify the original algorithm by introducing an additional Euclidean distance threshold just to identify those spurious points because it does not make much sense to create a pair with a spurious point in it. Also if the next point is too far, it is also created a new segment. In the experiments it was considered $D_{thd} = 3.0\ m$ which is a high value for segmentation purposes, the goal is just to break-point clear boundaries and let the other be detected by the cosine distance condition.

In short, the following methodology was implemented:

Multivariable Segmentation

Inputs: $Scan_s$: Sequence of n_s measurement points; $CosD_{thd}$: Cosine distance threshold value; D_{thd}: Euclidean distance threshold value to consider a point as an isolated point;
Outputs: S: Set of segments;

n_s: Number of points in the Scan $Scan_s$;
$k \leftarrow 0$: Number of segments;
for i=1; $i < n_s$; i+1 **do**
    Calculate the Euclidean distance D_{eu} between p_i and p_{i-1};
    if $D_{eu} > D_{thd}$ **then**
        A break-point is detected;
        $k \leftarrow k + 1$ Increase the number of segments;
        $S \leftarrow S_k$ Form the segment;
        Calculate the Euclidean distance D_{euc} between p_i and p_{i+1}
        if $D_{euc} < D_{thd}$ **then**
            The pair is formed;
        end if
        continue;
    else
        Calculate the vector of attributes $s_i = (f_1, ..., f_6)$ for the pair (p_{i-1},p_i) and $s_{i+1} = (f_1, ..., f_6)$ for the pair (p_i,p_{i+1})
        Calculate the cosine distance $CosD_i$ between s_i and s_{i+1}
        if $CosD_i < CosD_{thd}$ **then**
            A break-point is detected;
            $k \leftarrow k + 1$ Increase the number of segments;
            $S \leftarrow S_k$ Form the segment;
            Calculate the Euclidean distance D_{euc} between p_i and p_{i+1}

```
22:             if D_euc < D_thd then
23:                 The pair is formed;
24:             end if
25:             continue;
26:         else
27:             continue;
28:         end if
29:     end if
30: end for
```

The mentioned cosine distance is a measure of similarity between two vectors (in our case the vectors are the pairs of laser points) of an inner product space that measures the cosine ($\cos(\theta)$) of the angle θ between them. It is important to understand that the cosine distance is a judgement of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90^o have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

Considering the vectors of attributes s_i and s_{i+1}, the cosine distance between them is given by:

$$CosD_i = \frac{s_i.s_{i+1}}{\|s_i\|\|s_{i+1}\|} = \frac{\sum_{j=1}^{6} s_i(j) \times s_{i+1}(j)}{\sqrt{\sum_{j=1}^{6}(s_i(j))^2} \times \sqrt{\sum_{j=1}^{6}(s_{i+1}(j))^2}} \tag{3.12}$$

For this method, the tested variable in the evaluation stage will be the cosine distance threshold $CosD_{thd}$, it is expected to have a higher number of segments when the $CosD_{thd}$ has a value near 1 which means that the points need to be very similar in order to belong to the same segment.

3.3 Ground Truth definition

After successfully implementing the segmentation algorithms, the next step is to compare the results of each algorithm with the Ground-truth and check the errors given by each method when the correspondent threshold varies.

So, a Ground-truth to make the comparison between the methods is needed. To accomplish that, a Matlab program made by Jorge Almeida, the author of [6], was used. In brief, the program follows the directives:

1. Read the raw laser data from a text file;
2. Converted data to Cartesian coordinates;
3. Create a structure composed by the points coordinates and their labels;
4. Manual labelling of the scan data: perform a segmentation process from a human perspective; in this stage segments should correspond exactly to one real world object.

The data set was collected during a ride to a big roundabout nearby Aveiro. This roundabout is shown in figure 3.10.

Figure 3.10: Roundabout where the data set was created. [19]

In order to obtain a more correct hand labelling, pictures took by the car's cameras were used. In the next figures are represented examples of the manual labelling operation. Figure 3.11 shows the raw data not yet labelled and figure 3.12 shows the labelled data in which segments are represented by a different color and have different ids. On the right side of the images are pictures taken by the three *AtlasCar's* cameras which give an idea how is the surrounding environment during the time of the acquisition experiment.

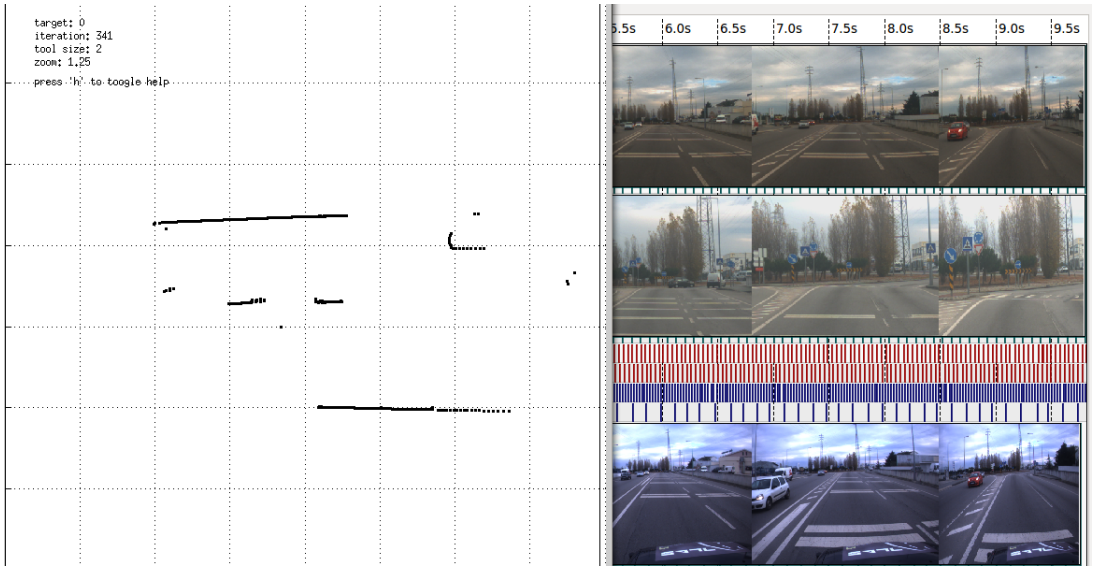


Figure 3.11: Unlabelled scan data.

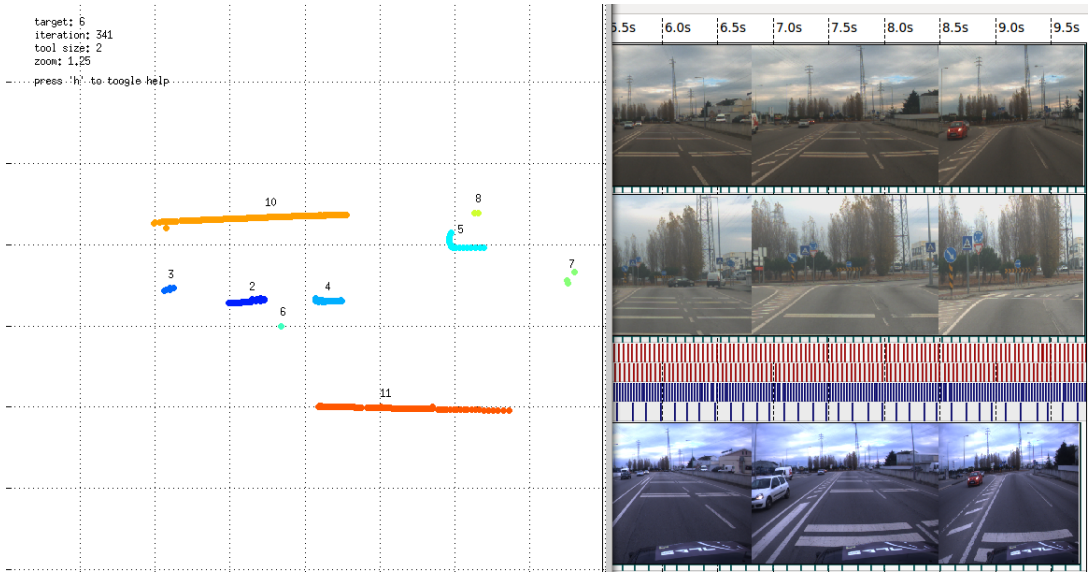


Figure 3.12: Hand labelled scan data. Each scan is marked with a different color and has a unique id.

At first, only the moving objects (objects of interest for tracking) were labelled. However, it was decided that the segmentation of only those objects wasn't enough to assess the algorithms performance because a segmentation algorithm needs to process all the laser data and not just the the data that corresponds to moving objects, which may not always be known.

In the end a total of 1220 scans were manually segmented the best way a human could. However not all the scans were as elegant and clean and relatively easy to label as the one presented in the last figure.

Examples of problematic scans:

- The roundabout where the car's journey was made (figure 3.10) has some trees in it which was hard to label (group of the single points on the top left of the image 3.13). "With some objects, the outline seen by the scanner appears to fluctuate in a random way as the scanner moves and vegetation has this problem" [3].

- The next example presents a zone of extreme subjectivity to segment: that zone is marked with a red ellipse in figure 3.14. The problem with that marked zone is that it isn't clear what those range points may represent in "real world" so the main point here is that, for Ground-truth purposes, scans like this one will not be used for comparison so they don't compromise the truth of the results when the comparison between the several algorithms is made.

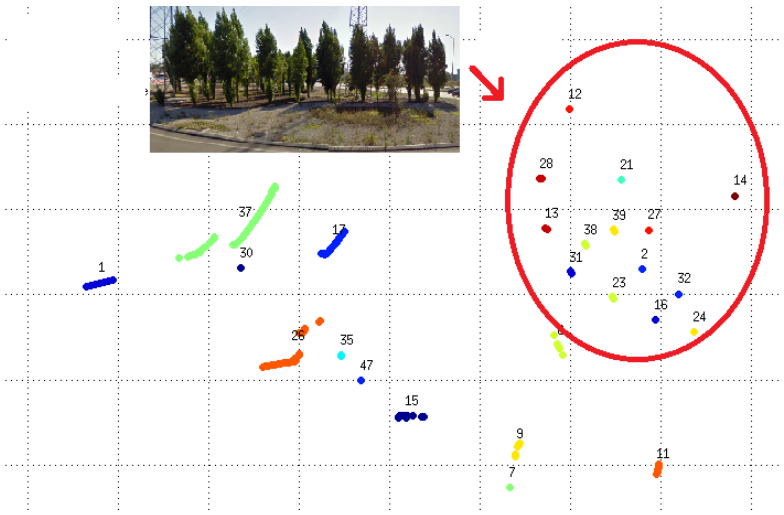


Figure 3.13: Example of a subjective segmentation: scan containing the roundabout with vegetation (zone marked by the red ring).

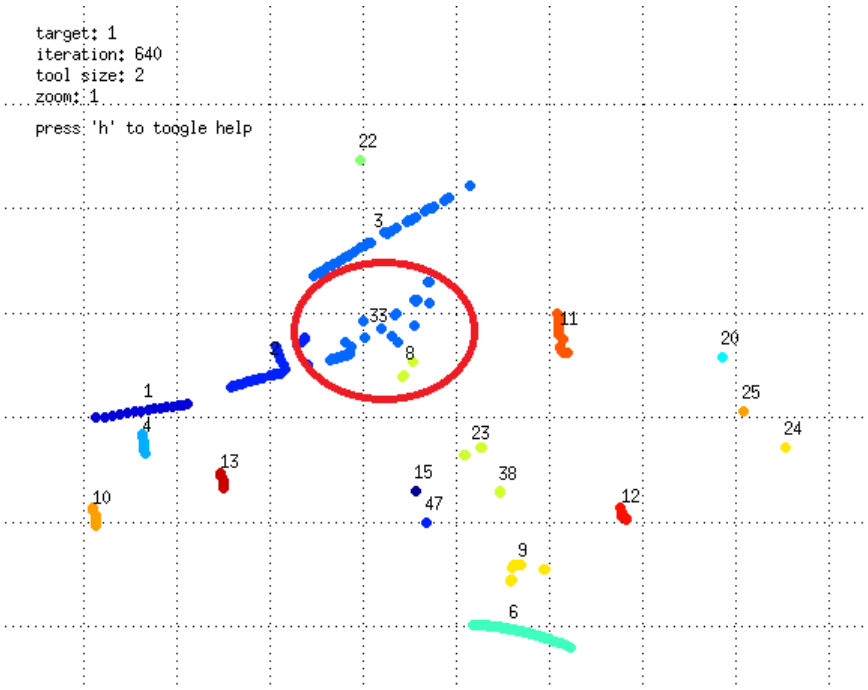


Figure 3.14: Example of a zone very subjective to segment.

3.3.1 Data Visualization

After manually segmenting the scans, the results of the manual segmentation were written into a text file with the following information:

- Number of the scan
- Cartesian coordinates of the points ranges
- Labels of the points

That text file is later read on the *ROS* platform. In order to guarantee the synchronization between the Matlab imported data and the data from the *rosbag* from which he laser scan data was extracted, it was decided to use the text file containing the Ground-truth data to create a Point class containing laser point information to be segmented with the segmentation algorithms.

One very important topic in LIDAR related work is the visualization of the scanned data and treatment effects applied to them. So it was relevant to display those results, preferably in a platform that allows having all the resulted segments at once in order to enable a direct comparison between each algorithm's results.

Resorting to the the *Rviz* graphical user interface, it was possible to represent the segments obtained for the application of each segmentation method. In figure 3.15 each layer represents the segments for each implemented segmentation method, counting upwards:

1. Simple Segmentation (SS)
2. Multivariable Segmentation (MS)
3. Dietmayer Segmentation (DS)
4. Santos approach (SA)
5. Adaptive Break-point Detector (ABD)
6. Spatial Nearest Neighbour (SNN)
7. Ground-truth (GT)

3.4 Performance Evaluation

With all algorithms and the Ground-truth synchronized, it's time to perform a quantitative evaluation of the algorithms performance. Therefore, at first, four comparison performance measures (PM) were developed; these performance measures return an energy value E. The energy is a cost function were the lower the energy, the better the performance of the algorithm, which means that the result of segmentation is close to

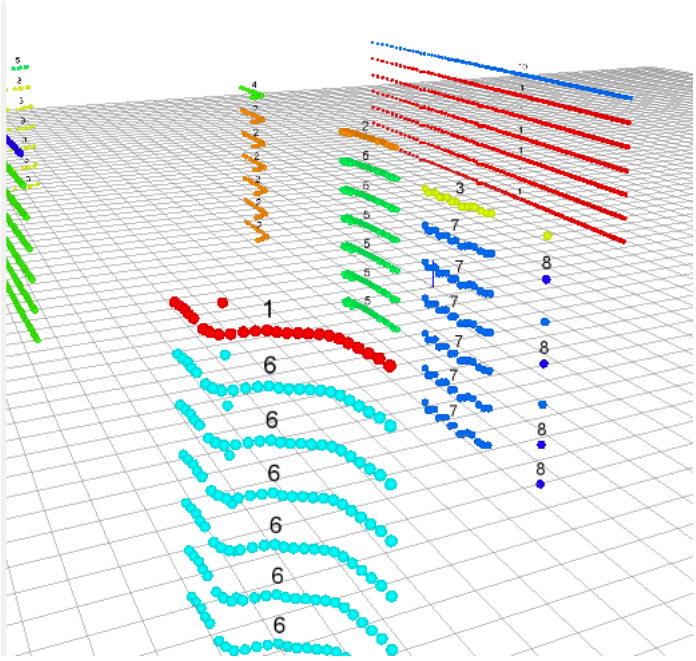


Figure 3.15: Representation of the segmentation results in the *Rviz* platform. Each layer represents the segments obtained for each segmentation method.

the Ground-truth. A Matlab program was developed to make the comparison between the algorithms and the Ground-truth by applying the performance measures.

The energy is calculated for every implemented algorithm for each tested threshold value segmentation results. An energy is calculated for each scan (iteration it) using all the four performance measures ($E1_{it}$, $E1A_{it}$, $E2_{it}$ and $E2A_{it}$) and an average energy for all scans. The performance measures can be described by:

Performance measure 1:

$$E1_{it} = \frac{\sum_{j=1}^{n\_Segments_{Alg}} [min\|CAlg_j - CGt_w\|.max(\frac{n\_pointsGt_w}{n\_pointsAlg_j}, \frac{n\_pointsAlg_j}{n\_pointsGt_w})]}{n\_Segments_{Alg}} \tag{3.13}$$

$$Average\ Energy_1 = \frac{\sum_{it=1}^{it=n_{it}} E1_{it}}{n_{it}} \tag{3.14}$$

Performance measure 1A:

$$E1A_{it} = \frac{\sum_{j=1}^{n\_Segments_{Alg}} [\|ib\_Alg_j - ib\_Gt_w\| + \|fb\_Alg_j - fb\_Gt_w\|]}{n\_Segments_{Alg}} \tag{3.15}$$

$$Average\ Energy_{1A} = \frac{\sum_{it=1}^{it=n_{it}} E1A_{it}}{n_{it}} \tag{3.16}$$

Performance measure 2:

$$E2_{it} = \frac{\sum_{w=1}^{n\_Segments_{Gt}} [min\|CGt_w - CAlg_j\|.max(\frac{n\_pointsGt_w}{n\_pointsAlg_j}, \frac{n\_pointsAlg_j}{n\_pointsGt_w})]}{n\_Segments_{Gt}} \tag{3.17}$$

$$Average\ Energy_2 = \frac{\sum_{it=1}^{it=n_{it}} E2_{it}}{n_{it}} \tag{3.18}$$

Performance measure 2A:

$$E2A_{it} = \frac{\sum_{j=1}^{n\_Segments_{Alg}} [\|ib\_Gt_w - ib\_Alg_j\| + \|fb\_Gt_w - fb\_Alg_j\|]}{n\_Segments_{Gt}} \tag{3.19}$$

$$Average\ Energy_{2A} = \frac{\sum_{it=1}^{it=n_{it}} E2A_{it}}{n_{it}} \tag{3.20}$$

Where:

- $n\_Segments_{Alg}$: Number of segments resulted from the algorithm's Segmentation process (for a given threshold);
- $n\_Segments_{Gt}$: Number of segments presented in the Ground-truth;
- $n\_pointsAlg_j$: Number of points of the algorithm's segment j;
- $n\_pointsGt_w$: Number of points of the Ground-truth's segment w;
- $CAlg_j$: Central point of the algorithm's segment j;
- CGt_w: Central point of the the Ground-truth's segment w;
- ib_Alg_j: Initial point of the algorithm's segment j;
- ib_Gt_w: Initial point of the Ground-truth's segment w;
- fb_Alg_j: Final point of the algorithm's segment j;
- fb_Gt_w: Final point of the Ground-truth's segment w;
- n_{it}: Number of scans (iterations);

Before explaining the meaning of the performance measures terms, it is important to understand the difference between the "segment's centroid" and "segment's central point" (figure 3.16). The centroid is the medium value in both Cartesian directions, while the central point has the coordinates of the point in central index.

With the purpose of comparing the similarity between the results of the algorithm and the Ground-truth, it was decided that the central points of the segments would be used instead of the centroids.

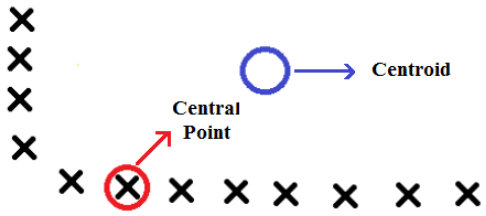


Figure 3.16: Blue circle - segment's centroid; Red circle - segment's central point.

In the performance measure 1 (equation 3.13), and for every iteration *it*:

For each segment given by the method in test: calculate the Euclidean distance to every Ground-truth segment (central point distance). It is considered that a segment j is associated with a Ground-truth's segment w when the Euclidean distance between their central points has the minimum value (distance to the closest segment in GT, see figure 3.17). So, the higher this distance the higher the Energy. In order to penalize both under-segmentation and over-segmentation scenarios; the distance between those associated segments is multiplied by $n\_pointsAlg_j/n\_pointsGt_w$ when in a under-segmentation situation or multiplied by $n\_pointsGt_w/n\_pointsAlg_j$ for penalizing over-segmentation situation. In order to normalize the process, the result is divided by the number of segments resulted from the algorithm's segmentation process $n\_Segments_{Alg}$.

The performance measure 1A (equation 3.15) is based on the segment association (algorithm segment j to Ground-truth segment w) made in the performance measure 1. The main difference is that instead of calculating the Euclidean distance between associated segments' central points, it is calculated the Euclidean distance between the segments' initial points and final points, in other words, the Euclidean distance between the segments' extremes (figure 3.18) and then they are summed. This way cases of under-segmentation and over-segmentation are already penalized with no need to compute their sizes. As in performance measure 1, the process is normalized by dividing the result by the number of segments resulted from the algorithm's segmentation process $n\_Segments_{Alg}$.

In the performance measure 2 (equation 3.17), for every iteration *it*:

For each Ground-truth's segment: calculate the Euclidean distance to every segment given by the algorithm in test. As in the performance measure 1, a segment j is associated with a Ground-truth's segment w when the distance between their central points has the minimum value (distance of the closest segment given by the tested algorithm, see figure 3.17) and penalizes the under-segmentation and over-segmentation scenarios in the same way. The result is normalized by diving it by the number of segments given by the Ground-truth $n\_Segments_{Gt}$.

The performance measure 2A (equation 3.19) is based on the segment association (Ground-truth segment w to algorithm segment j) made in the performance measure 2. Then, the Euclidean distances between associated segments' initial and final boundaries (figure 3.18) are summed. As in performance measure 2, the result is normalized by dividing it by the number of segments given by the Ground-truth $n\_Segments_{Gt}$.

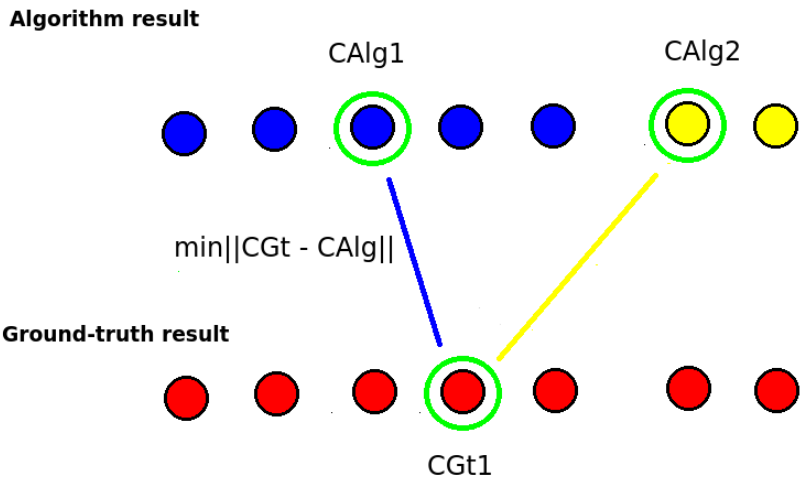


Figure 3.17: Example of an algorithm's segment and Ground-truth's segment association; the association is made by the proximity between the centres of the segments, in this case, the blue segment's center is closer to the Ground-truth's segment center than the yellow segment's center - Red dots represent the GT range points, the blue and yellow dots are the range points belonging to two algorithm's segments; the segments' centres are marked with the green ring.

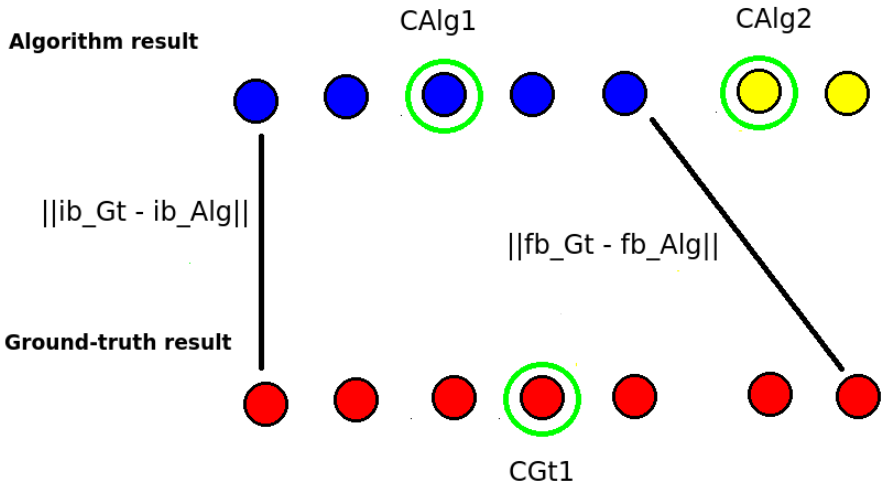


Figure 3.18: Geometrical representation of the Euclidean distance between extreme points of the closest pair of segments - Red dots represent the GT range points, the blue and yellow dots are the range points belonging to two algorithm's segments; the segments' centres are marked with the green ring.

In general terms, performance measures 1 and 1A focus on the non-correspondence of segments, whereas the performance measures 2 and 2A checks the direct correspondence between the Ground-truth and the algorithm.

However, the results obtained with these performance measures didn't make very clear the optimal threshold values for each algorithm for the different road scenarios, so the results of these performance measures will not be presented. It was decided that it was more important to favour the segments' correspondence than to penalize the non-correspondence, therefore the performance measures 1 and 1A were discarded. Furthermore, the performance measures 2 and 2A are normalized with the number of segments presented in the Ground-truth which is always constant; therefore, in order to penalize even more over-segmentation and under-segmentation situations, the result is normalized by dividing it by:
$min(n\_Segments_{Alg}/n\_Segments_{Gt}, n\_Segments_{Gt}/n\_Segments_{Alg})$, resulting in:

Performance measure A:

$$EA_{it} = \frac{\sum_{w=1}^{n\_Segments_{Gt}} [min\|CGt_w - CAlg_j\|.max(\frac{n\_pointsGt_w}{n\_pointsAlg_j}, \frac{n\_pointsAlg_j}{n\_pointsGt_w})]}{min(\frac{n\_Segments_{Alg}}{n\_Segments_{Gt}}, \frac{n\_Segments_{Gt}}{n\_Segments_Alg})} \tag{3.21}$$

$$Average\ Energy_A = \frac{\sum_{it=1}^{it=n_{it}} EA_{it}}{n_{it}} \tag{3.22}$$

Performance measure B:

$$EB_{it} = \frac{\sum_{j=1}^{n\_Segments_{Alg}} [\|ib\_Gt_w - ib\_Alg_j\| + \|fb\_Gt_w - fb\_Alg_j\|]}{min(\frac{n\_Segments_{Alg}}{n\_Segments_{Gt}}, \frac{n\_Segments_{Gt}}{n\_Segments_Alg})} \tag{3.23}$$

$$Average\ Energy_B = \frac{\sum_{it=1}^{it=n_{it}} EB_{it}}{n_{it}} \tag{3.24}$$

Even though these performance measures may not be the only ones to evaluate the algorithms performances, they gather some advantages:

1. Have the same treatment for all the tested algorithms;
2. Penalizes the distance between the segments;
3. Penalizes differences between the segments sizes;
4. Take into account the number of segments;

The results from the application of the performance measures are presented in the next chapter.

Chapter 4

Results and Discussion

This chapter presents the results obtained by applying the performance measures described in section 3.4 in order to evaluate the performance of various algorithms as well as making some comparisons between them.

4.1 Results obtained

As it was mentioned in previous sections, it is intended to establish the reliability of the algorithms and their limitations. Therefore, to perform a quantitative analysis to all the implemented algorithms, they were compared to the Ground-truth through the performance measures developed. The performance measures return an "Energy" value which depends on the distance between associated segment centres, the number of points of the segments and the number of segments. This energy amount takes lower values when the results presented by the algorithm are similar to the ones present in the Ground-truth.

As mentioned in section 3.3, the data set was collected during a ride to a roundabout near Aveiro; the complete route is presented in figure 4.1 resorted to the *Google Maps* application.

With the purpose of inquiring the behaviour of the algorithms in different road situations, it was decided to divide the course into three parts:

1. Straight line course
2. Roundabout approach (time of giving away included)
3. Roundabout

In order to give an idea how the referred pathways look like, figure 4.2 shows a image taken from the application *Google Maps Street View* for each pathway.

As mentioned in section 3.3, 1220 scans were labelled and some of them were discarded due to segmentation ambiguities that could compromise the quality of results. Therefore, after subtracting the discarded scans, we ended up with a total of 867 scans which is still a representative amount distributed as follows:

Figure 4.1: Route where the data set was collect, the color marks represent the different part of the course: Blue - straight line; Green - roundabout approach; Red - roundabout. [19]

- Straight line course (Scene 1) - 413 scans
- Roundabout approach (Scene 2) - 108 scans
- Roundabout (Scene 3) - 346 scans

Thus the most influential routes will be the straight line route and the roundabout route since they are the ones that held most iterations.

4.2 Expected results and precautions

Having a priori knowledge of the course, some occurrences are expected:

The scene 1 is characterized by having segments with a high number of points and for having the segments too distanced from each other. These big segments generally correspond to the roadsides. A typical scene 1 scan is shown in the figure 4.3.

Therefore, for this scene, it is expected that the energy value stabilize starting from a given threshold value, as it is needed a overly high threshold value to over-segment scans as the one presented.

Another situation that has already been mentioned in the section 3.3, is the fact that the vegetation in the roundabout is very subjective to segment. Thus, new scans

(a) Straight line course

(b) Roundabout approach

(c) Roundabout

Figure 4.2: Google Maps Street View images of the three pathways. [19]

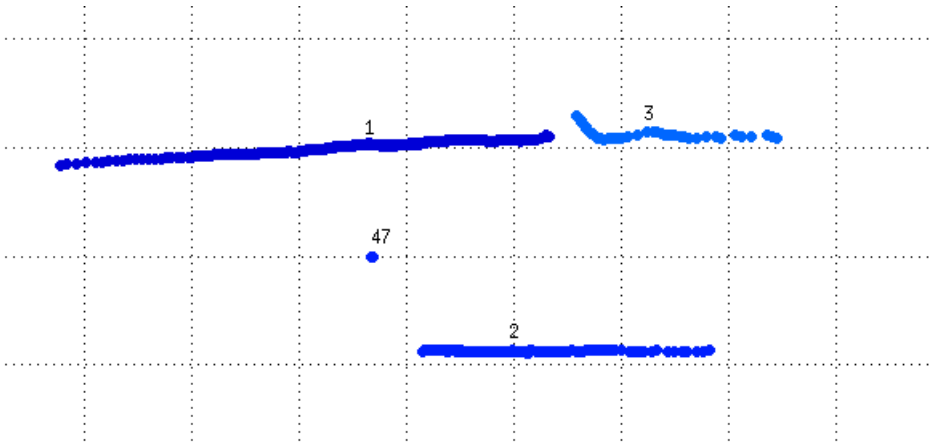


Figure 4.3: Typical scene 1 scan. Large segments and very distanced from each other.

were created: the Points that form Ground-truth segments with less than tree points are removed. Consequently, these new scans only contain valid points and non-spurious points, so it is expected that segments that represent illumination poles and trees or bushes are removed.

4.2.1 Threshold variations

For every implemented algorithm, the respective threshold value was varied twenty one times between values which hold physical meaning and that are considered acceptable (eg. it makes no sense to consider two consecutive points distanced by more than 5 meters to belong to the same segment.)

The next set of graphics show the energy values resulted from the performance measures evaluation for each implemented algorithm. Those graphics also represent the variation of the [*number of segments Ground-truth/number of segments algorithm*] ratio; this allows a better detection of the thresholds values for which there are scenarios of under-segmentation or over-segmentation.

The perfect threshold value for an algorithm would be the one that has a minimum energy value for both performance measures and has the same number of segments than the ones on the Ground-truth, meaning: ratio [*number of segments Ground-truth/number of segments algorithm*] = 1

As mentioned in the section 3.2.3; the Santos Approach has two configurable parameters (β and C_0 constants). Therefore, when the influence of C_0 was tested, β had a fixed value of 15^0. When the β variable was made varying, the C_0 had the fixed value of 1.00 meter. Those fixed values were assigned for being theoretically acceptable.

On the tables 4.1 and 4.2 are presented the evaluation results according to each performance measure for each pathway: scene 1 - straight line course, scene 2 - roundabout approach; scene 3 - roundabout. The tables show the values of the tested parameters of each algorithm in which resulted in the minimum average energy value which is marked in the graphs by a red asterisk.

Note: Despite having made twenty two tests for each algorithm, some of them are not represented in some graphics. In order to highlight the minimum Energy value, threshold values with a energy value too far from the minimum value are not included as it they disrupt the viewing.

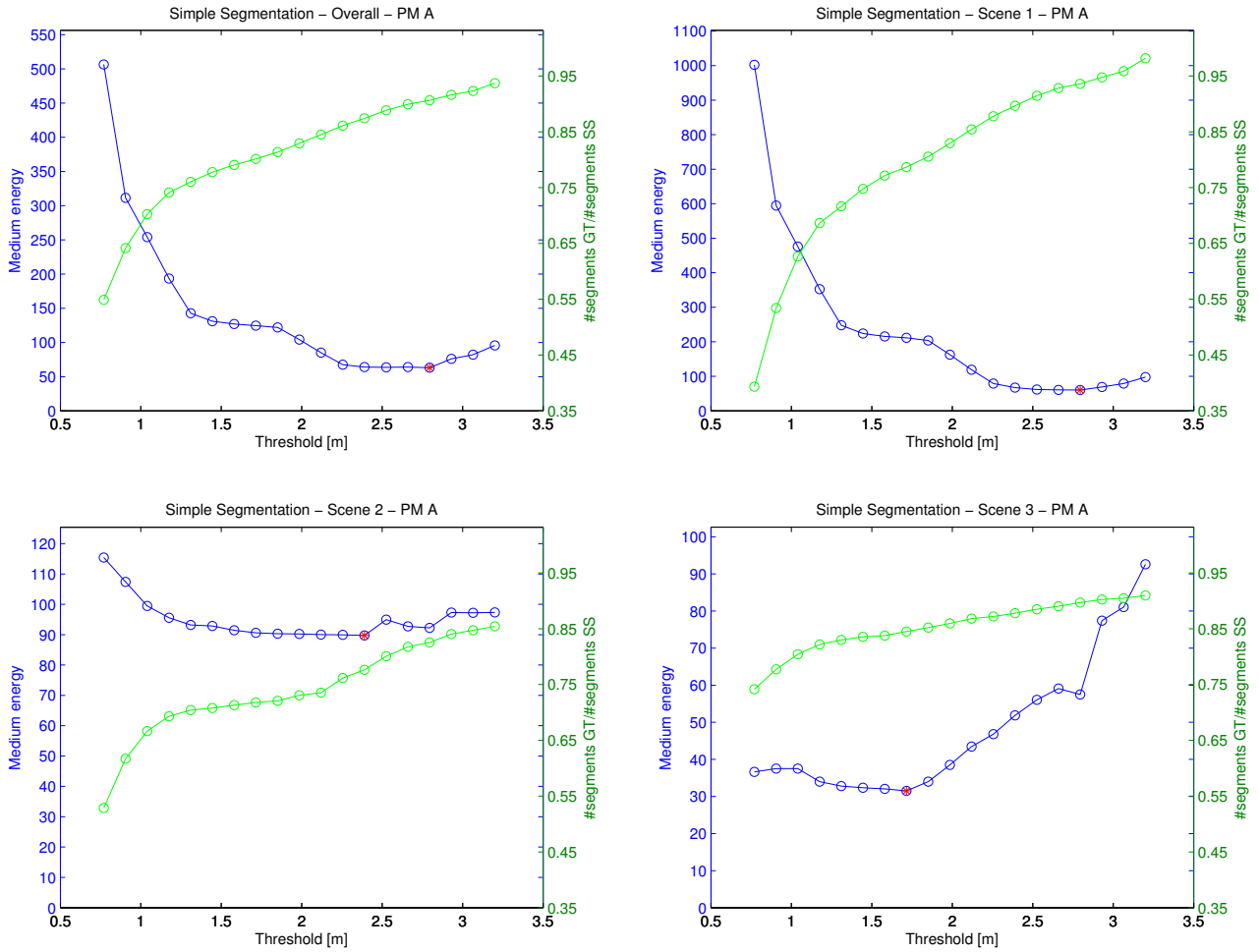


Figure 4.4: Energy given by PM A and segments ratio for the **Simple Segmentation** method.

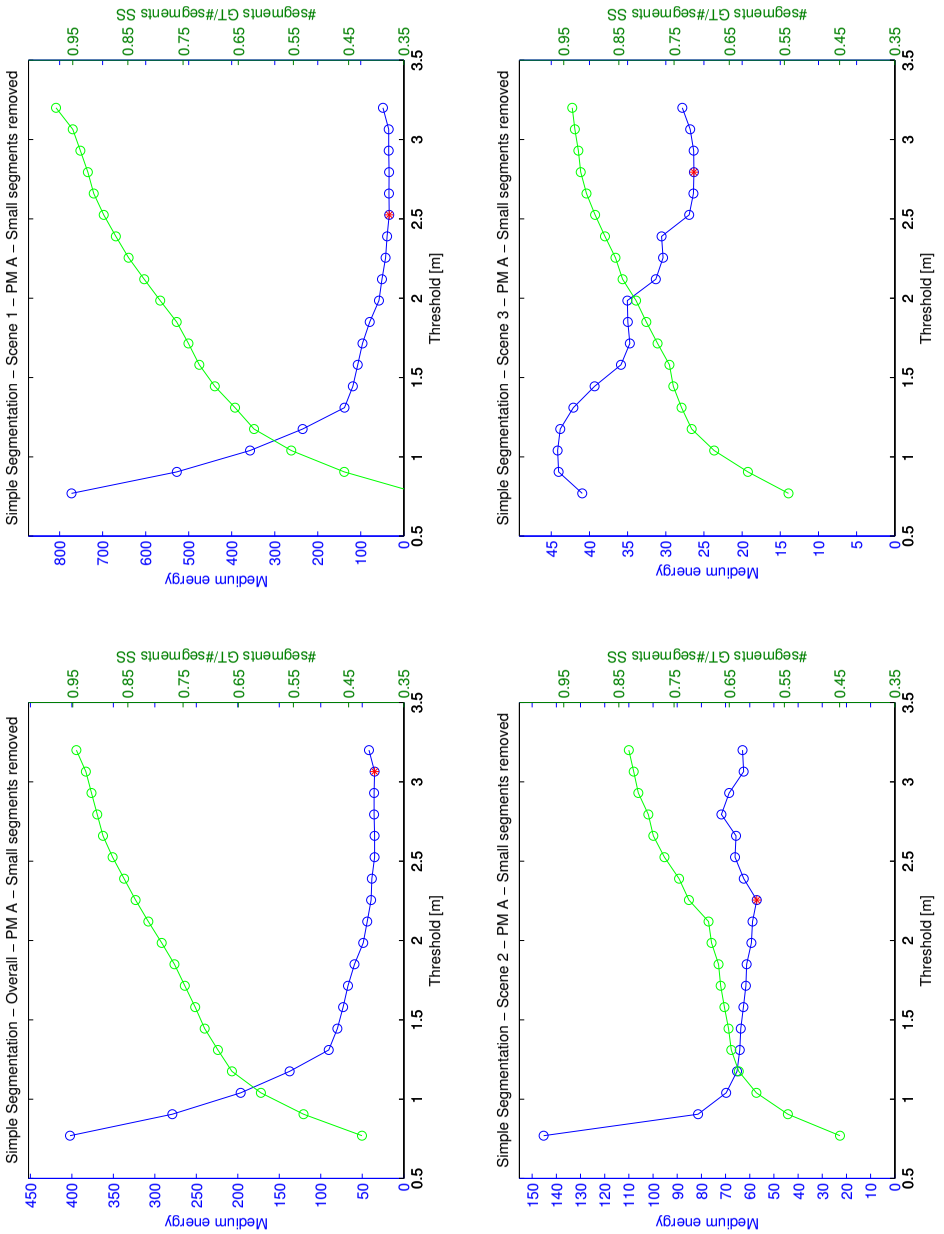


Figure 4.5: Energy given by PM A and segments ratio for the **Simple Segmentation** method; scans with small segments on the Ground-truth removed.

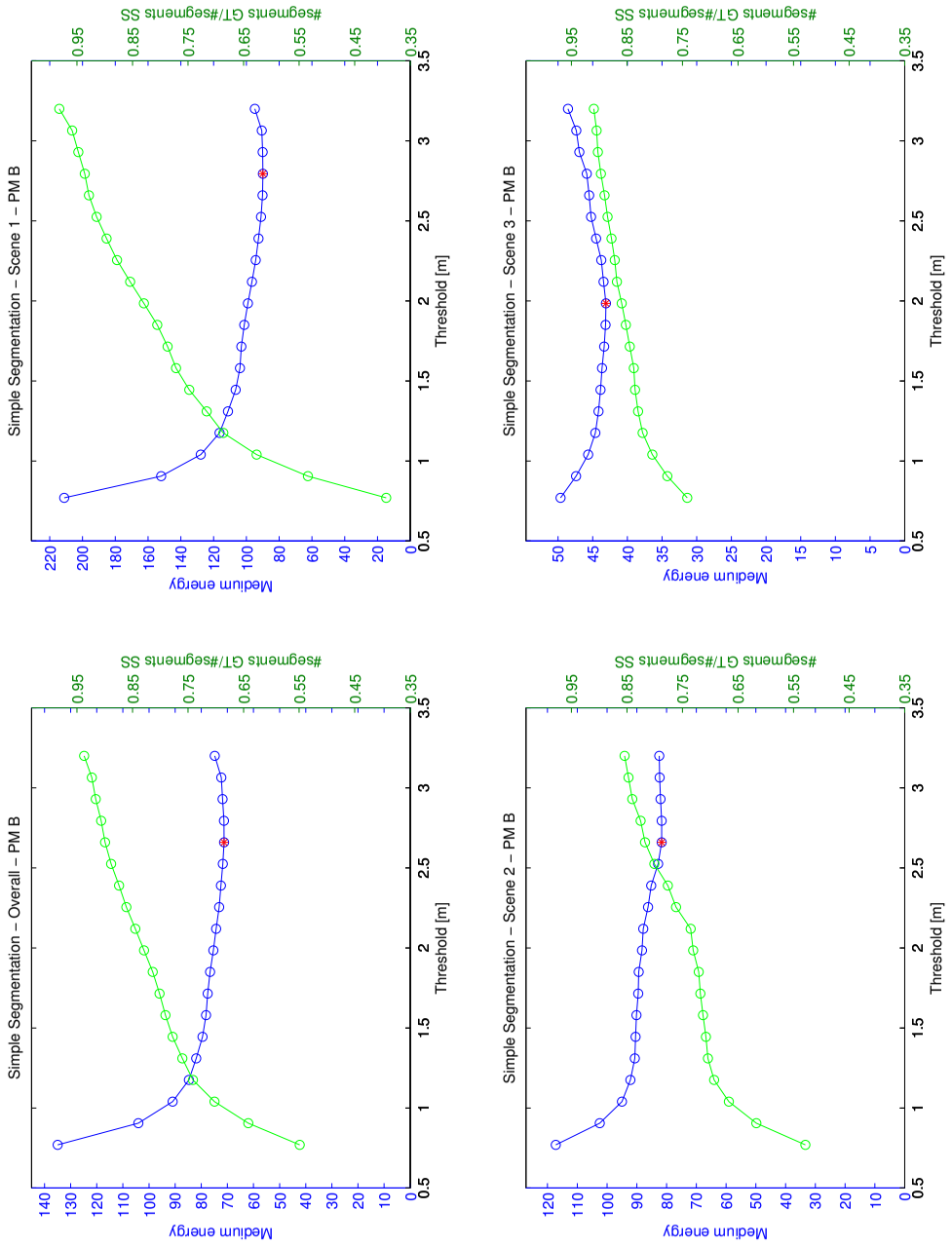


Figure 4.6: Energy given by PM B and segments ratio for the **Simple Segmentation** method.

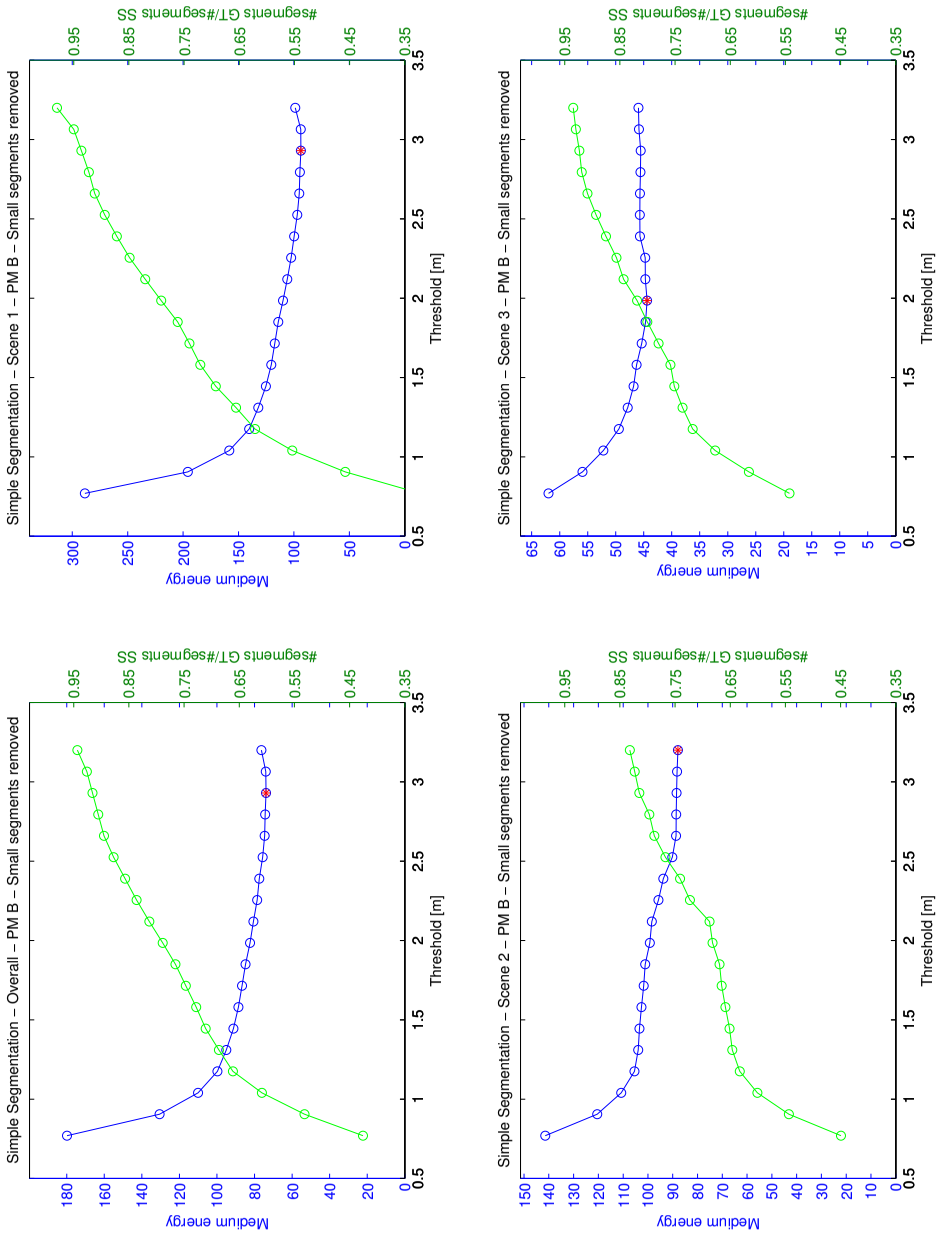


Figure 4.7: Energy given by PM B and segments ratio for the **Simple Segmentation** method; scans with small segments on the Ground-truth removed.

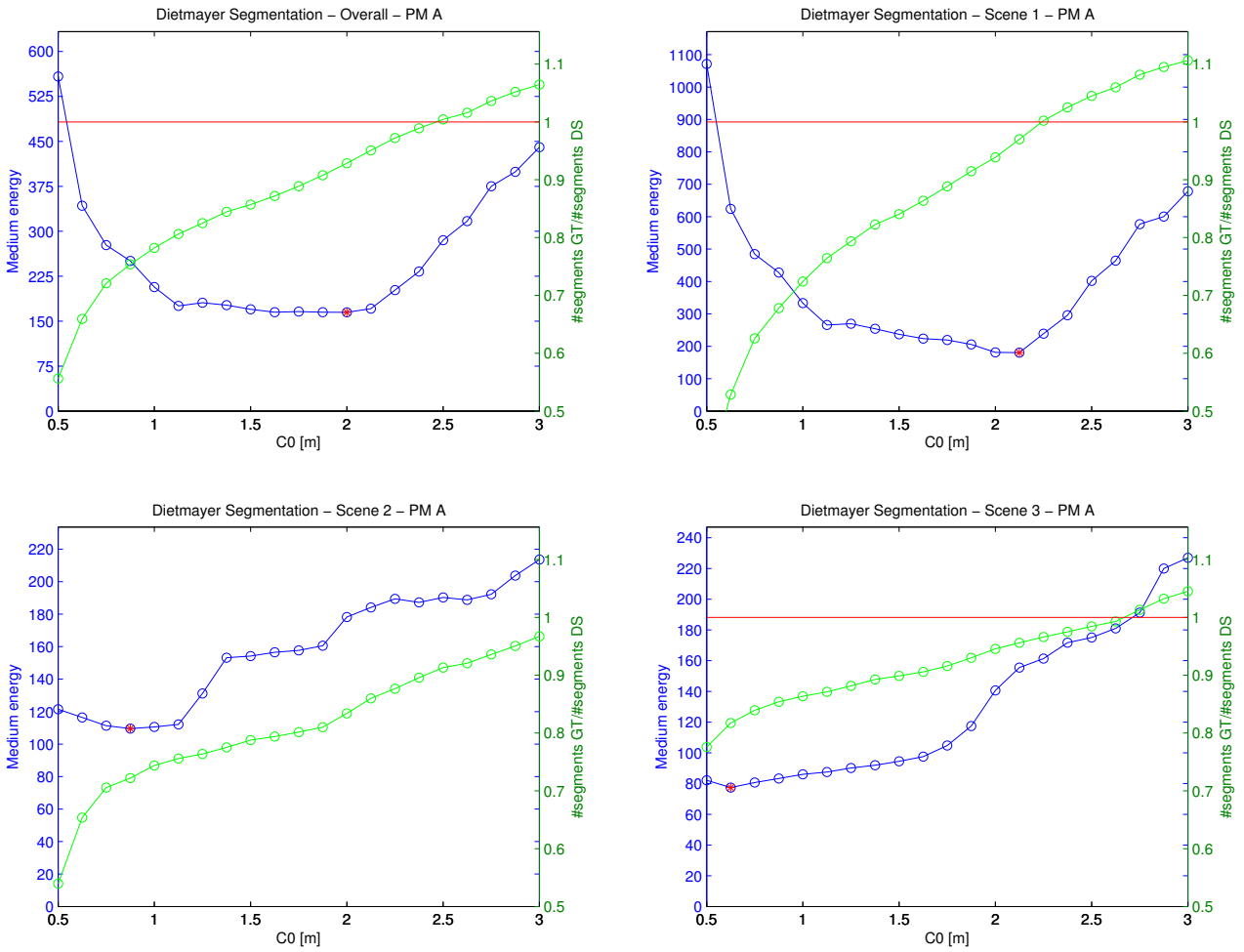


Figure 4.8: Energy given by PM A and segments ratio for the **Dietmayer Segmentation** method.

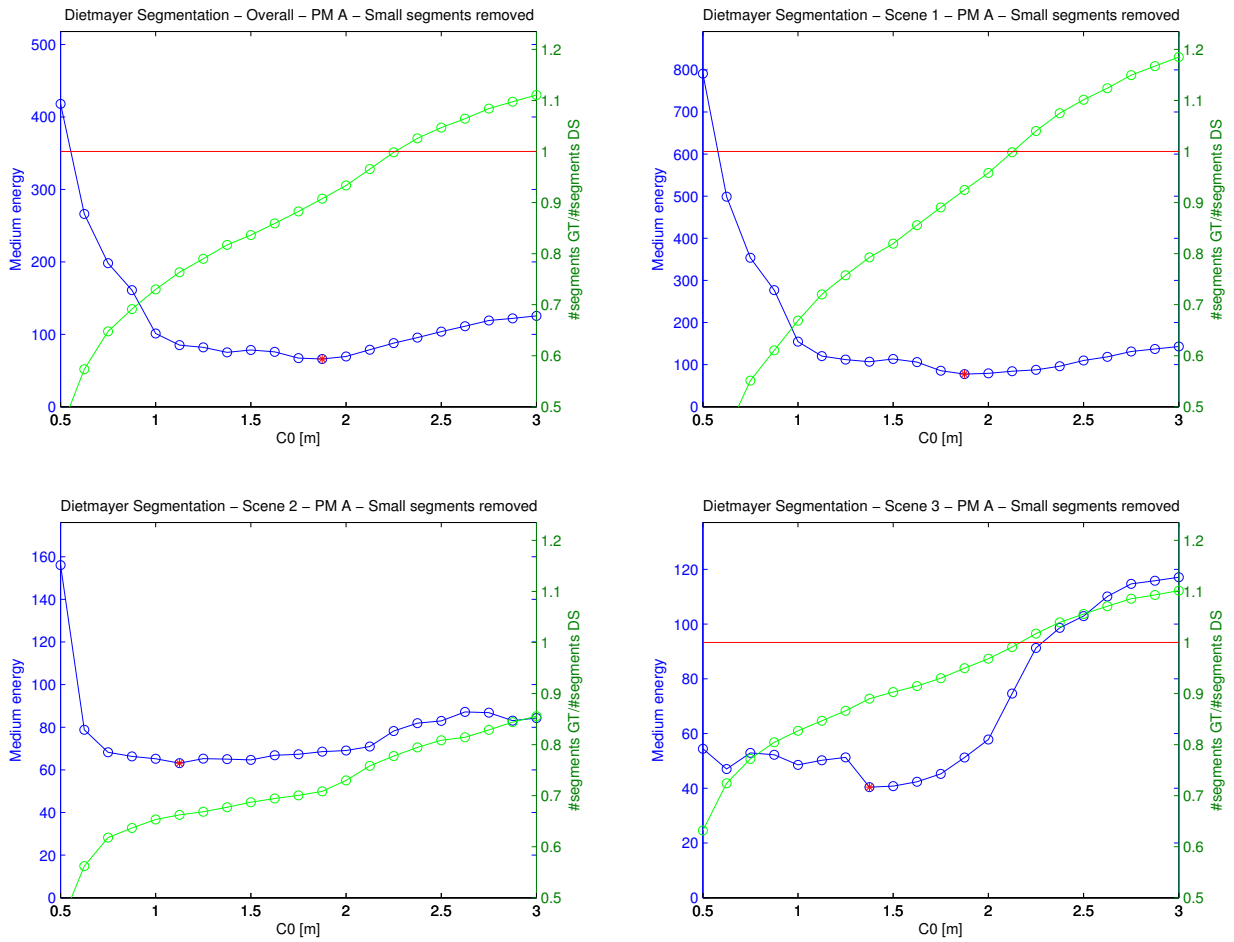


Figure 4.9: Energy given by PM A and segments ratio for the **Dietmayer Segmentation** method; scans with small segments on the Ground-truth removed.

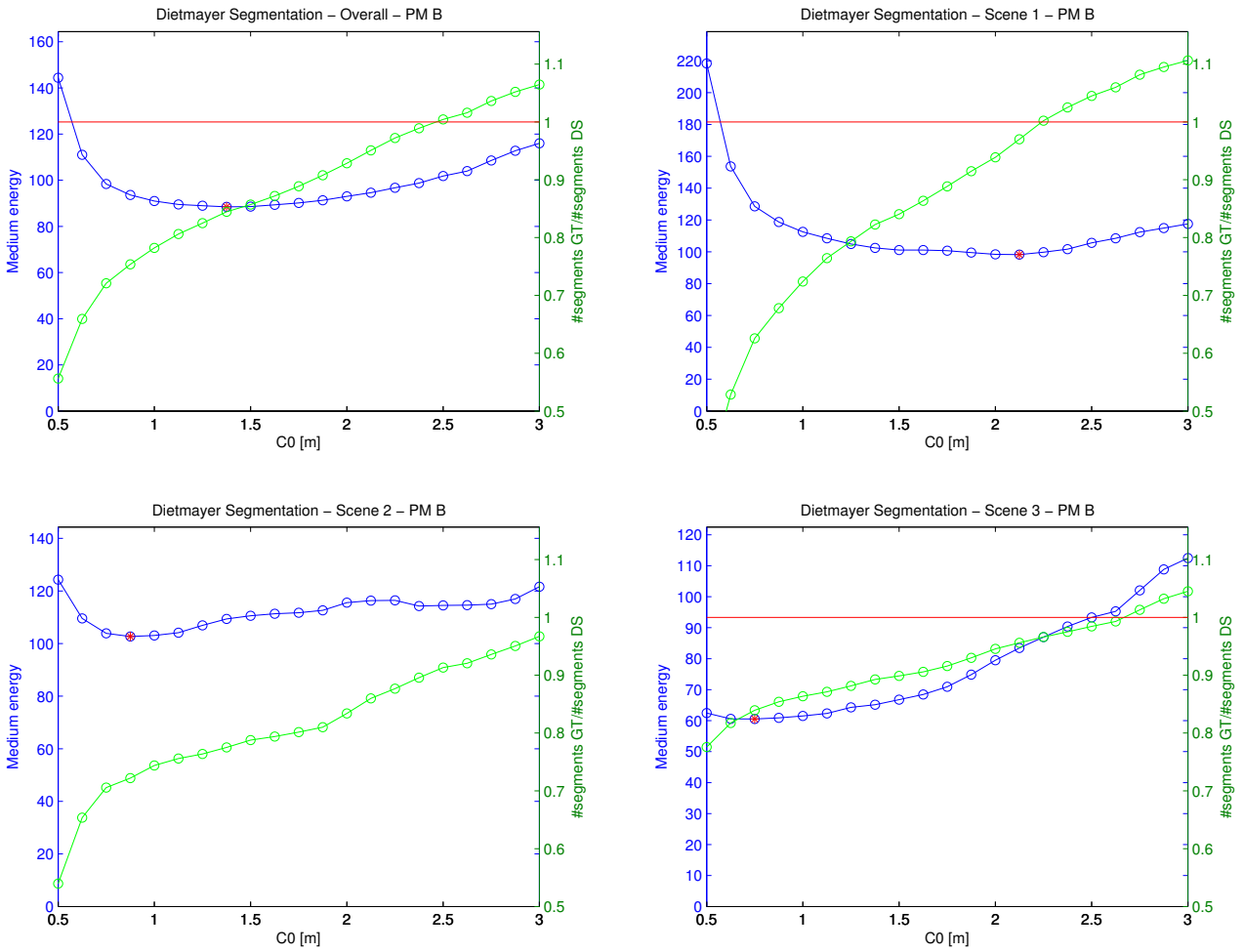


Figure 4.10: Energy given by PM B and segments ratio for the **Dietmayer Segmentation** method.

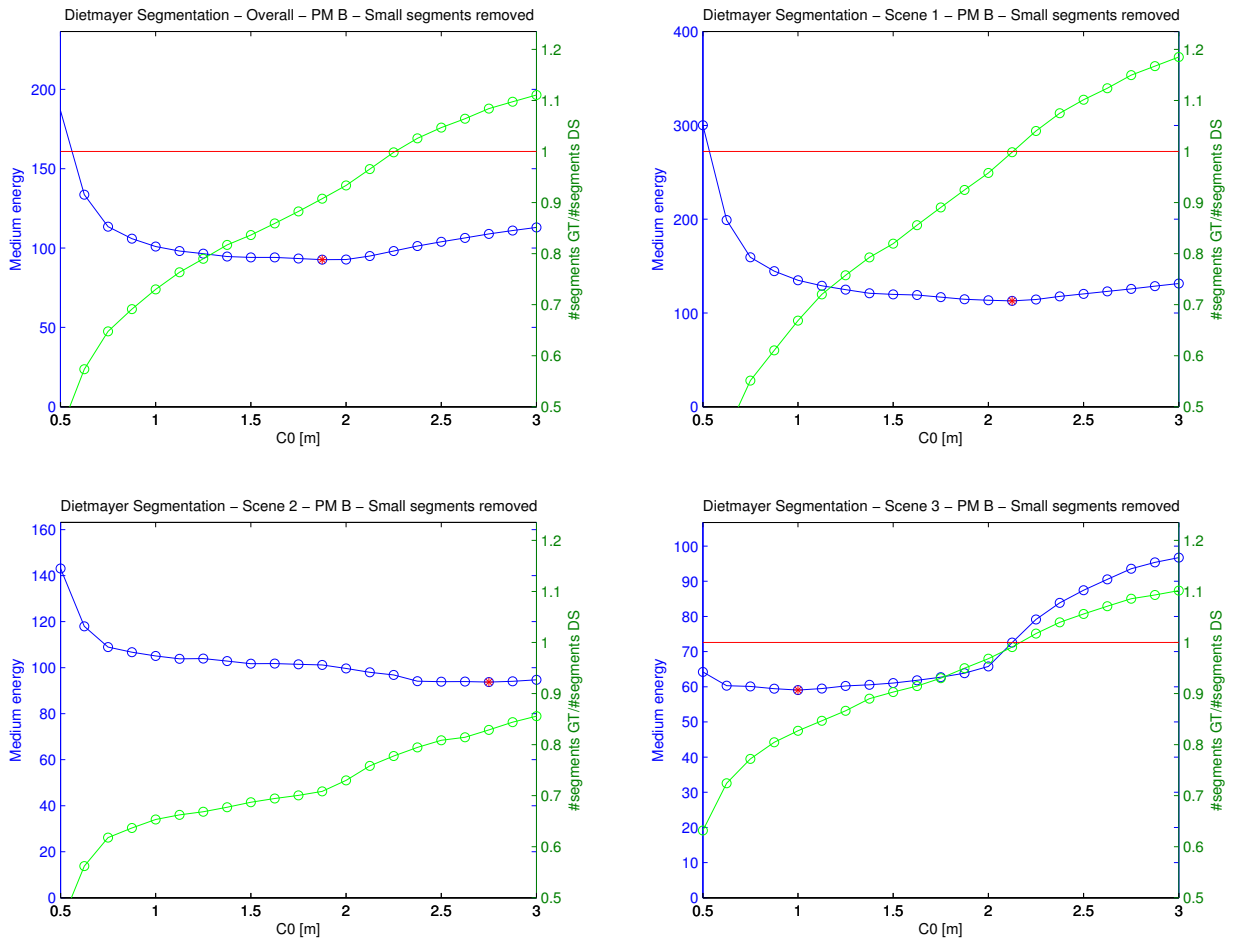


Figure 4.11: Energy given by PM B and segments ratio for the **Dietmayer Segmentation** method; scans with small segments on the Ground-truth removed.

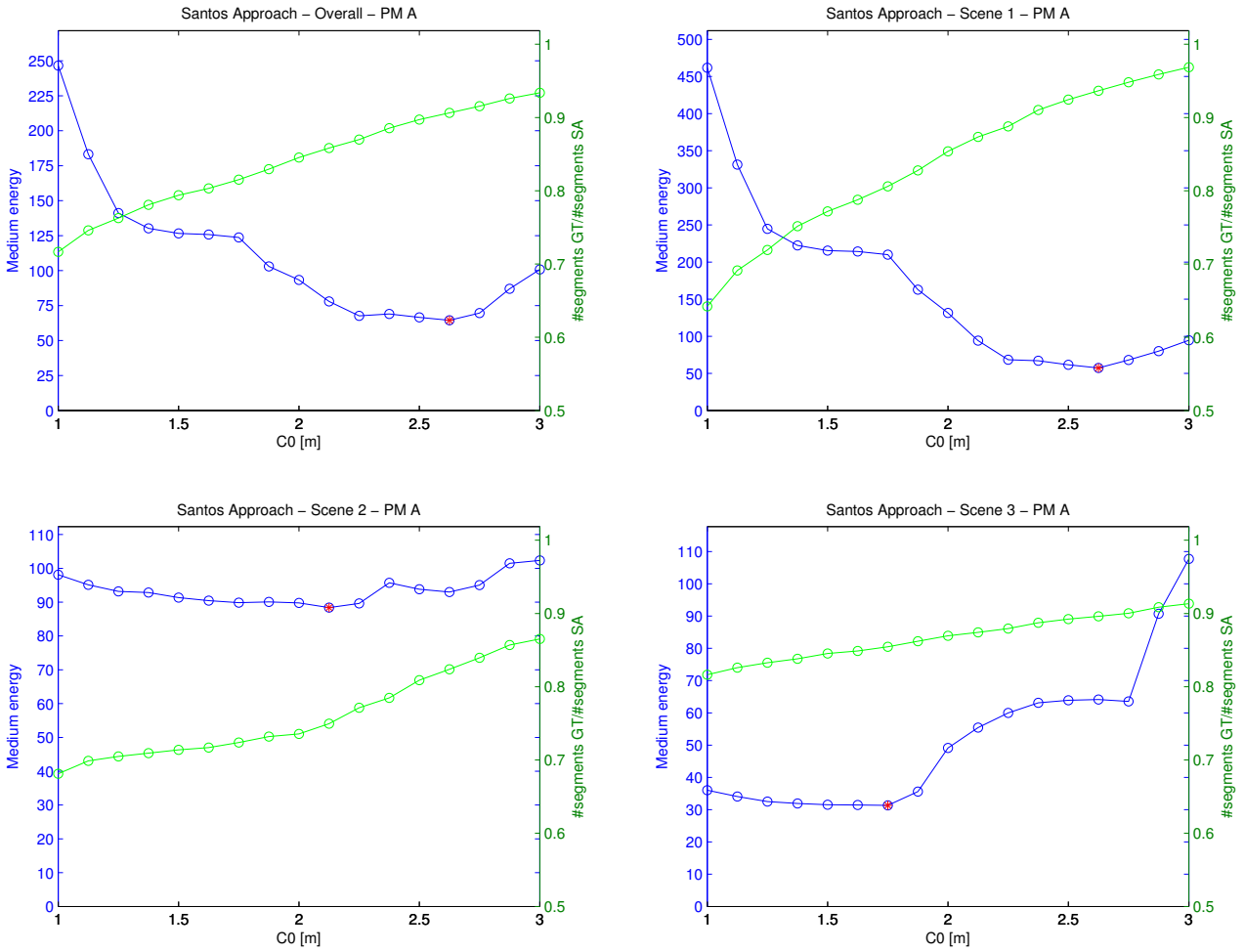


Figure 4.12: Energy given by PM A and segments ratio for the **Santos approach** method; C_0 is the free variable; $\beta = 15^o$.

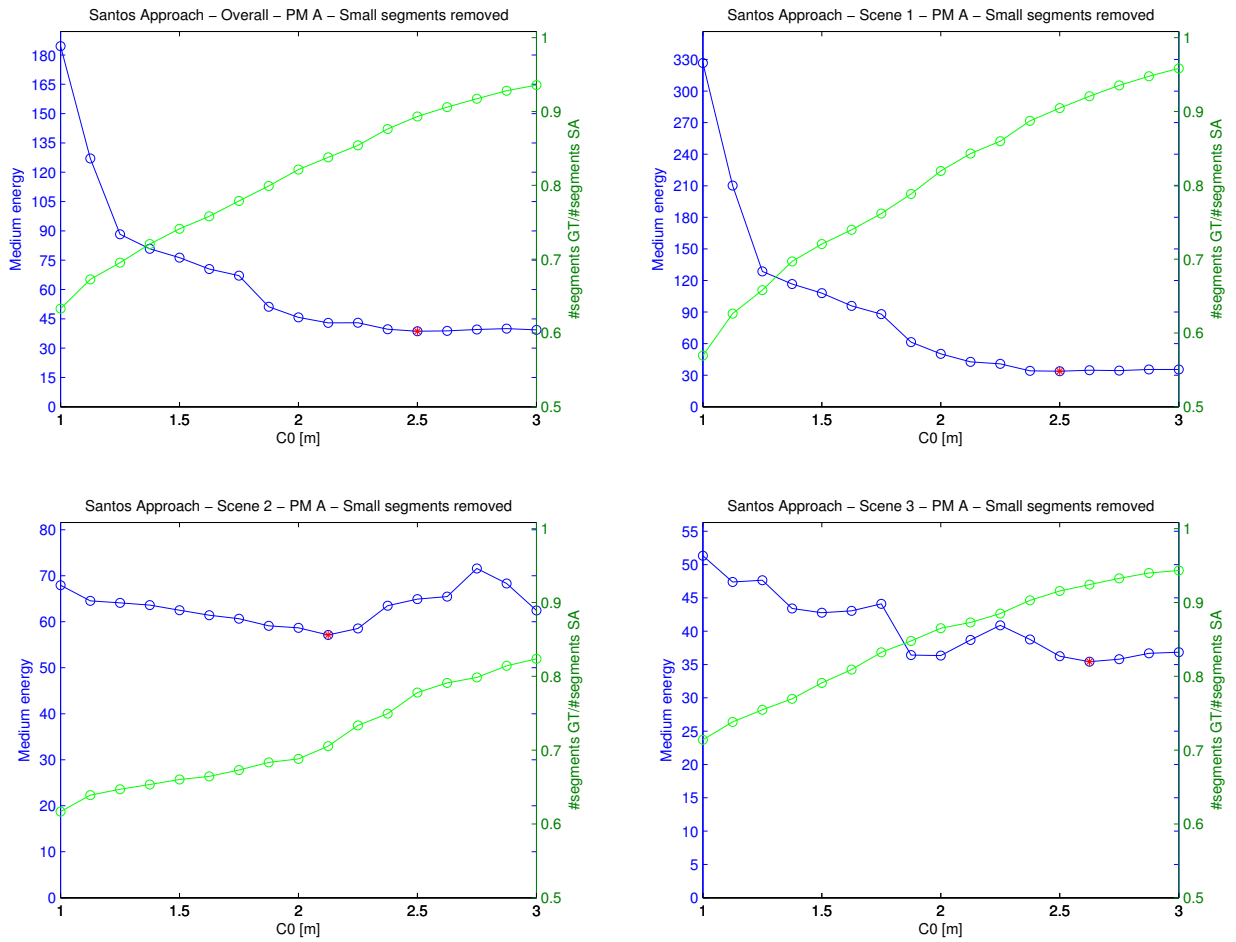


Figure 4.13: Energy given by PM A and segments ratio for the **Santos approach** method; C_0 is the free variable; $\beta = 15^o$; scans with small segments on the Ground-truth removed.

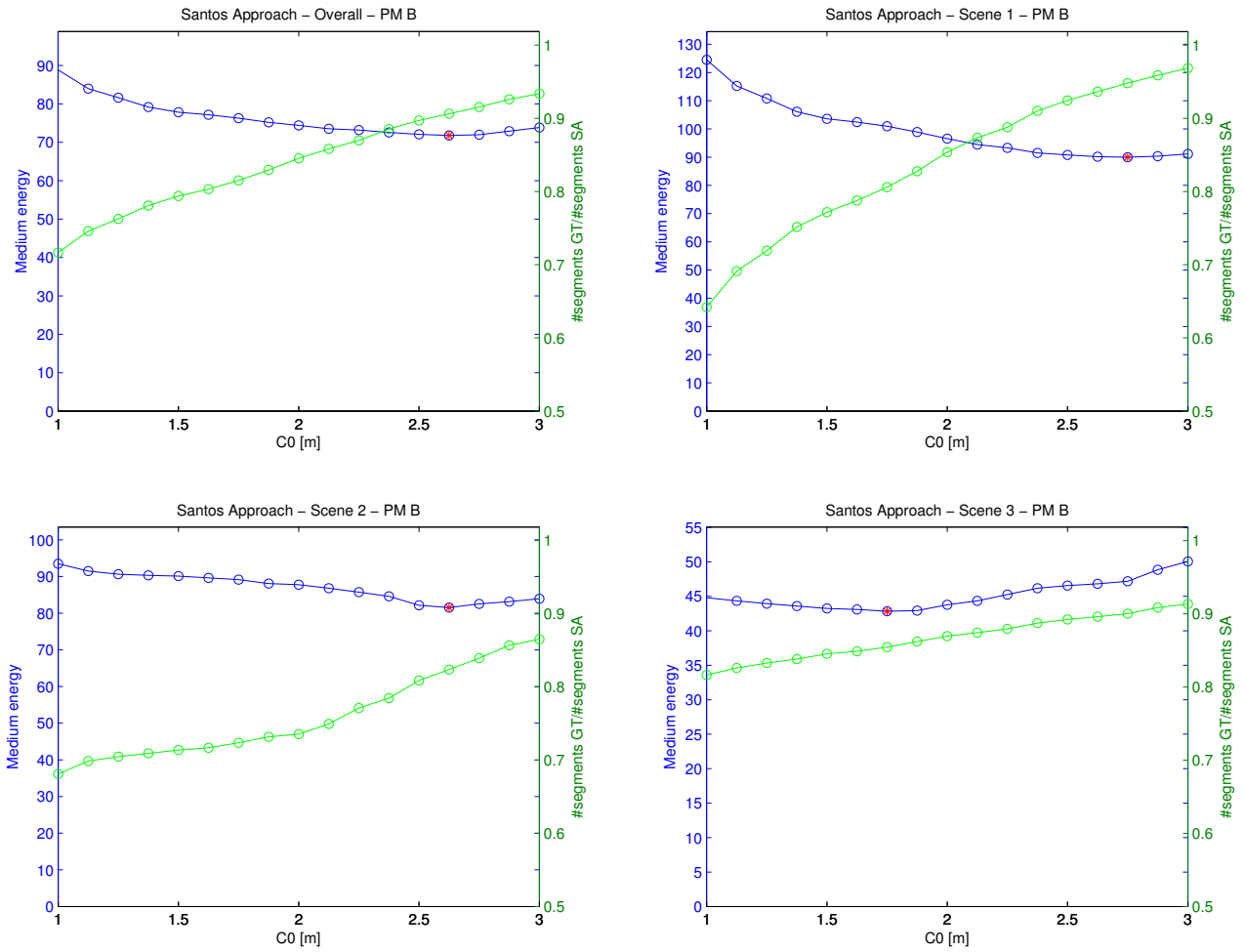


Figure 4.14: Energy given by PM B and segments ratio for the **Santos approach** method; C_0 is the free variable; $\beta = 15^o$.

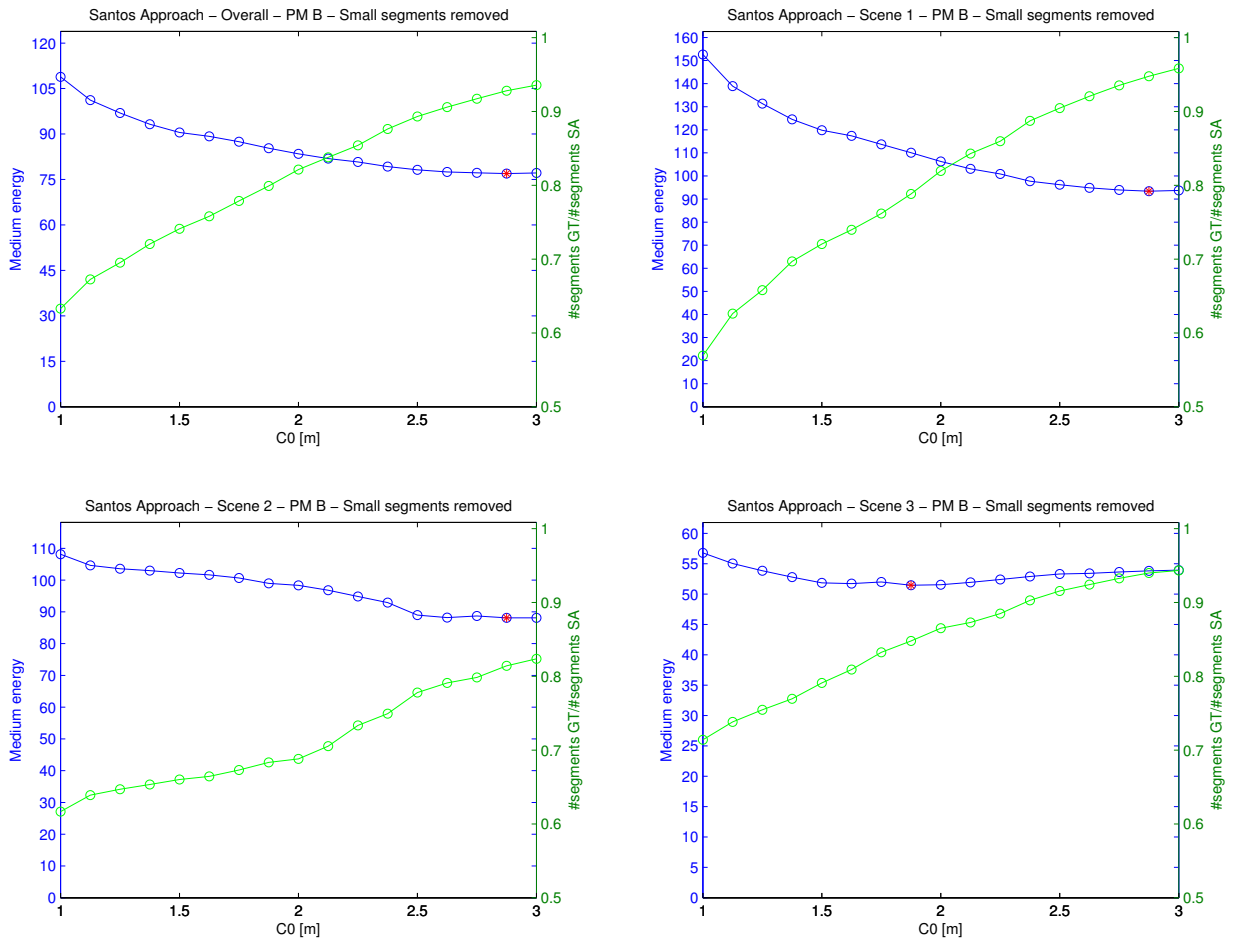


Figure 4.15: Energy given by PM B and segments ratio for the **Santos approach** method; C_0 is the free variable; $\beta = 15^o$; scans with small segments on the Ground-truth removed.

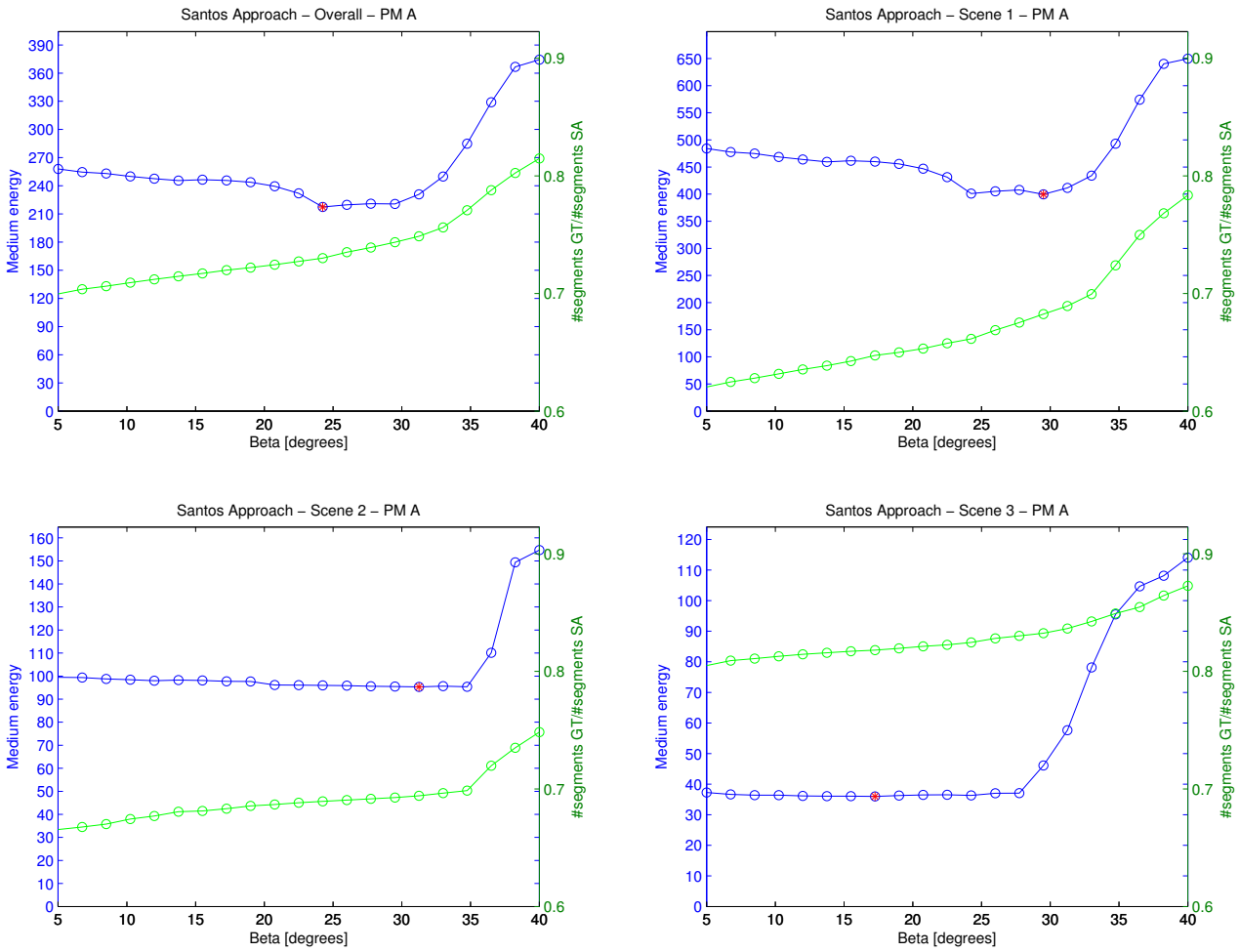


Figure 4.16: Energy given by PM A and segments ratio for the **Santos approach** method; β is the free variable; $C_0 = 1.0$ m.

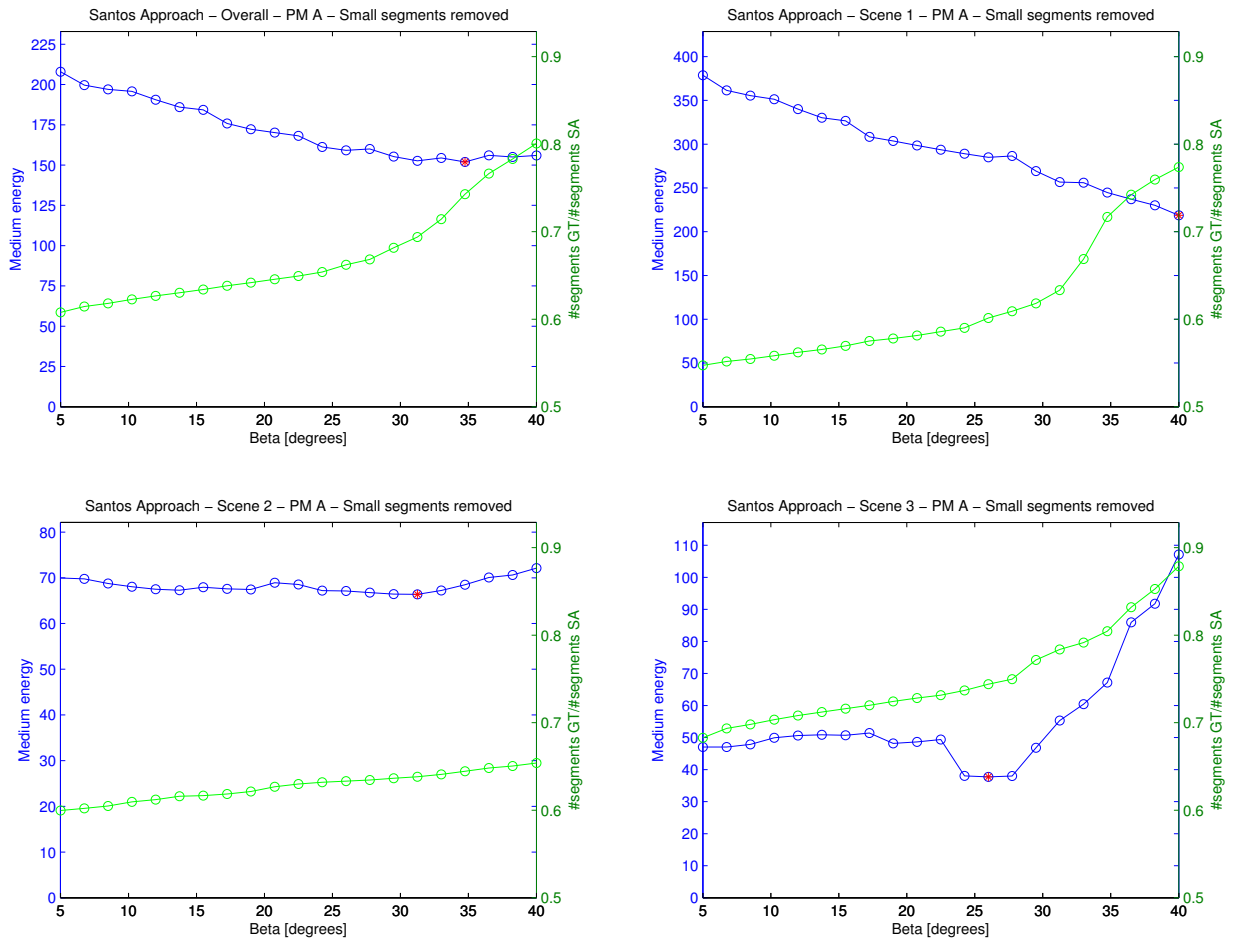


Figure 4.17: Energy given by PM A and segments ratio for the **Santos approach** method; β is the free variable; $C_0 = 1.0$ m; scans with small segments on the Ground-truth removed.

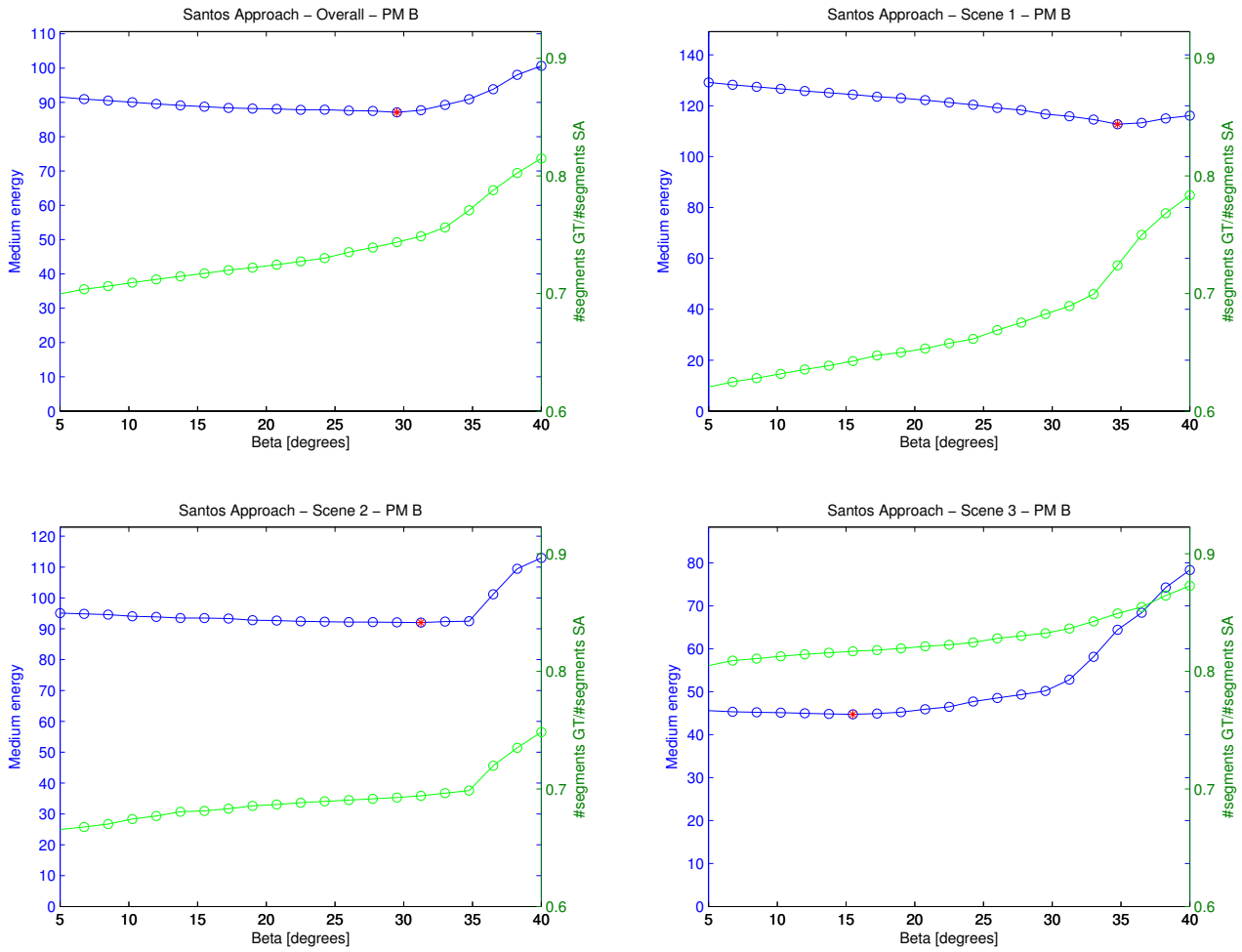


Figure 4.18: Energy given by PM B and segments ratio for the **Santos approach** method; β is the free variable; $C_0 = 1.0$ m.

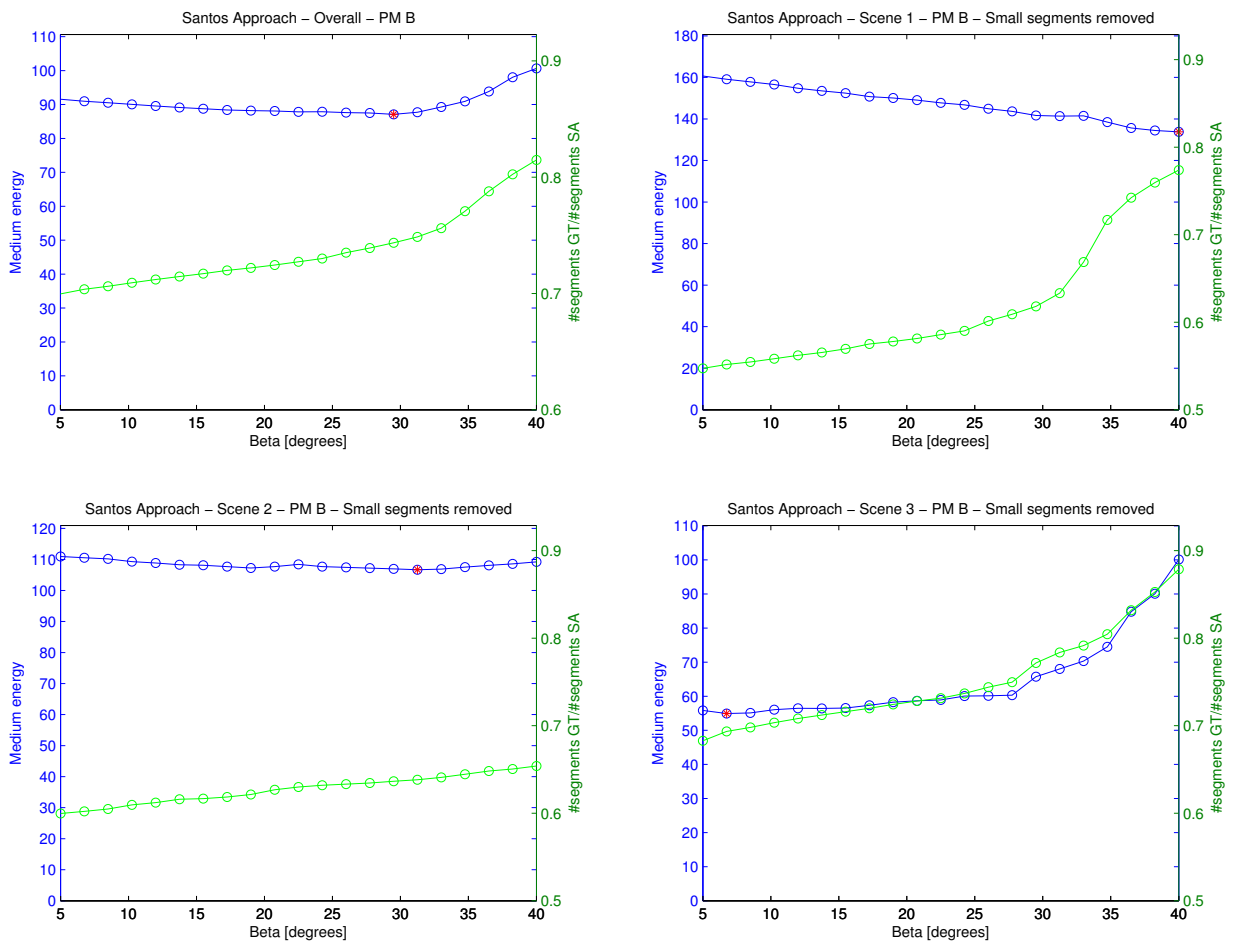


Figure 4.19: Energy given by PM B and segments ratio for the **Santos approach** method; β is the free variable; $C_0 = 1.0$ m; scans with small segments on the Ground-truth removed.

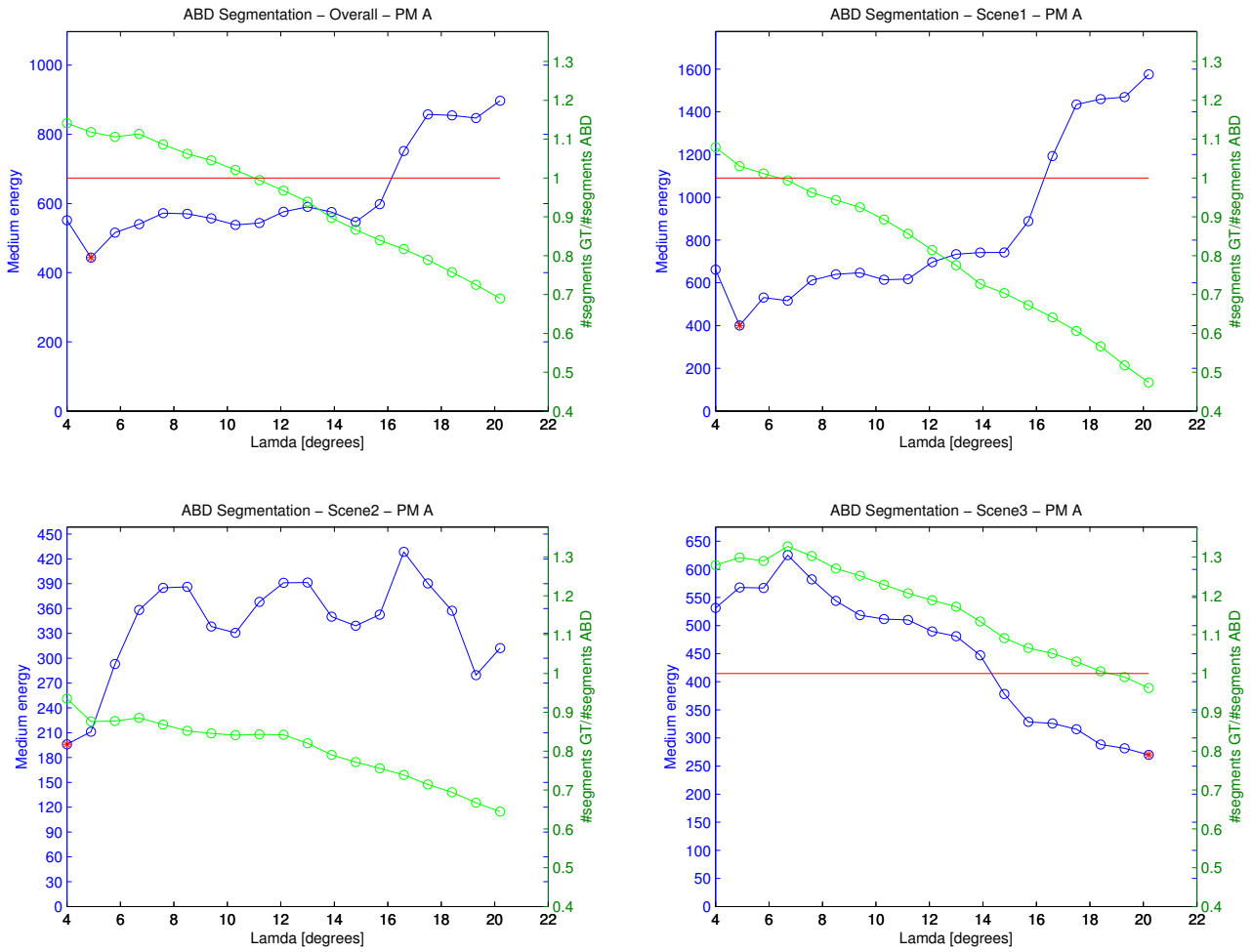


Figure 4.20: Energy given by PM A and segments ratio for the **Adaptive Breakpoint Detector** method.

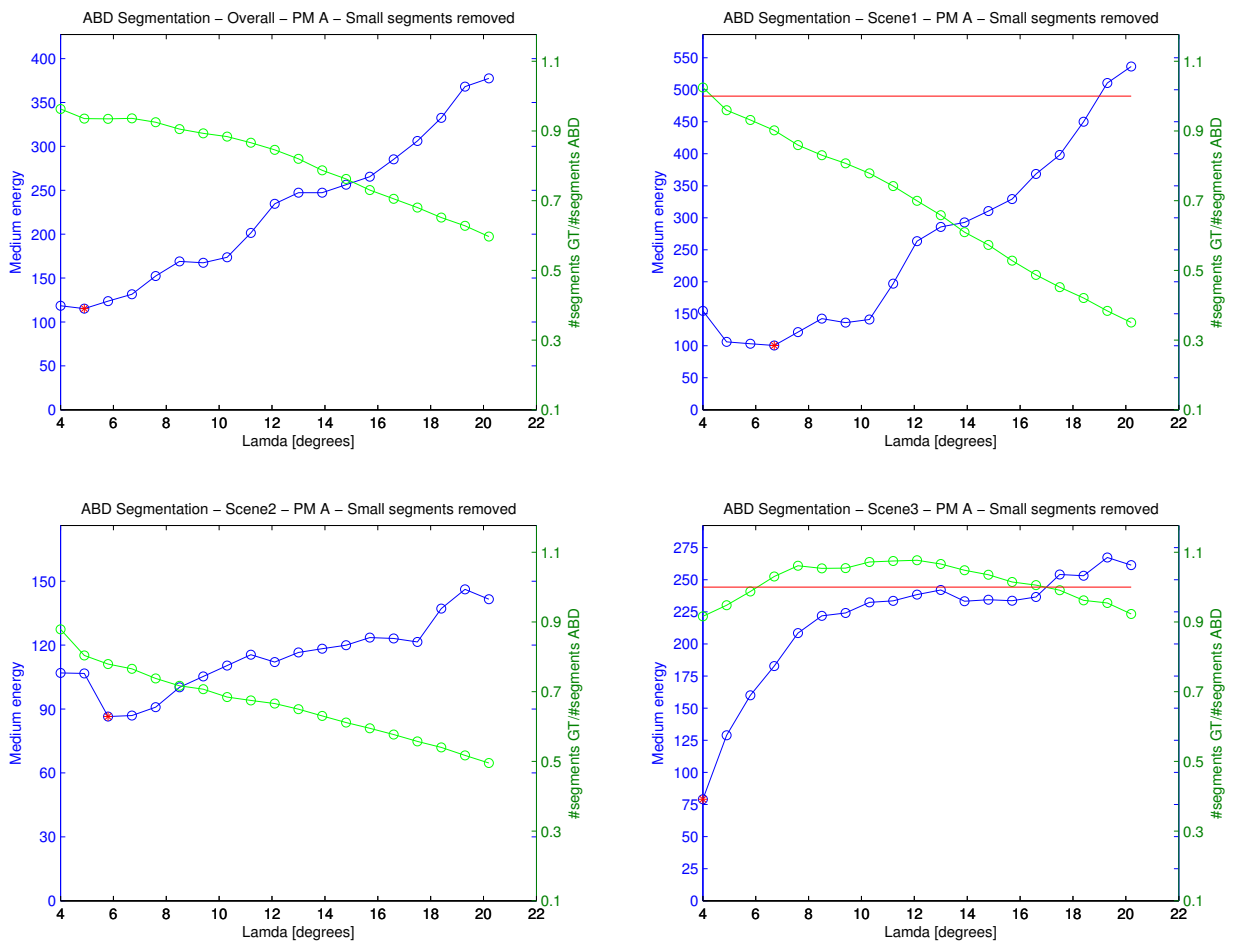


Figure 4.21: Energy given by PM A and segments ratio for the **Adaptive Breakpoint Detector** method; scans with small segments on the Ground-truth removed.

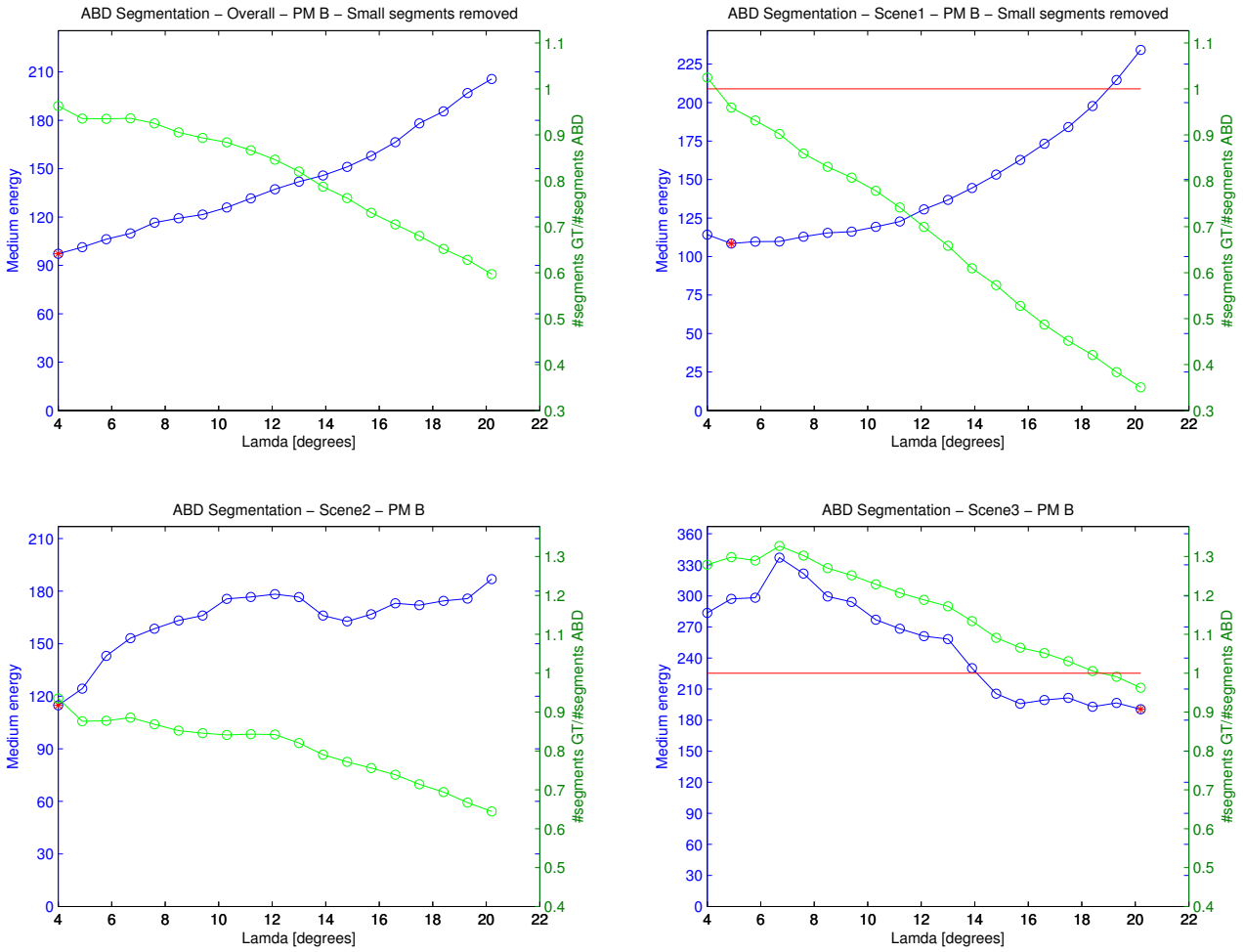


Figure 4.22: Energy given by PM B and segments ratio for the **Adaptive Breakpoint Detector** method.

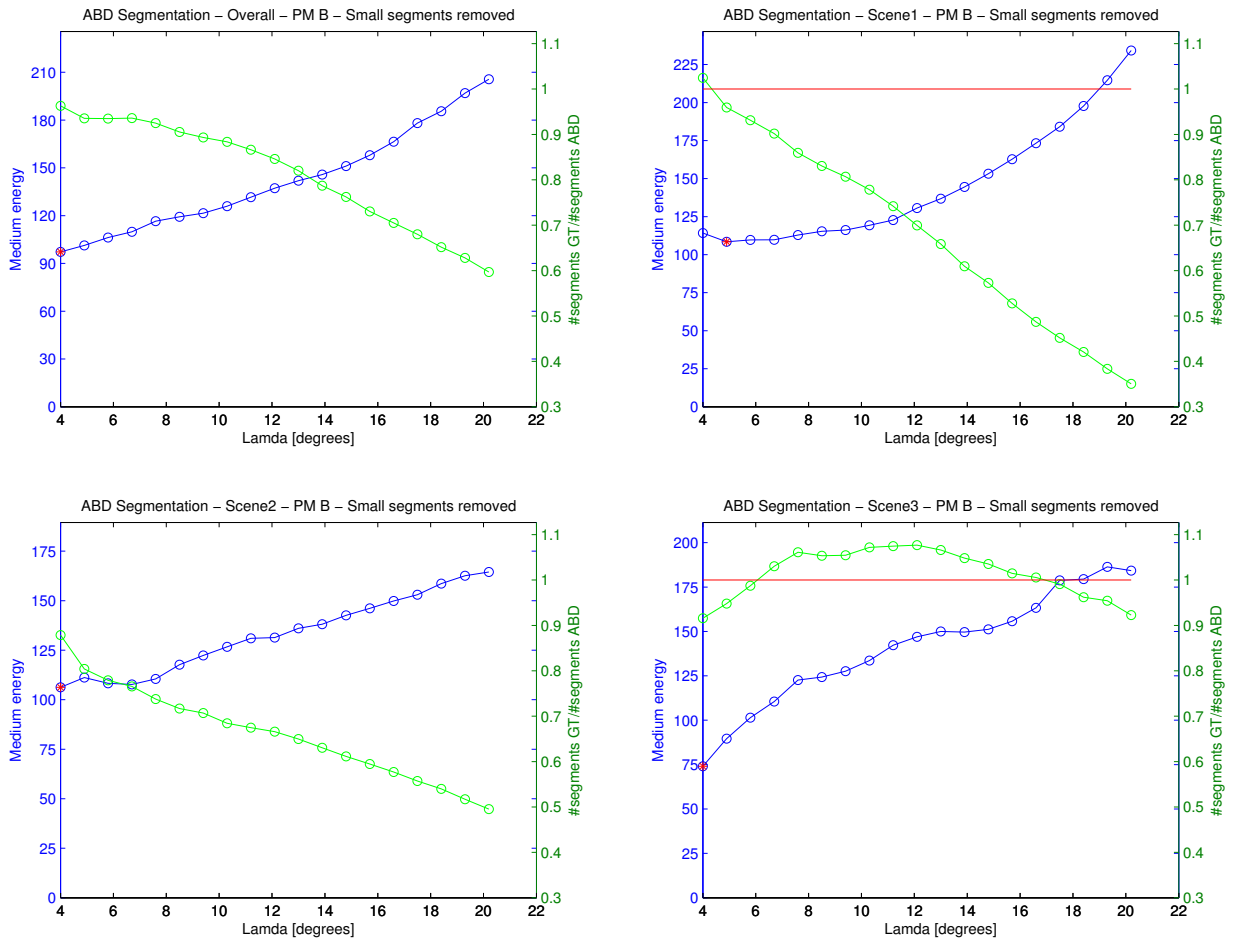


Figure 4.23: Energy given by PM B and segments ratio for the **Adaptive Breakpoint Detector** method; scans with small segments on the Ground-truth removed.

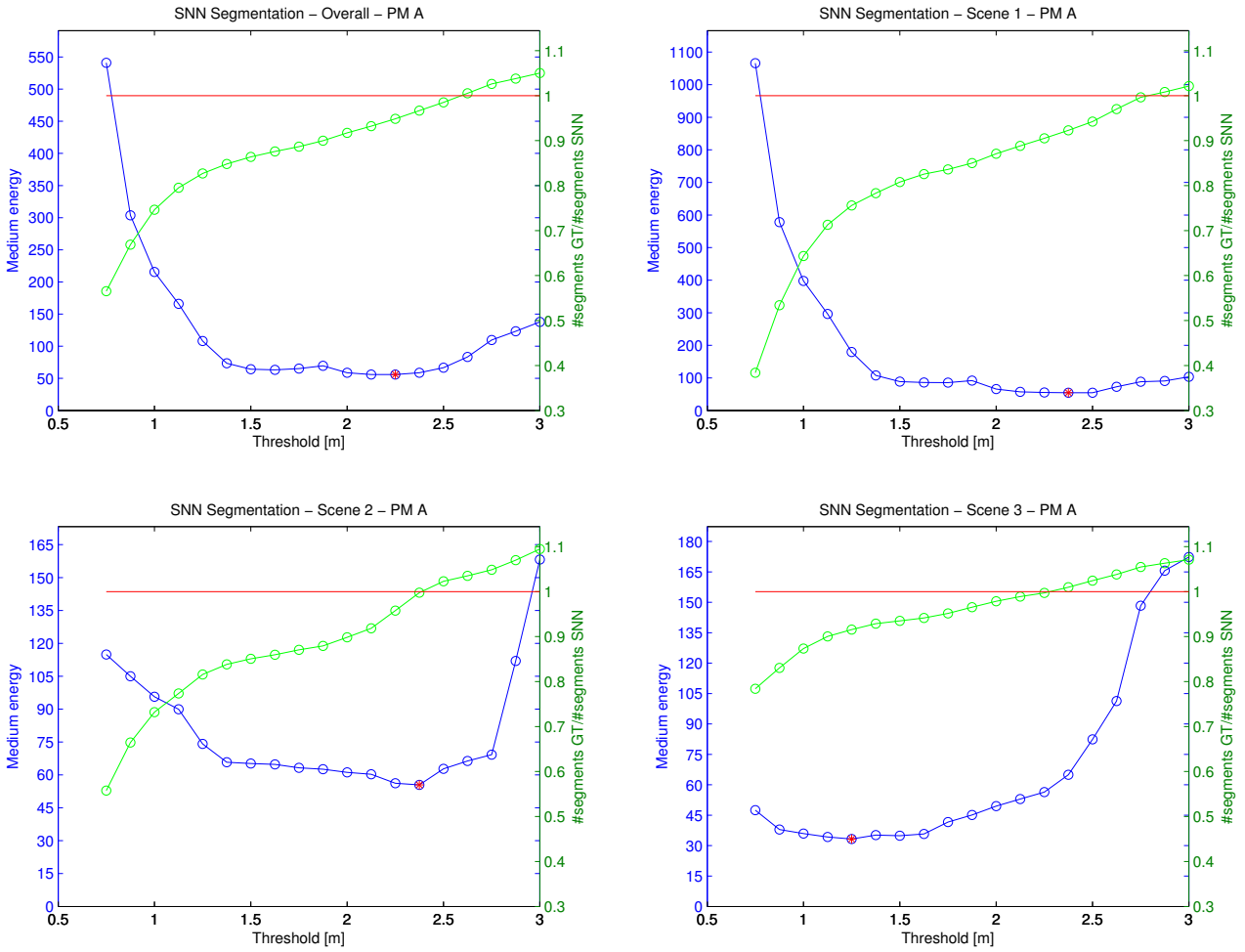


Figure 4.24: Energy given by PM A and segments ratio for the **Spatial Nearest Neighbour** method.

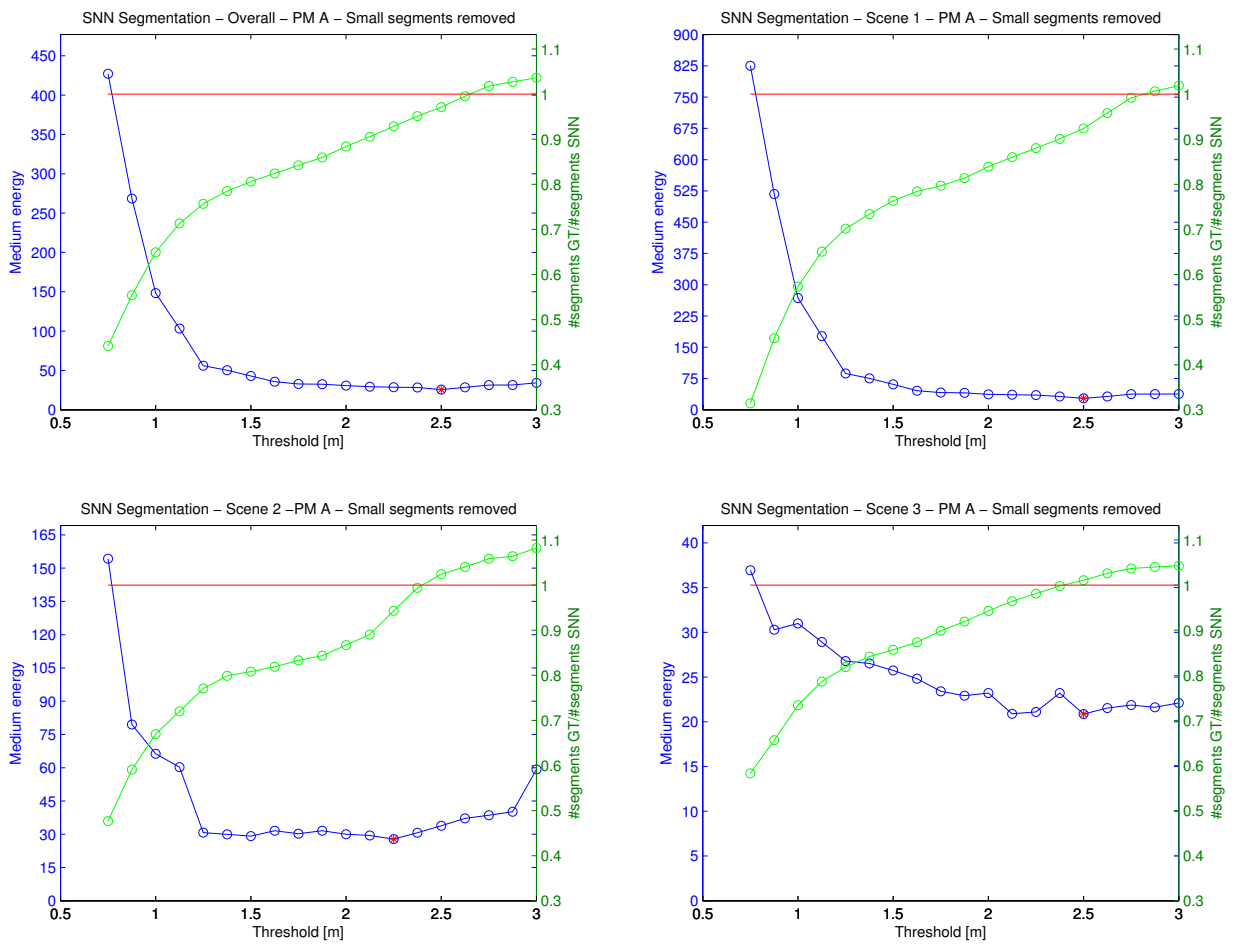


Figure 4.25: Energy given by PM A and segments ratio for the **Spatial Nearest Neighbour** method; scans with small segments on the Ground-truth removed.

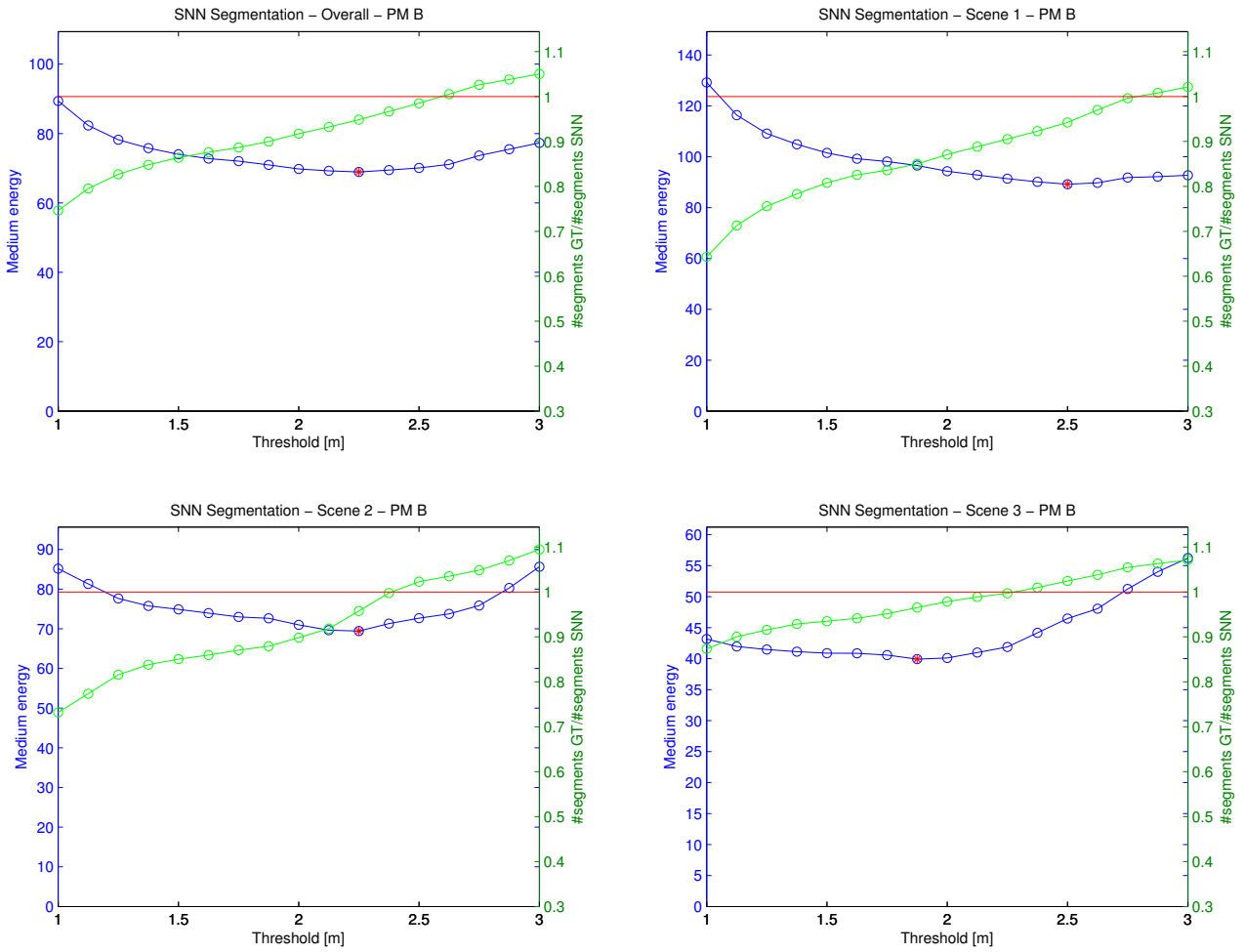


Figure 4.26: Energy given by PM B and segments ratio for the **Spatial Nearest Neighbour** method.

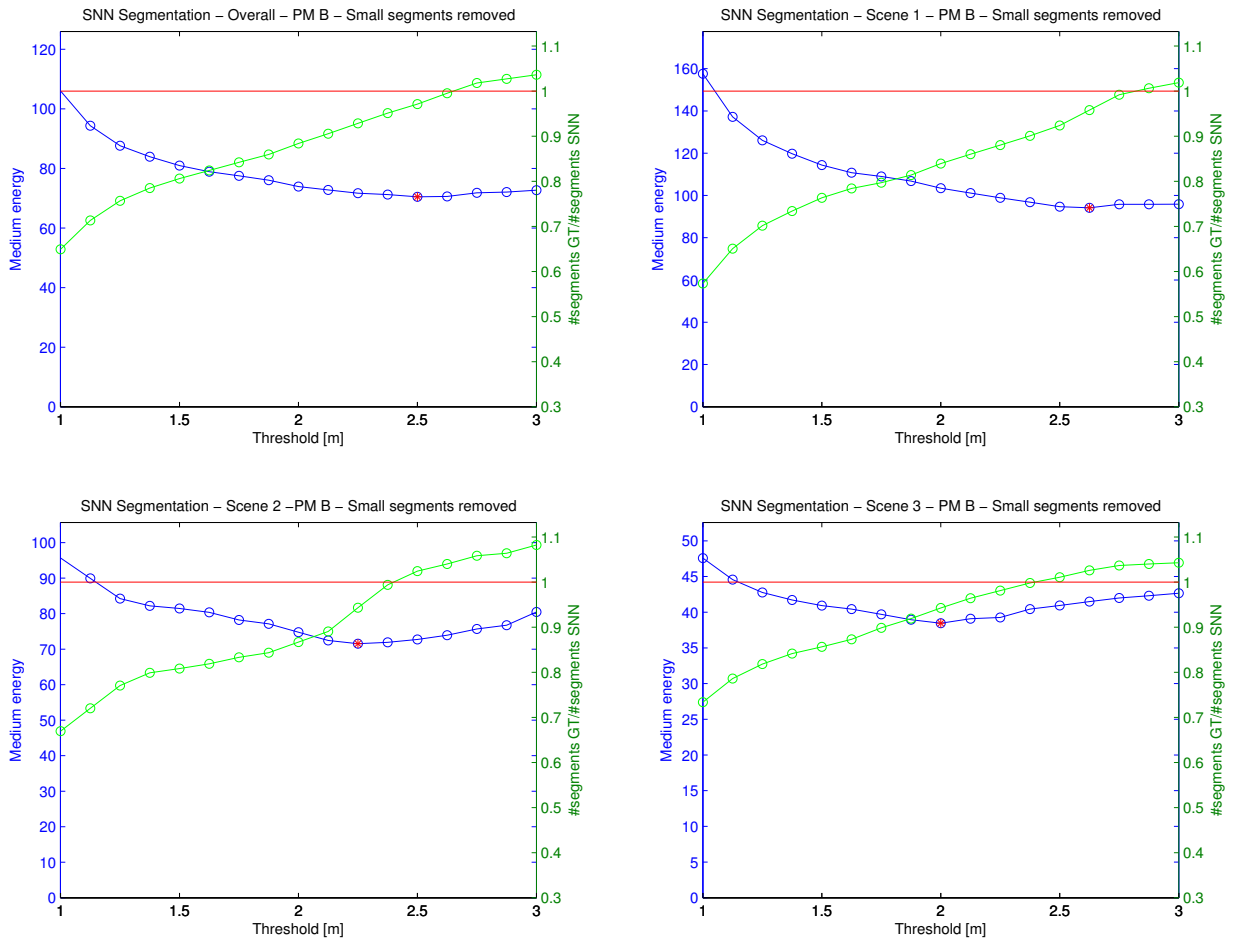


Figure 4.27: Energy given by PM B and segments ratio for the **Spatial Nearest Neighbour** method; scans with small segments on the Ground-truth removed.

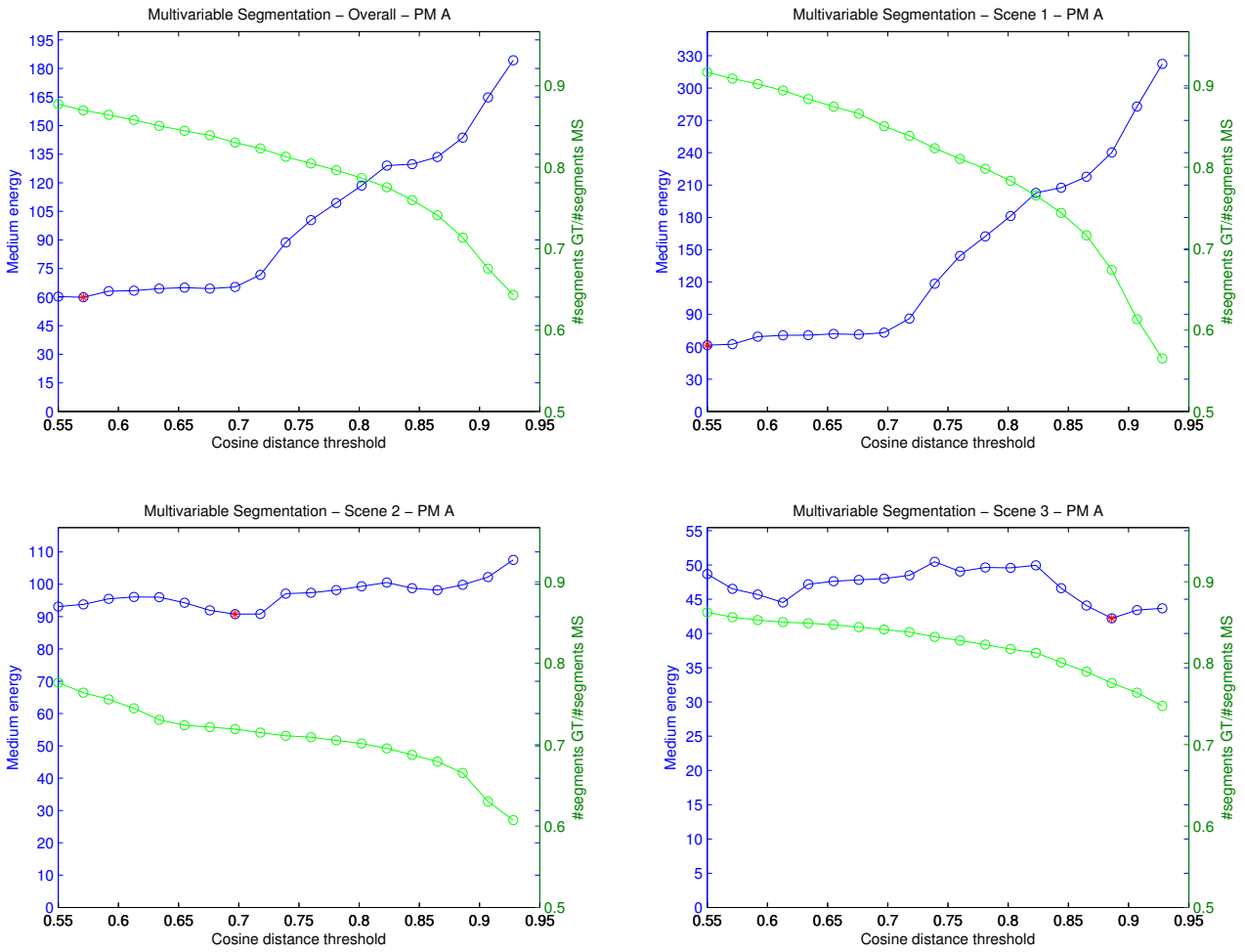


Figure 4.28: Energy given by PM A and segments ratio for the **Multivariable Segmentation** method.

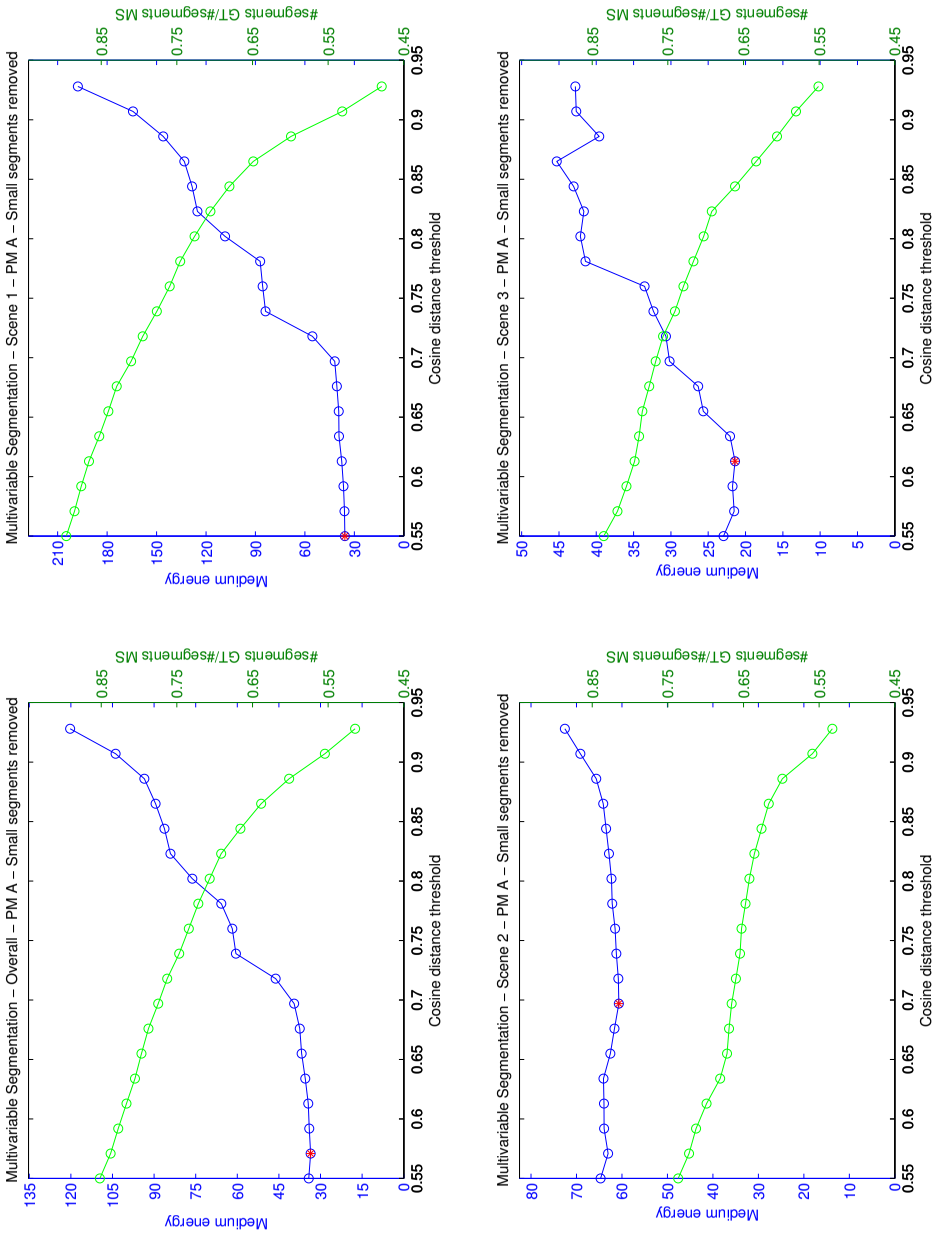


Figure 4.29: Energy given by PM A and segments ratio for the **Multivariable Segmentation** method; scans with small segments on the Ground-truth removed.

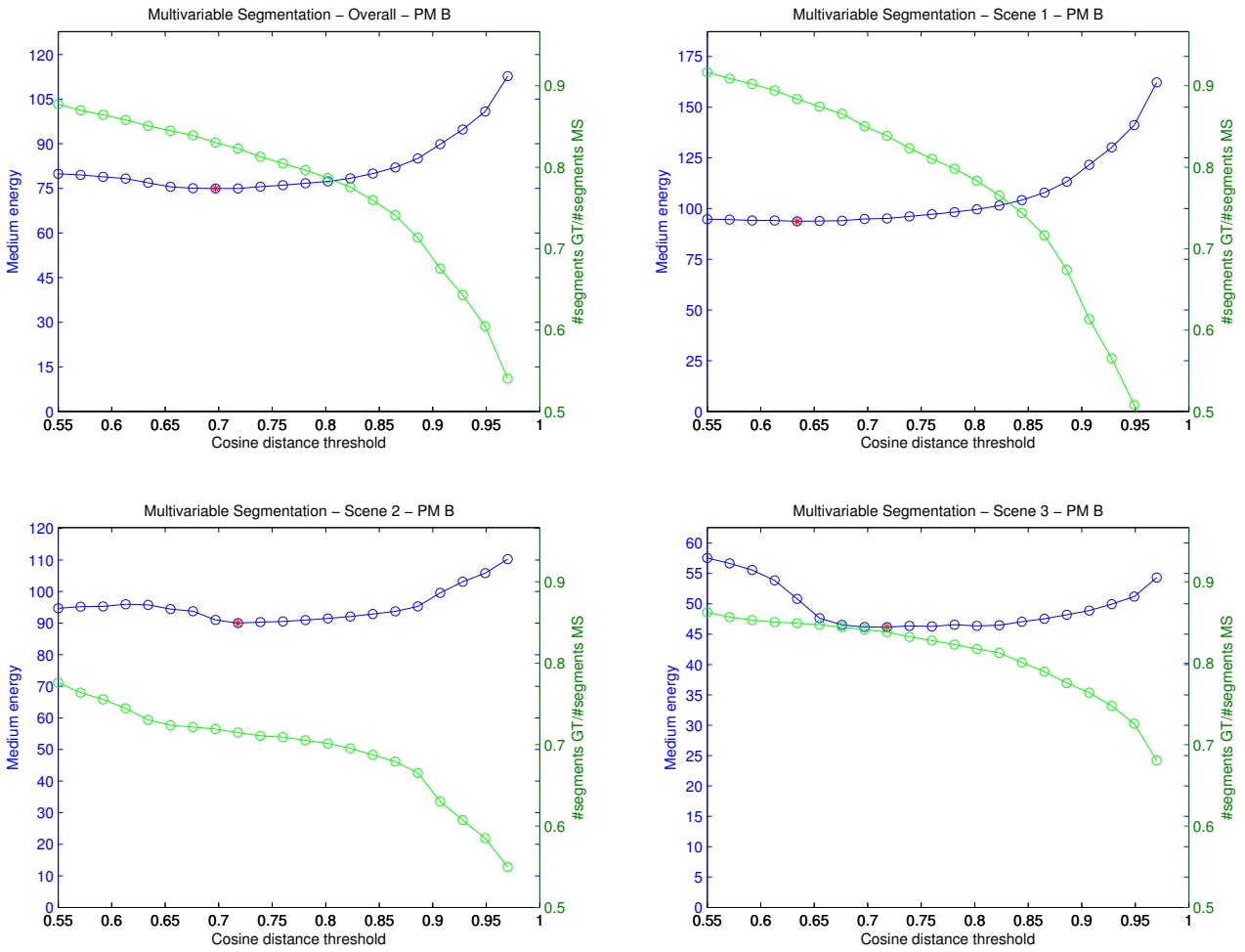


Figure 4.30: Energy given by PM B and segments ratio for the **Multivariable Segmentation** method.

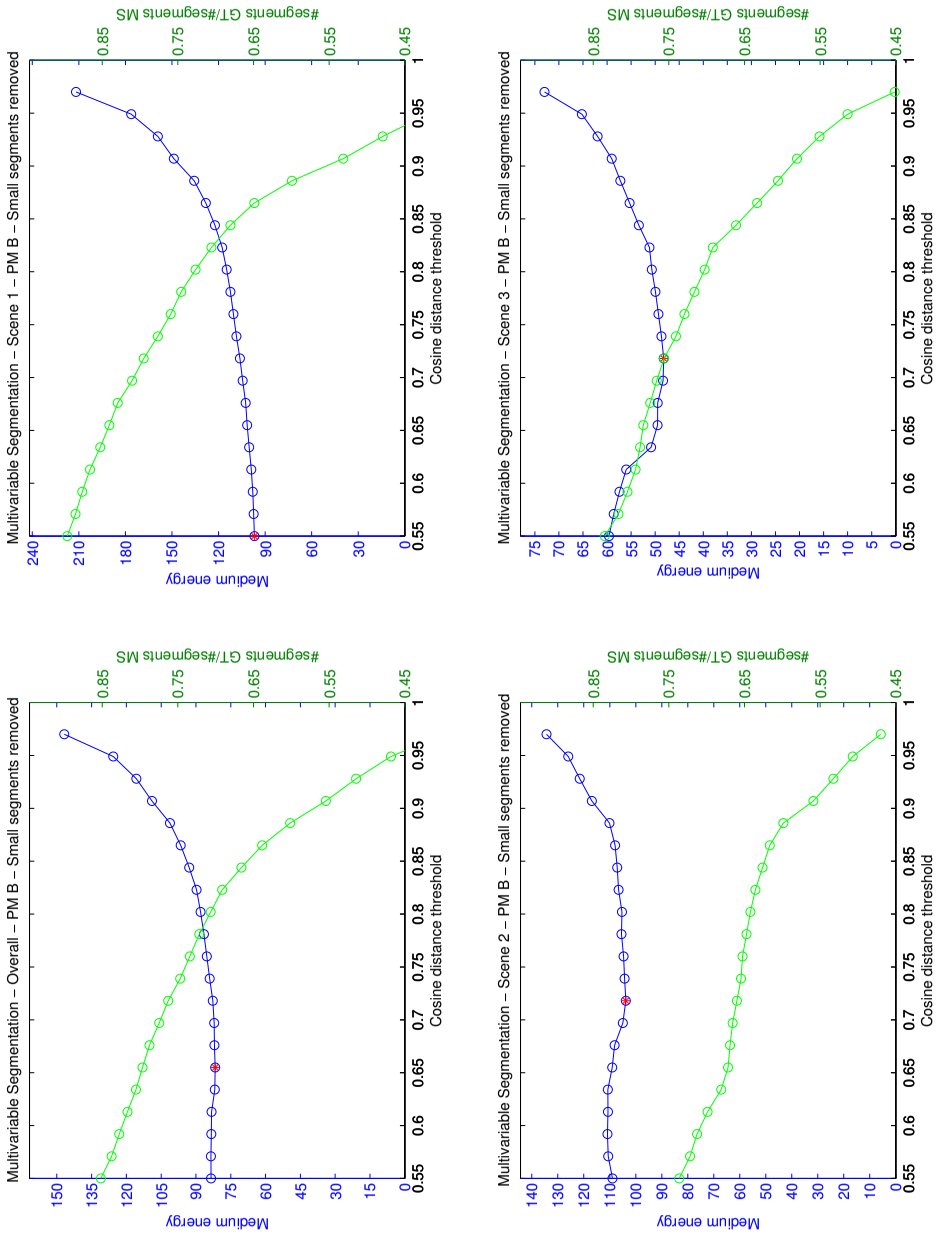


Figure 4.31: Energy given by PM B and segments ratio for the **Multivariable Segmentation** method; scans with small segments on the Ground-truth removed.

Table 4.1: PM A results. **Note:** BT - Best threshold

			SS ($D_{thd}[m]$)	DS ($C_o[m]$)	SA ($C_o[m]$)	SA ($\beta[^o]$)	ABD ($\lambda[^o]$)	SNN ($D_{thd}[m]$)	MS ($CosD_{thd}$)	Algorithms with best score
PM A - Ordinary scans	Scene 1	BT	2.795	2.125	2.625	29.500	4.900	2.375	0.550	SNN; SA
		Energy BT	60.243	180.346	57.326	399.558	400.593	54.362	61.477	
	Scene 2	BT	2.390	0.875	2.125	31.250	4.000	2.375	0.697	SNN; SA
		Energy BT	89.741	109.511	88.397	95.336	195.984	55.477	90.752	
	Scene 3	BT	1.715	0.625	1.750	17.250	22.000	1.250	0.886	SA; SS
		Energy BT	31.493	77.379	31.323	35.983	255.125	33.278	42.213	
	Overall	BT	2.795	2.000	2.625	24.250	4.900	2.250	0.571	SNN; MS
		Energy BT	63.130	164.660	64.485	217.412	443.696	55.883	59.967	
PM A - sc removed	Scene 1	BT	2.525	1.875	2.500	40.000	6.700	2.500	0.550	SNN; SA
		Energy BT	33.860	77.490	33.718	218.735	100.396	27.547	35.749	
	Scene 2	BT	2.255	1.125	2.125	31.250	5.800	2.250	0.697	SNN; SA
		Energy BT	57.066	63.183	31.250	66.369	86.454	27.803	60.697	
	Scene 3	BT	2.795	1.375	2.625	26.000	4.000	2.500	0.613	SNN; MS
		Energy BT	26.312	40.425	35.435	37.705	79.071	20.876	21..408	
	Overall	BT	3.065	1.875	2.500	34.750	4.900	2.500	0.571	SNN; MS
		Energy BT	35.102	65.911	38.613	151.900	115.260	25.669	33.566	

Table 4.2: PM B results. **Note:** BT - Best threshold

			SS ($D_{thd}[m]$)	DS ($C_o[m]$)	SA ($C_o[m]$)	SA ($\beta[^o]$)	ABD ($\lambda[^o]$)	SNN ($D_{thd}[m]$)	MS ($CosD_{thd}$)	Algorithms with best score
PM B - Ordinary scans	Scene 1	BT	2.795	2.125	2.750	34.750	4.900	2.500	0.634	SNN; SS
		Energy BT	89.941	98.245	89.982	112.718	118.049	89.167	93.745	
	Scene 2	BT	2.660	0.875	2.625	31.25	4.000	2.250	0.718	SNN; SA
		Energy BT	81.635	102.677	81.551	92.024	114.799	69.426	90.008	
	Scene 3	BT	1.985	0.750	1.750	15.500	22.000	1.875	0.718	SNN; SA
		Energy BT	43.906	60.550	42.839	44.750	181.235	39.924	46.159	
	Overall	BT	2.660	1.375	2.625	29.500	14.800	2.250	0.697	SNN; SS
		Energy BT	71.269	88.451	71.782	87.086	172.578	68.890	74.961	
PM B - sc removed	Scene 1	BT	2.930	2.125	2.875	40.000	4.900	2.625	0.550	SA; SS
		Energy BT	93.748	112.961	93.408	133.794	108.479	94.181	96.882	
	Scene 2	BT	3.200	2.750	2.875	31.250	4.000	2.250	0.718	SNN; SS
		Energy BT	87.907	93.714	88.115	106.663	106.267	71.539	103.816	
	Scene 3	BT	1.985	1.000	1.875	6.750	4.000	2.000	0.718	SNN; SS
		Energy BT	44.405	59.054	51.463	54.934	74.094	38.459	48.220	
	Overall	BT	2.930	1.875	2.875	27.750	4.000	2.500	0.655	SNN; SS
		Energy BT	73.849	92.725	76.959	105.845	97.212	70.492	81.721	

Table 4.3: Algorithm's processing time of all the 867 scans; the time value is given in seconds.

	SS	DS	SA	ABD	SNN	MS
Scene 1	1.739	1.569	1.489	1.457	1.523	2.953
Scene 2	0.237	0.208	0.181	0.179	0.400	0.355
Scene 3	0.428	0.392	0.375	0.383	0.708	0.563
Overall	2.426	2.193	2.070	2.045	2.692	3.902

In order to be aware of the computational weight of each algorithm, the processing time of the 867 scans was measured; the results are presented in table 4.3 were the values are given in seconds. These time values are purely indicative, because in theory simple segmentation should have given a lower processing time than the Santos Approach or even ABD as they have more complex threshold conditions; however, as expected, Multivariable Segmentation is by far, the algorithm with higher processing time as for each pair of points is calculate the array of attributes which demands some computation power.

4.3 Discussion

In this section, the results will be discussed in detail. There will be given more relevance to the results for the scans with all the valid points as they represent the real world scenario; the results for the scans where the points that form small segments in the Ground-truth are merely indicative.

Simple Segmentation commentary

Performance measure A

- Looking at the overall graphic, for lower threshold values the resulting energy reaches high values as the algorithm is over-segmenting: ratio $[n\_S_{GT}/n\_S_{Alg}]$ has low values. For a $D_{thd} \approx 1.500\ m$ the curve starts to stabilize reaching the "optimal value" for $D_{thd} \approx 2.500\ m$.
- As expected, the energy values are higher in the scene 1, where the GT segments possess more range points, so the distance between the segments' central points is higher, the energy value is exacerbated by the fact that the energy value is multiplied by the quotient between the number of points of the GT segment and the ones in the segment given by the algorithm.
- In the scene 3, the energy highly increases for a $D_{thd} \approx 1.750\ m$.

Performance measure B

- The shapes of the curves for this PM are similar to the ones given by the PM A: High energy values for $D_{thd} < 1.500\ m$ and then a stabilization of the energy values; scene 1 with the most influence for the Overall result.

- Optimal value should be a $D_{thd} \approx 2.500\ m$.

Dietmayer Segmentation commentary

Performance measure A

- As in the previous algorithm, the overall result is highly influenced by the results of scene 1. In that scene, the energy has high values for small C_0 values, the curve begins to stabilize when $C_0 \approx 1.000\ m$ until $C_0 \approx 2.000\ m$ where the ratio $[n\_S_{GT}/n\_S_{Alg} \approx 1.000]$; then the resulting energy values highly increase due to under-segmentation situations.
- In scene 2, the energy has the minimum value for $C_0 \approx 1.000\ m$; for higher C_0 values the energy will increase.
- For scene 3, the "optimal value" is reached for $C_0 \approx 0.500\ m$; however when the small segments are removed, the ideal C_0 value will be between 1 and 1.500 m.

Performance measure B

- As in the previous PM, the overall energy curve begins to stabilize when $C_0 \approx 1.000m$ until $C_0 \approx 2.000\ m$; however, the minimum energy value is obtained with $C_0 \approx 1.500\ m$.
- For scenes 2 and 3, the ideal C_0 is lower than 1 m.
- Optimal value should be a $C_0 \in [1.000, 1.500]\ m$.

Santos Approach

Performance measure A

- When the free variable is the C_0 parameter, the Energy curve is similar to the Dietmayer curve; however, the optimal value for the overall path occur for $C_0 > 2.500\ m$ and around $C_0 = 2.000\ m$ for the Scenes 2 and 3.
- The average energy values for this PM are generally lower than the ones in the Dietmayer Segmentation method; which proves that the β angle can have a positive influence to the segmentation.
- For a $C_0 = 1.000\ m$ the overall result shows that the ideal β angle value is around 25^o. If the using C_0 parameter had a higher value, the β could have a lower value.

Performance measure B

- In this PM, the energy variation is not so pronounced as on the PM A; the ideal C_0 value should be around $2.500\ m$ when $\beta = 15^o$.
- The ideal β value occurs is $\beta \approx 30^o$ when $C_0 = 1.000\ m$.

Adaptive Breakpoint Detector

Performance measure A

- High energy values in general general ($Energy > 400$); the ratio $[n\_S_{GT}/n\_S_{Alg}]$ rapidly decreases with the raise of the value of the λ value;
- Overall, the optimal value for this method was $\lambda = 5^o$
- For scene 3 the energy seems to decrease with the raise of the λ angle; however, when the small segments are removed; the energy increases with the raise of the λ angle

Performance measure B

- For this PM, the best threshold is always on the extreme values: $lamda = 4^o$ for scenes 1 and 2 and $lamda = 22^o$ for the scene 3.
- When the small segments are removed, the best threshold seems to tend to zero.

Spatial Nearest Neighbour

Performance measure A

- Considering the overall results, the best D_{thd} values are set between the 1.5 and $2.5m$, as the values outside this range present a raise of the average energy value.
- In the scene 3, the minimum energy threshold was achieved for $D_{thd}1.25m$, while when the small segments are removed, the minimum value is for $D_{thd}2.500\ m$.

Performance measure B

- For this PM, all the curves are similar among themselves; with the optimal threshold value situated between 2.000 and 2.500 m.

Multivariable Segmentation

Performance measure A

- Overall, the energy value is increasing as the $CosDi$ raises, as the algorithm is being penalized by the over-segmentation. The raise is negligible until a cosine distance value of $CosDi \approx 0.700$; after this value the energy results escalate.
- For the scene 2, it can be observed that the minimum energy value occurs when $CosDi \approx 0.700$.
- On the scene 3, the ideal cosine distance value was $CosDi \approx 0.900$; however when the small segments are removed this optimal value is $CosDi \approx 0.625$.

Performance measure B

- For this PM, results show that the optimal cosine distance value is around $CosDi = 0.70$.
- Cosine distance values lower than 0.85 also present a similar result.
- The overall curve presents stable results until $CosDi \approx 0.850$ when the $[n\_S_{GT}/n\_S_{Alg}]$ ratio begins to highly decrease.

Note that the performance measures' results for the Multivariable Segmentation algorithm are in some way limited by the Euclidean distance threshold used to identify those spurious points as mentioned in the section 3.2.6.

4.3.1 Comparison

Observing tables 4.1 and 4.2, it is possible to infer the following conclusions:

1. The Spatial Nearest Neighbour was the most consistent method in all scenes as expected because it is the one that can partially overcome occlusion problems as discussed in section 3.2.5; a threshold distance $D_{thd} = 2.250\ m$ was the value that according to both performance measures can be considered the best.
2. The Santos Approach also revealed a good performance in all scenes; the results for this algorithm could be a little bit better for the right combination between C_0 and β: According to the results a consistent C_0 value would be $C_0 \approx 2.500$ with a $\beta \approx 30^o$.
3. Simple Segmentation method often appears as the second best algorithm, particularly in PM B; a simple segmentation with a $D_{thd} \approx 2.660\ m$ presents good results according to that PM.
4. The Multivaribale Segmentation also presents some interesting results often appearing as the third or second best method.
5. On the other hand, the Adaptive Breakpoint Detector was the one that presented the worst results among the tested segmentation methods, which means that this method can be more useful for indoor environment than complex scenarios like the road environment.

Chapter 5

Conclusions and Future Work

The final conclusions of this dissertation will be detailed in this chapter, stating the success or unsuccess of the initially proposed objectives.

5.1 Conclusions

One of the main concerns in mobile robotics, specially in autonomous driving, is the collision avoidance and path planning; in order to do that it is necessary to identify moving objects around the vehicle and to predict their trajectories, so Multi-target tracking algorithms are used. The first stage of a Multi-target tracking algorithm is the Segmentation stage which has the main purpose of defining the objects physical limits. It is an important process as errors in this stage are very difficult to detect and correct in advanced stages. Thus in this project, a study of several segmentation algorithms is made in order to infer which ones have the best behaviour and validity in road scenarios.

The main goal of this work, which was to perform a comparative and exhaustive evaluation of different segmentation algorithms in several road scenarios, was successfully achieved. The proposed algorithms have been successfully implemented; a Ground-truth was defined through an exhaustive hand-labelling process and were created several comparison performance measures that preform a quantitative evaluation of the algorithms performance.

The main issue when performing this type of evaluation has to do with the Ground-truth definition, as it is a very subjective topic. Even when using real-world images given by the car's cameras, it is still a difficult process to make the correspondence between a group of range points and its correspondent in the "real world". This problem becomes even more severe in challenging scenarios like the road environment than in a controlled indoor environment where data association task is facilitated. This subjectivity to define what a Ground-truth should really be, has a significant impact in the evaluation of the algorithms' validity as they are all compared with it.

Another issue during the development of the work was the creation of the comparison performance measures, many performance measures were tested, every one of them in some way penalizes incorrect segmentations and all algorithms have been evaluated equally; the differences between the performance measures were mainly in the way of how the bad segmentation occurrences are penalized.

As for methods with better results, the Spatial Nearest Neighbour was the one that revealed the most consistency in the several road scenarios as it can partially overcome occlusion problems which are one off the main problems when performing segmentation in road scenarios. The Santos approach of the Dietmayer Segmentation also presented interesting results. A combination between these two methods can result in interesting segmentation results: Combining the capacity of partially overcome occlusion of the Spatial Nearest Neighbour method and the adaptive threshold of the Santos Approach.

5.2 Future work

One of the main issues of Segmentation using LIDAR in road environment is the vegetation related problems. Future work will be the application of a K-means clustering to identify vegetation zones; the main idea is to keep only the smallest segments resulted from the application of a certain segmentation algorithm, and then use two or tree K segments, to group those remaining points; if some K cluster has few dispersion then we have a group of bushes or threes.

Another problem when working with LIDAR is that only range and angle positions of the surrounding objects is given, thus, for tracking purposes, it would be interesting to perform Segmentation combined with computer vision information given by the *AtlasCar's* cameras. The images can be useful to detect occluded objects and to object classification: vehicles and pedestrians.

5.3 Final Notes

All the files developed on the implementation of this project are well commented and can be found here:
http://lars.mec.ua.pt/lartk/doc/lidar_segmentation/html/index.html

Bibliography

[1] Atlas project (2013). Available from: http://atlas.web.ua.pt. Accessed on: May, 2013.

[2] Lidar (2013). Lidar Design: https://en.wikipedia.org/wiki/Lidar. Accessed on: May, 2013.

[3] MacLachlan, R. (2005). Tracking moving objects from a moving vehicle using a laser scanner. Pittsburgh: Carnegie Mellon University.

[4] Sick LMS100 Product Information, Technical data. (2008). *Laser Measurement System.*

[5] Talhada, T. (2012). Perceção 3D utilizando uma câmara estéreo. Master's thesis. University of Aveiro.

[6] Almeida, J. (2010). Target tracking using laser range finder with occlusion. Master's thesis. University of Aveiro.

[7] ROS (2013) Available from: http://www.ros.org/wiki. Accessed on May, 2013.

[8] Quigley, M., Berger, E., A. Y. N. (2007). Stair: Hardware and software architecture. AAAI 2007 Robotics Workshop.

[9] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibes, J., Berger, E., and Wheeler, R. (2009). ROS: *an open-source robot operating system. ICRA Workshop on Open Source Software.*

[10] Dietmayer, K.C.J., Sparbert, J. and Streller D. (2001). Model based Model based object classification and object tracking in traffic scenes from range images. In *Proceedings of IV IEEE intelligent vehicles sysmposium,* Tokyo, Japan.

[11] Santos, S., Faria, J.E., Soares, F., Araujo, R. and Nunes, U. (2003). Tracking of Multi-Obstacles with Laser Range Data for Autonomous Vehicles. In: *Proc. 3rd National Festival of Robotics Scientific Meeting (ROBOTICA)*, pp. 59-65, Lisbon, Portugal.

[12] Borges, G.A. and Aldon M.J. (2004). Line Extraction in 2D Range Images for Mobile Robotics. In: *Journal of Intelligent and Robotic Systems*, v. 40, n. 3, pp. 267-297.

[13] Premebida, C. (2012). Pedestrian Detection Using Laser and Vision. Phd thesis University of Coimbra.

[14] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In: *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1:281-297.

[15] Teknomo, Kardi (2007). K-Means Clustering Tutorial. University of Ateneo de Manila, Philippines.

[16] Kumar, V. (2002). Parallel Issues in Data Mining, VECPAR 2002

[17] Prembida, C. and Nunes, U. (2005). Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications. In *Proc. 5th National Festival of Robotics, Scientific Meeting (ROBOTICA)*, Coimbra, Portugal.

[18] Rabbania, T., Heuvel F. A., Vosselmanc G. (2004) Segmentation of Point Clouds Using Smoothness Constant, In: *SPRS Commission V Symposium 'Image Engineering and Vision Metrology*

[19] Google Earth. http://www.google.com/earth/index.html. Accessed on: May,2013.