Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática
2018

Nuno Miguel Soares Silva

Perceção Aumentada para o AtlasCar
Augmented Perception for AtlasCar

DOCUMENTO PROVISÓRIO

Nuno Miguel Soares Silva

Perceção Aumentada para o AtlasCar
Augmented Perception for AtlasCar

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática , realizada sob a orientação científica de Paulo Miguel de Jesus Dias, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, de Vitor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica e de Miguel Armando Riem de Oliveira, Professor Auxiliar do Departamento de Engenharia Mecânica.

DOCUMENTO PROVISÓRIO

o júri / the jury

presidente / president	**Prof. Doutor A** Professor Catedrático da Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)
vogais / examiners committee	**Prof. Doutor B** Professor Catedrático da Universidade de Aveiro (orientador)
	Prof. Doutor C Professor associado da Universidade J (co-orientador)

agradecimentos / acknowledgements

É com muito gosto que aproveito esta oportunidade para agradecer a todos os que me ajudaram durante este longos e penosos anos, cheios de altos e baixos (mais baixos que altos)...

Desejo também pedir desculpa a todos que tiveram de suportar o meu desinteresse pelas tarefas mundanas do dia-a-dia, ...

Palavras-chave

Navegação autónoma; ATLASCAR; Agrupamento de dados; Calibração; Etiquetação de dados; Deteção de Objetos; ADAS.

Resumo

Este trabalho baseia-se no desenvolvimento de um pacote ROS para o ATLASCAR 2. O ATLASCAR 2 consiste em um Mitsubishi i-MiEV usado exclusivamente para investigação de sistemas avançados de assistência à condução no Departamento de Engenharia Mecânica da Universidade de Aveiro. O ATLASCAR 2 está equipado com uma câmera PointGrey que foi usada para aquisição de imagens, dois scanners planares LIDAR e um laser de quatro feixes frontais. Este pacote ROS implementa nós para deteção semi-automática de objetos. Estes nós são utéis para etiquetar os dados ligados aos objetos detetados. Estes objetos, na prática, serão por exemplo veículos e peões na estrada. A manipulação dos dados na imagem foi realizada através das bibliotecas de OpenCV. Em paralelo, foi desenvolvido e melhorado a deteção da bola para o pacote de calibração multi-sensorial.

Keywords Autonomous driving; ATLASCAR; Data Clustering; ; Calibration; Data Labelling; Object Detection; ADAS.

Abstract This thesis is based on the development of a ROS package for the ATLASCAR 2. The ATLASCAR 2 consists in a Mitsubishi i-MiEV used exclusively for research of Advanced driver assistance systems (or ADAS) in the Mechanical Engineering Department at University of Aveiro. The ALTASCAR 2 is equipped with a PointGrey camera used to acquire images, two planar LIDAR scanners and a four-beam frontal scanner. This ROS package implements nodes for semi-automatic object detection. This nodes are useful to label data to the detected objects. This objects, in practice, are for example vehicles or pedestrians on the street. The handling of the image data was done through OpenCV libraries. In parallel, it was developed and improved the ball detection for the multi-sensorial calibration package.

Contents

List of Figures

List of Tables

Acronyms

AD	Autonomous Driving
ADAS	Advanced Driver Assistance Systems
ROS	Robot Operating System
GUI	Graphical User Interface
PCL	Point Cloud Library
OpenCV	Open Source Computer Vision Library
LIDAR	Light Detection And Ranging
LAR	Laboratory of Automation and Robotics
MTT	Multi Target Tracking
HSV	Hue, Saturation and Value
RCA	Radio Corporation of America
ATC	Advanced Technologies Center
SCOT	Shared Computer-Operated Transit
SMART	Singapore-MIT Alliance for Research and Technology
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
MPII	Max Planck Institut Informatik
ETHZ	Eidgenssische Technische Hochschule Zrich
EPFL	École polytechnique fédérale de Lausanne
ROI	Region of Interest

Chapter 1

Introduction

Technological studies in the fields of Advanced Driver Assistance Systems (ADAS) and Autonomous Driving (AD) have been substantially growing in the past decades in the automobile industry and in the academic environment.

An electric car platform equipped with sensors can be programmed to have the sensation of perception. It is important for AD vehicles to actuate accordingly to their environment. This thesis is focused on the research of a method to segment data into labels using a camera and Light Detection And Ranging (LIDAR) sensors in ATLASCAR 2, so that the vehicle can later create a model of the objects it will detect, track and label.

The objects detected by ATLASCAR 2 are to be used in deep-learning algorithms. The acquired data must be large enough so that the models for each object can be well defined.

This dissertation will focus on an improvement on the calibration of the camera installed in ALTASCAR 2. In addition to the calibration, a tool for image labeling and tracking of objects will be developed. This tool will be used to gather image templates while tracking objects in the camera video.

1.1 ATLAS Project

ATLAS is a project created by the Group of Automation and Robotics at the Department of Mechanical Engineering of the University of Aveiro, Portugal. The mission of the ATLAS project is to develop and enable the proliferation of advanced sensing and active systems designed for implementation in automobiles and affine platforms. Advanced active systems being improved, or newly developed, use data from vision, laser and other sensors. The ATLAS project has vast experience with autonomous navigation in controlled environments and is now evolving to deal with real road scenarios. To ensure that the developments are meeting the ATLAS project mission statement, a full sized prototype, the ATLASCAR 1, has been equipped with several state of the art sensors. [1] Currently, ATLASCAR 2 is the new full sized prototype being used for research. The ATLASCAR 2 is also equipped with LIDAR sensors and a camera.

1.1.1 History

The ATLAS Project was created in 2003 and began with robots developed to participate at AD competitions taking place at Portuguese National Robotics Festival. From this project,

three small-sized platform robots were built (figure 1.1). These robots were very successful having won prizes in some of the robotics competitions.

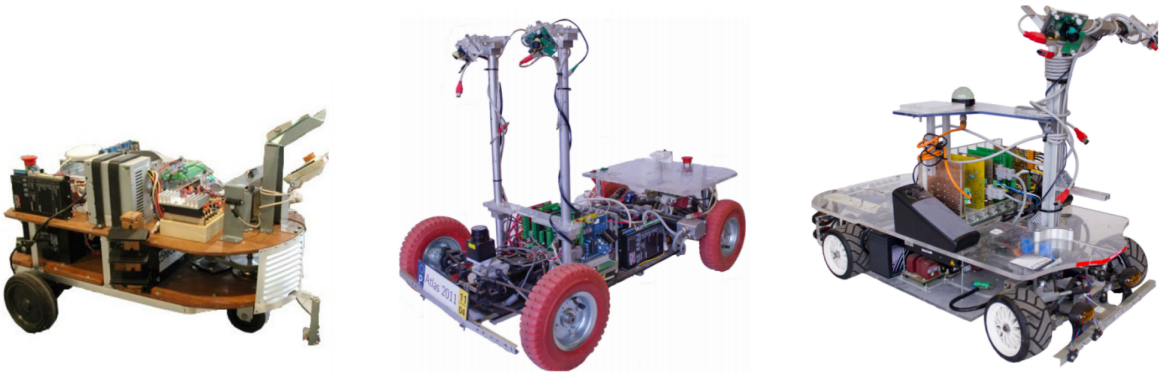


Figure 1.1: ATLAS project small-sized prototypes

As the project grew, it has evolved into full-sized prototypes: the ATLASCARs. ATLASCAR 1 (figure 1.2) is the first full-sized platform and it is based on a Ford Escort Station Wagon. The ATLASCAR 1 is well equipped with several LIDAR sensors and a camera. Data about its environment is gathered by the scanners which is then processed building perception into the car. With perception is possible for the vehicle to actuate allowing the car to move autonomously.

Figure 1.2: ATLASCAR 1 based on the Ford Escort platform

The ATLASCAR 1 brought successful results. In the end, the vehicle was able to move and execute maneuvers autonomously in small and controlled places. The ATLASCAR 1 was then replaced by a more recent vehicle. The ATLASCAR 2 (figure 1.2) is the new full-sized platform of the ATLAS project and it is based on a Mitsubishi i-MiEV. This is the vehicle

used for research in this dissertation. The ATLASCAR 2 is well equipped with various LIDAR sensors and a camera. It is also a full electric vehicle which will be easier to modify, test and control.

Figure 1.3: ATLASCAR 2 based on the Mitsubishi i-MiEV platform

1.1.2 Related work on ATLASCAR2

AD and ADAS are trendy topics at Laboratory of Automation and Robotics (LAR) in the University of Aveiro. Some research on the ATLASCARs related to AD has been made in the past. There are some relevant dissertations done at LAR that were useful for this project.

Multisensor Calibration and Data Fusion Using LIDAR and Vision

This is a master's thesis made by Silva [2]. The work presents an expansion to an existing extrinsic calibration package to vision-based sensors where a ball is used as calibration target.

The calibration consists in a appearance-based algorithm to detect the ball in the image and a range-based algorithm to detect the ball in the surroundings.

The calibration package consists in a graphical interface that allows the user to configure the various sensors to be calibrated. The estimated positions between sensors are achieved with sensor data fusion.

Visual and Depth Perception Unit for ATLASCAR2

This is a master's thesis made by Correia [3]. It is focused on the installation of an aluminum infrastructure on ATLASCAR 2 to support ranging and vision-based sensors. The sensors setup also include the electrical project in which a power distribution circuit was developed, consisting in the wiring installation and the communication infrastructure.

In addition, sensor calibration was done using the calibration graphical interface developed by Silva [2]. New sensors were added to the package so that the calibration could be proceeded. To demonstrate the functionalities of the platform setup, a multisensor data merging application was developed representing the free space to navigate around the car.

Active Tracking of Dynamic Multivariate Agents

This is a PhD Thesis made by Almeida [4]. The thesis is based in MTT in the fields of advanced safety systems. The focus lies in the prediction of the movement and actions of external agents. Two main targets are studied: vehicles and pedestrians.

This thesis proposes techniques to improve motion prediction to achieve the development of algorithms capable of target tracking. These algorithms make use of the 3D point clouds of the environment and vision-based sensors.

1.2 Motivation

The ATLASCAR 2 is the new platform of the ATLAS Project. The car has a mounted infrastructure for LIDAR scanners and a vision-based sensors. With a fully equipped vehicle, the data from the surroundings is ready to be received and processed.

1.3 Objectives

The objectives for this dissertation are, firstly, to improve the calibration of the camera, in particular regarding the detection of the ball by the camera in the existing multi-sensor calibration package developed using the Robot Operating System (ROS). Secondly, the development of other ROS package used for semi-automatic detection and labelling of objects in the field of view.

The calibration package was already developed by Silva[2]. The methods used to detect the ball in the camera image were basically filtering values from the Hue, Saturation and Value (HSV) color space. There are methods that can make this detection more robust that will be explained in this thesis.

To complete the goal of object tracking and image labelling there were a set of techniques and libraries used. In vision-based sensors, the object tracking algorithm is based on template matching methods and in range-based sensors the tracking is accomplished by the MTT library developed by Almeida [4].

1.4 Document Structure

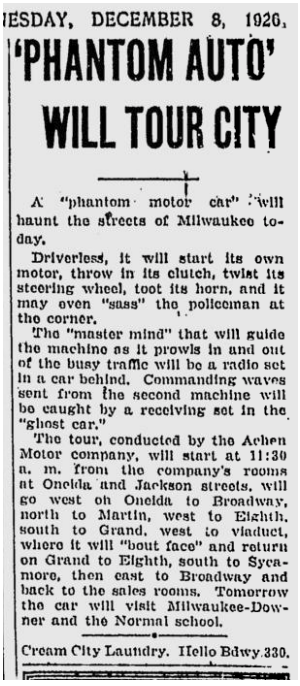
Escrever estrutura do documento quando este estiver completo

Chapter 2

State of the art

2.1 Milestones on the history of autonomous vehicles

Overall, motorized road transport led to the accidental deaths of around 200,000 US citizens in the 1920s; by far the greatest number of these were pedestrians. [5] The idea of substituting error-prone humans with technology thus practically suggested itself. The first registered experiments for AD have been conducted circa the 1920's [6] in Milwaukee. A 1926 Chandler was equipped with a transmitting antennae and was radio-controlled by a second car that followed it.



IESDAY, DECEMBER 8, 1926.

'PHANTOM AUTO'
WILL TOUR CITY

A "phantom motor car" will haunt the streets of Milwaukee today.

Driverless, it will start its own motor, throw in its clutch, twist its steering wheel, toot its horn, and it may even "sass" the policeman at the corner.

The "master mind" that will guide the machine as it prowls in and out of the busy traffic will be a radio set in a car behind. Commanding waves sent from the second machine will be caught by a receiving set in the "ghost car."

The tour, conducted by the Achen Motor company, will start at 11:30 a. m. from the company's rooms at Oneida and Jackson streets, will go west on Oneida to Broadway, north to Martin, west to Eighth, south to Grand, west to viaduct, where it will "bout face" and return on Grand to Eighth, south to Sycamore, then east to Broadway and back to the sales rooms. Tomorrow the car will visit Milwaukee-Downer and the Normal school.

Cream City Laundry. Hello Bdwy.330.

Figure 2.1: The Milwaukee Sentinel - 8 Dez 1926 - 'Phantom Auto' Will Tour City

In the 1950s promising trials in AD took place. General Motors conducted experiments in miniature models along with the electronic manufacturer Radio Corporation of America (RCA). The two companies later developed a full size system that was successfully demon-

strated completing a test route of one mile. [5]

In the 1980s, pioneer Ernst Dickmanns designed a vision-guided Mercedes Benz along with the Bundeswehr University Munich engineering team, achieving a speed of 63 km/h on streets with no traffic. In the late 80s, projects with both LIDAR scanners and computer vision were carried out. In 1989 the first experiments with vehicles making use of neural networks were conducted. [7]

Since then, many companies and research organizations have been developing various prototype cars. In the past decade, electric motored cars have emerged and new opportunities for AD and ADAS research have appeared.

Waymo, the Google self-driving car project, begun testing driverless cars without someone at the driver position. The Waymo project started in 2009 and it counts more than 5 million miles self-driven. Google has recently partnered with Jaguar and designed self-driving Jaguar I-PACEs (figure 2.2). Tests on the newest self-driving Waymo's vehicle will be conducted in 2018. [8]

Figure 2.2: Waymo's Jaguar I-PACE

Another example of an autonomous vehicle project is the Uber Advanced Technologies Center (ATC) car based on an hybrid Ford Fusion (figure 2.3). The vehicle is equipped with state of the art LIDAR scanners, and several vision-based sensors and radars.

Audi released its A8 (figure 2.4) and the company stated that they would be the first manufacturer to use laser scanners in addition to cameras and others sensors in autonomous vehicles. The vehicle was designed to a level 3 autonomous driving: it is capable of self-driving with the expectation that the human driver will respond appropriately to a request to intervene. The Audi AI traffic jam pilot takes over the driving task in slow-moving traffic up to 60 km/h. [9]

Like the University of Aveiro, many other universities and research institutes study the AD and ADAS paradigms.

Another interesting autonomous vehicle project is the Shared Computer-Operated Transit

Figure 2.3: Ford Fusion Uber ATC car

Figure 2.4: The new Audi A8

(SCOT) vehicle (figure 2.5), conducted by the Singapore-MIT Alliance for Research and Technology (SMART). [10] Like the ATLASCAR 2, SCOT is also a Mitsubishi i-MiEV used to research ADAS and AD at SMART and it is designed for operations on public roads. The SCOT vehicle also relies on LIDAR sensors similar to ATLASCAR 2. [11]

Figure 2.5: SCOT - Shared Computer-Operated Transit vehicle

2.2 Multi Sensor Calibration

For vehicles to become fully autonomous, it is needed for them to recognize their environment. The perception of their surroundings is obtained from data acquired through ranged-based sensors. The more sensors the vehicle has, the more accurate it can be about the environment. A car equipped with several sensors needs to know the position of each sensor so that the readings can be aligned and exact perception can be obtained.

For this dissertation, the camera calibration is to be improved. Multi sensor calibration in ATLASCAR 2 is done using a ball as a target. While moving the ball around the sensors, a point cloud is created for each sensor. These point clouds are aligned so that the estimate pose of each sensor can be obtained using an arbitrary sensor as reference. [2]

2.3 Image Labelling

Tracking of objects in image processing is done frequently using bounding boxes around the target. These boxes are often linked to a class in which the object in the template represents. Image labelling is the act of relating the object to this class, or more specifically, the label.

2.3.1 KITTI Dataset

Some image labelling datasets already exist. One of them and probably the most well-known in the fields of AD is the KITTI dataset. [12] The KITTI Dataset was captured from a Volkswagen station wagon (figure 2.6) for use in mobile robotics and AD research. The KITTI benchmark suite is born in 2012 at Karlsruhe Institute of Technology by the need to have a dataset to classify objects on the streets. This project has grown by increasingly adding more results with more sensors. The KITTI benchmark started with the stereo, flow and odometry benchmarks and today it includes standards for object tracking and more.

Figure 2.6: Volkswagen Station Wagon used in the KITTI Dataset

Just like ATLASCAR 2, the car used in the KITTI dataset is equipped with LIDAR sensors and Point Grey Video Cameras. The dataset is used for automatic recognition and tracking of vehicles and pedestrians. It consists in image sequences (see figure 2.7) and a text file in which, for each frame the various objects in the field of view are depicted with and identification number, a label, and coordinates about their position regarding the car and in the image. [13]

In listing 2.1 it can be analyzed an example snippet in which there are two lines of what the KITTI dataset looks like. Each line starts with the frame ID and the ID of the object being tracked. Then it is added a label to classify this object. There are also flags to indicate if the object is either truncated or occluded in the image sequence. The following numbers consist in the alpha (observation angle of object), the left, top, right and bottom of the 2D bounding box, the height, width and length of the 3D bounding box and its XYZ coordinates. The last number consists in the 3D rotation angle in the Y axis. [14] The legend of this KITTI dataset snipped would be the following:

```
1 ...
2 3 1 Cyclist 0 0 -1.931469 759.786603 146.098339 954.280160
   374.000000 1.739063 0.824591 1.785241 1.821119 1.569936 5.783265
    -1.642450
3 3 2 Pedestrian 0 0 -2.547728 1154.836779 148.360923 1241.000000
   321.627088 1.714062 0.767881 0.972283 6.463579 1.474131 7.560739
    -1.860031
4 4 -1 DontCare -1 -1 -10.000000 252.530000 168.660000 284.460000
   202.850000 -1000.000000 -1000.000000 -1000.000000 -10.000000
   -1.000000 -1.000000 -1.000000
5 4 0 Van 0 0 -1.808333 290.287584 146.641981 444.387179 269.473545
    2.000000 1.823255 4.433886 -4.934786 1.601945 14.098646
   -2.139796
6 4 1 Cyclist 0 0 -1.929519 767.158958 140.942948 961.992360
   374.000000 1.739063 0.824591 1.785241 1.881359 1.534695 5.785600
    -1.631447
7 4 2 Pedestrian 1 0 -2.557045 1180.675035 151.025283 1241.000000
   325.015204 1.714062 0.767881 0.972283 6.516488 1.497786 7.267796
    -1.846627
8 ...
```

Listing 2.1: KITTI dataset file snippet.

```
1 frame_id object_id label truncated occluded alpha left top right
   bottom height width length x y z rotation_y
```

Listing 2.2: KITTI dataset file snippet legend.

The legend in listing 2.2 is not included in the dataset files. Analyzing the snippet, it is possible to locate a a cyclist and a pedestrian in frame 3 and the same cyclist and pedestrian (because they have the same object_id) in the next frame with also a van. The `DontCare` label is often shown representing an object detected that is not related to the scope of the KITTI dataset. Other information indicate where these objects are found relatively to the car.

2.3.2 HumanEva II Dataset

The HumanEva II Dataset from the Max Planck Institut Informatik (MPII) was also a dataset worth researching. Although it is used for pedestrian detection only, it was important to have a comparator with the KITTI dataset, specially regarding its structure. This dataset appears in the demand for a way to represent information about detection and tracking of humans and their poses captured by a single image camera. The HumanEva dataset has information about the bounding boxes position used to track and detect pedestrian limb poses. This information is useful to know which direction the person is facing from the 3D skeleton derived from the pose. The data structure in the dataset is similar to a XML file.

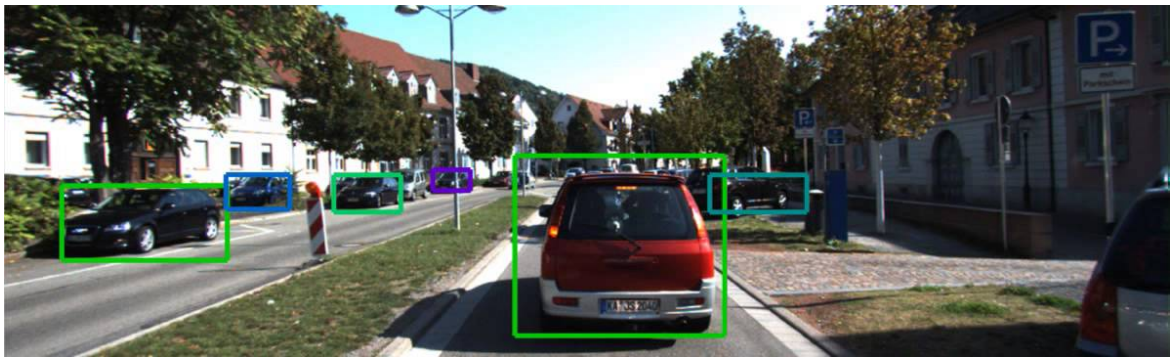


Figure 2.7: Sequence example of tracking by detection using the KITTI dataset

For each frame in the image sequences there are several bounding boxes with the respective coordinates. [15]

By looking at listing 2.3, in this dataset snippet is easy to identify the interest points in the given frame. The files are a set of annotations called *annotationList* in which a path to the image corresponding to the frame is given. For each image there is *annorect* with bounding box coordinates $(x1, y1, x2, y2)$, a score, silhouette, articulation and viewpoint id. There is also a section called *annopoints* where sets of points with an id are annotated.

2.3.3 Other relevant datasets

Other datasets included in the research for this dissertation are found in ETHZ and in EPFL projects.

ETHZ dataset

ETHZ conducted studies for detection and tracking of people on the street. The dataset is simple: for each frame there are several bounding boxes in the image.

This dataset is focused just in the detection and tracking of pedestrians in the image. [16] In listing 2.4 each line is composed with a string defining a path to the image representing the actual frame, followed by tuples of four elements $(x1, y1, x2, y2)$ representing the bounding boxes where pedestrians are found in the respective frame.

EPFL dataset

The EPFL designed a dataset for multiple people in a camera environment, independent of the scenario. This dataset used various synced video cameras filming the same area in different angles.

In listing 2.5 there is a snippet of the dataset. The dataset includes, for each frame, various objects identified with a number, a label, bounding box coordinates, and flags to point out if the person is occluded, lost, or if the detection was automatically interpolated from the other camera's information. [17] The particularity of the structure of this dataset is that, for each object, it is tracked in the image sequences individually, and only then another object is tracked and labeled. In listing 2.6 a legend of this dataset can be found.

```
<annotationlist>
   ...
<annotation>
      <image>
         <name>test/00050.png</name>
      </image>
      <annorect>
         <x1>65</x1>
         <y1>45</y1>
         <x2>118</x2>
         <y2>213</y2>
         <score>636</score>
         <silhouette>
            <id>2</id>
         </silhouette>
         <articulation>
            <id>-1</id>
         </articulation>
         <viewpoint>
            <id>-1</id>
         </viewpoint>
         <annopoints>
            <point>
               <id>0</id>
               <x>92</x>
               <y>114</y>
            </point>
            <point>
               <id>1</id>
               <x>108</x>
               <y>96</y>
            </point>
         </annopoints>
      </annorect>
      <imgnum>50</imgnum>
   </annotation>
   ...

</annotationlist>
```

Listing 2.3: HumanEva dataset file snippet.

Figure 2.8: Example image of the HumanEva dataset that generated the snippet in listing 2.3

```
...
"left/image_00000015_0.png": (222, 177, 268, 312), (373, 105,
  463, 393), (458, 220, 487, 285), (310, 225, 327, 265), (335,
  228, 352, 264), (267, 228, 281, 261);
"left/image_00000016_0.png": (220, 172, 266, 313), (378, 407,
  476, 102), (462, 219, 486, 285), (312, 223, 327, 264), (337,
  226, 352, 262), (267, 231, 279, 260);
"left/image_00000017_0.png": (219, 173, 267, 316), (394, 94, 489,
   423), (313, 222, 330, 262), (338, 227, 354, 262), (267, 228,
  279, 260);
...
```

Listing 2.4: ETHZ dataset dataset file snippet.

Figure 2.9: One of the images of the ETHZ dataset that generated part of the snippet in listing 2.4

```
1  ...
2  1 80 45 99 98 9363 0 0 1 "PERSON"
3  1 80 45 99 98 9364 0 0 0 "PERSON"
4  1 77 45 96 98 9365 0 0 1 "PERSON"
5  1 74 45 93 98 9366 0 0 1 "PERSON"
6  1 71 46 90 99 9367 0 0 0 "PERSON"
7  2 81 45 110 126 0 0 0 0 "PERSON"
8  2 80 45 109 126 1 0 0 1 "PERSON"
9  2 80 45 109 126 2 0 0 1 "PERSON"
10 2 80 45 109 126 3 0 0 1 "PERSON"
11 2 80 45 109 126 4 0 0 1 "PERSON"
12 ...
```

Listing 2.5: ETHZ dataset file snippet.

Figure 2.10: One of the images of the EPFL dataset that generated part of the snippet in listing 2.5

```
1  track_id. All rows with the same ID belong to the same path.
2  xmin. The top left x-coordinate of the bounding box.
3  ymin. The top left y-coordinate of the bounding box.
4  xmax. The bottom right x-coordinate of the bounding box.
5  ymax. The bottom right y-coordinate of the bounding box.
6  frame_number. The frame that this annotation represents.
7  lost. If 1, the annotation is outside of the view screen.
8  occluded. If 1, the annotation is occluded.
9  generated. If 1, the annotation was automatically interpolated.
10 label. (human, car/vehicle, bicycle...)
```

Listing 2.6: EPFL dataset legend.

2.4 Object Detection and Tracking

Computer vision and image processing are often related to object detection. To detect a certain object it is common to look at their geometry and to their color. One of the uses of object detection is tracking its movement. In a still camera it is common to use methods like optical flow and background removal. In the fields of ADAS and AD, it is assumed that the camera is moving since it belongs to a vehicle, and recently, many car manufacturers already offer automatic pedestrian detection in their latest vehicles.

In this work, it will be developed for the the ATLASCAR 2 a semi-automatic system of detection and tracking of an object pointed by a human. The selected area in the image will be the given object and it will be tracked using template matching in the image, with

increased accuracy using object detecting with the LIDAR sensors.

Object detection using ranged-based sensors is often accomplished through data clustering. The ATLASCAR 2 is equipped with LIDAR scanners that send data in a message format. When a new scan is received, it is broken into small groups of points. After obtaining the current set of measurements from the most recent scan, these sets of points are associated with objects from past iterations. This association is based on the distance from the current object position and its predicted position. Therefore, it is possible to detect and track objects automatically. The MTT library developed by Almeida [18] was used for this project to fulfill the range-based object detection.

2.5 Contribution

Since the scope of this dissertation is meant for general objects detection in the street while expecting an human input, it will not be used a database or any set of image templates. The detection and tracking will be done semi-automatically. This work, in the end, results in a tool for machine learning. By gathering the data and creating a model for the object based on the previous frames, it's possible to merge the training and inference phases into one, creating datasets and image sequences that can be of use in a convolutional neural network in deep learning fields.

To accomplish this, a multi-modal approach was utilized, combining data retrieved from several ranged based and visual sensors. The clustered data is meant to detect and track objects in the sensor's field of view. The information from the laser points are treated with a range based detector algorithm and the image sequences are processed with an appearance based algorithm. [19]

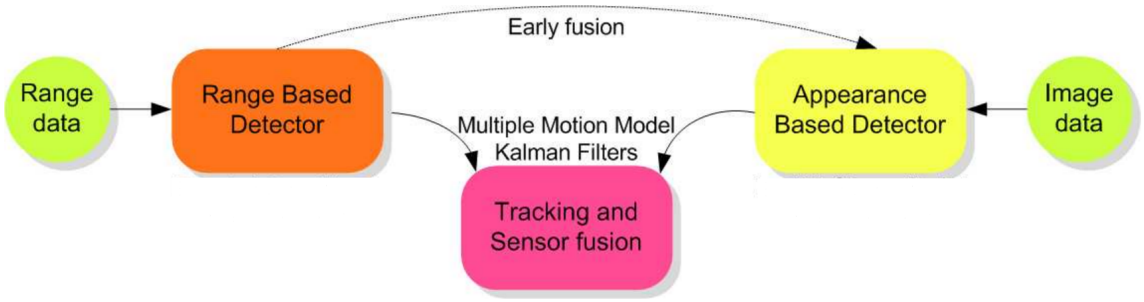


Figure 2.11: Overview of the multi-modal approach.

Utilizing a multiple motion model and Kalman filters it is possible to consolidate both range and image data. This process is called sensor fusion. By incorporating ranged based information in image data it is possible to have augmented perception of the surroundings. With the ability to have basic cognitive process of the environment, it is possible to detect and track objects in motion.

Chapter 3

Experimental Infrastructure

In this section, the hardware and frameworks used for this thesis will be described. The hardware used was mainly the ATLASCAR 2 and its sensors: two SICK LMS151 LIDAR, one SICK LD-MRS LIDAR and a PointGrey Zebra 2 Camera. The central framework that receives messages from the sensors is ROS. This data will be processed using the Open Source Computer Vision Library (OpenCV) for images, and several libraries will be used for the range-based sensors, such as the Point Cloud Library (PCL) and MTT.

3.1 ATLASCAR 2

The ATLASCAR 2 is based in the platform of the 2015 Mitsubishi i-MiEV, a full electric vehicle. The battery that powers the engine is the same powering the camera and the sensors. The main characteristics of the car are in table 3.1 [20].

Figure 3.1: The Mitsubishi i-MiEV

Table 3.1: Mitsubishi i-MiEV technical specifications

Characteristic	Unit	Value
Wheelbase	mm	2550
Track (Front/Rear)	mm	1310/1270
Vehicle weight	kg	1450
Engine	–	Electric
Electric energy consumption	Wh/km	135
Electric range (NEDC)	km	150
Maximum speed	km/h	130
Minimum turning radius	m	4.5
Max. Power output	kW	49
Max. torque	Nm	180
Traction battery type	–	Lithium-ion battery
Traction battery voltage	V	330
Traction battery energy	kWh	16
Regular charging (AC 230V 1 phase) 8A	hrs	10

3.2 Sensors

The sensors equipped in the ATLASCAR 2 are two SICK LMS151 LIDAR, a SICK LD-MRS LIDAR and a PointGrey Zebra 2 Camera. It is of most importance for the ATLASCAR 2 to have this sensors to be capable of perception. The sensors have been mounted in the front of the car in an aluminum infrastructure designed by Correia. [3] These devices are connected to a network switch installed in the car to which a computed can be plugged to receive the data from the sensors.

3.2.1 SICK LMS151 LIDAR

The SICK LMS151 (figure 3.2) is a LIDAR sensor designed to be used in outdoors. It is a planar infrared scanner with a large planar aperture angle often used in robotics and in AD fields for its high scanning frequency and operating range. This scanner is also able to scan distances through fog, glass and dust (multi-echo technology). This scanner is provided with an Ethernet TCP/IP interface with high data transmission rate. [21]

Table 3.2: SICK LMS151 specifications

Field of application	Outdoors
Laser Class	1 (IEC 60825-1:2014, EN 60825-1:2014)
Aperture Angle	270°
Scanning frequency	25 Hz / 50 Hz
Angular resolution	0.25° / 0.5°
Operating range	0.5 m ... 50 m
Max. range with 10 % reflectivity	18 m
Amount of evaluated echoes	2
Data transmission rate	10/100 MBit/s

For this project, the SICK LMS151 will operate at 50 Hz with an angle increment of

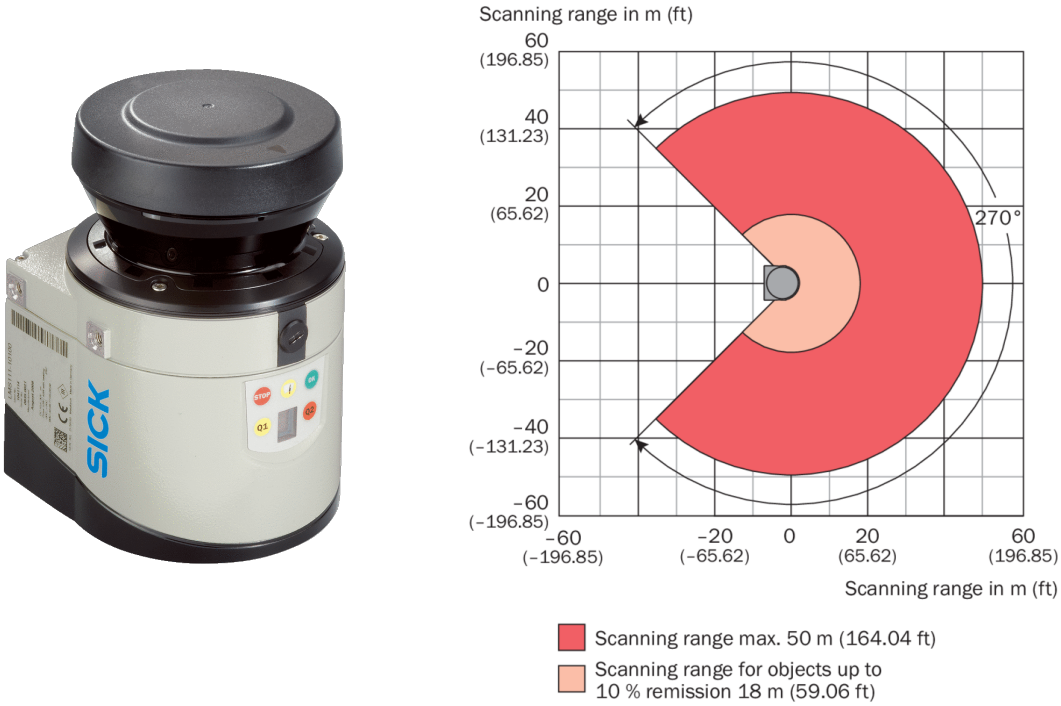


Figure 3.2: The SICK LMS151 LIDAR and its operating range

0.5° between readings. Each message will transmit a total of 540 points per scan in polar coordinates (r, θ). [21]

3.2.2 SICK LD-MRS LIDAR

The SICK LD-MRS (figure 3.3) is a LIDAR sensor also designed to be used in outdoors. It features 4 planar infrared scanners with 0.8° vertical aperture angle between each plan, offering tri-dimensional point clouds. It provides high scanning frequencies and long operating range up to 300 meters. This scanner is also provided with an Ethernet TCP/IP interface with high data transmission rate. [22]

Table 3.3: SICK LMS151 specifications

Field of application	Outdoors
Laser Class	1 (IEC 60825-1:2014, EN 60825-1:2014)
Scanner Planes	4 measuring planes
Aperture Angle	85°
Total Aperture	110°
Scanning frequency	12.5 Hz / 50 Hz
Angular resolution	0.125° / 0.25° / 0.5°
Operating range	0.5 m ... 300 m
Max. range with 10 % reflectivity	50 m
Amount of evaluated echoes	3
Data transmission rate	100 MBit/s

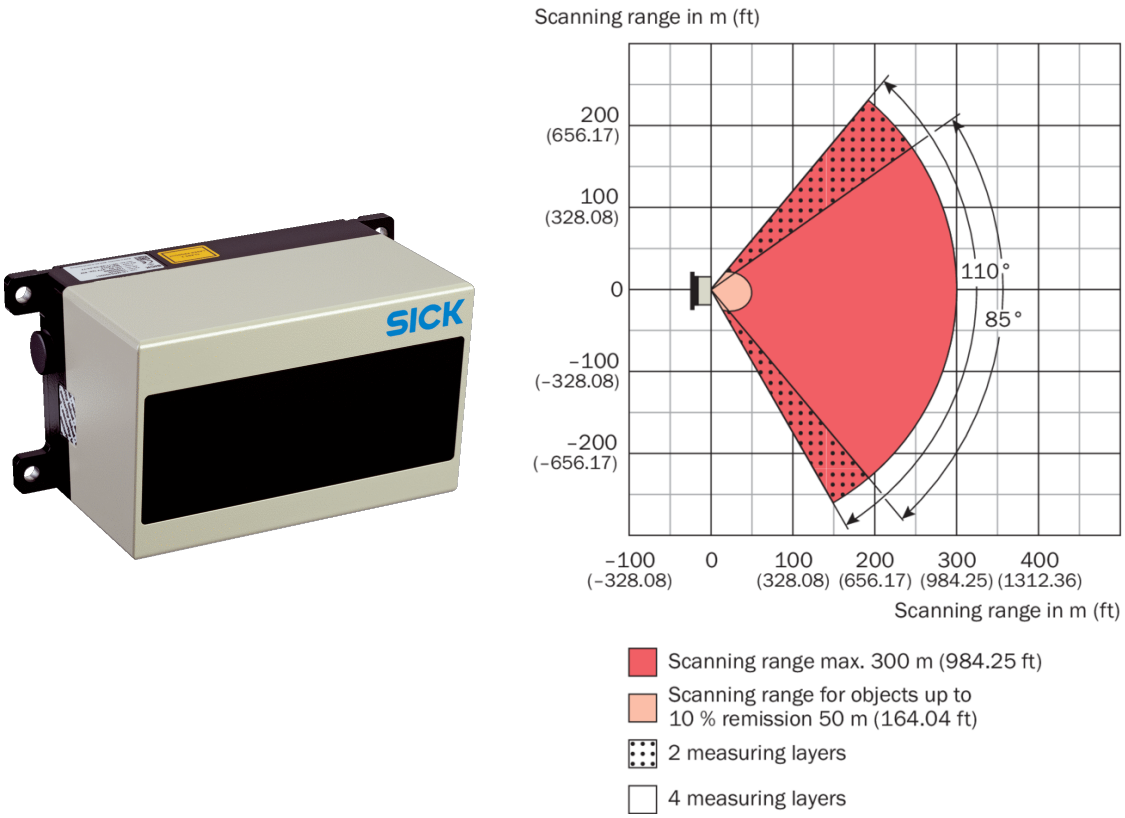


Figure 3.3: The SICK LD-MRS LIDAR and its operating range

For this project, the SICK LD-MRS will operate at 50 Hz with an angle increment of 0.5° between readings. Each message will transmit a total of 200 points per scan in polar coordinates (r, θ) for each plan. Since this scanner offers four planar scans, the final point cloud will total 800 points. [22]

3.2.3 PointGrey Zebra 2 Camera

The PointGrey Zebra 2 Camera (figure 3.4) is a high resolution camera with a Sony ICX274. It also features a GigE PoE Interface and it is highly configurable to fulfill any particular utilization needs. [23] The camera is inserted in a case made with 3D printing and designed by Correira [3]. Other relevant specifications are found in table 3.4.

Figure 3.4: The PointGrey Zebra 2 Camera

Table 3.4: PointGrey Zebra 2 Camera specifications

Resolution	1624 x 1224
Max. Frame Rate	25 FPS with HD-SDI
Megapixels	2.0 MP
Chroma	Color
Sensor	Sony ICX274 CCD
Image Buffer	32 MB
Interface	GigE PoE, HD-SDI

Working at maximum resolution and frame rate would saturate the network bandwidth and increase image processing times. In this project the camera's frame rate is set at 7.5 FPS so that a balance between image quality and processing optimization can be accomplished.

3.3 Software

In this section it will be discussed the software used in this dissertation. The base architecture of this dissertation is centered in the ROS framework. Many ROS tools and features are used such as Rviz, rosbags, roslaunch and rosrun. Two main nodes are to be developed in this work: the camera calibration package node (previously developed by Silva [2]) and

the labelling node. The handling of data from the sensors was done using PCL, MTT, and the image processing was performed with OpenCV libs. All software was programmed with C++.

3.4 ROS

The central framework in the ATLASCAR 2 is based on ROS Kinect on Ubuntu 16.04.

The Robot Operative System, although not an operating system, operates as a robotics middleware. ROS offers open-source services designed for developers that need hardware abstraction and low-level device control. Architectures in ROS are centered in applications called rosnodes which communicate with each other creating a graph of message-passing processes.

With ROS it is possible to receive data packets and transform them in messages that contain data from the sensors. It is possible to manipulate this data using ROS nodes and other tools.

3.4.1 Rviz

Rviz is the standard ROS tool for 3D visualization.

The Rviz is one of the most important tools as it will be used to visualize data from the ATLASCAR 2 either directly in real-time by connecting a computer to the car or in rosbags. It will also serve as a debugging tool in which pointcloud values can be analyzed. [24]

3.4.2 Rosbag

A rosbag is a file in ROS which contains messages saved from past events. While connected directly to a device, or several devices, multiple topics can be subscribed at once to be recorded into a rosbag.

The advantage of rosbags in this dissertation is to simulate the working environment of the ATLASCAR 2 without actually being in it. Several rosbags were recorded throughout the process of development in this project, either of calibration or for detection, tracking and labeling purposes.

In ROS, the rosbag package contains a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoid deserialization and reserialization of the messages. It also features command line tools for working with bags as well as code APIs to read/write and manipulate bags. [25]

3.4.3 Roslaunch

The roslaunch files are used as a tool for easily launching multiple ROS nodes. A roslaunch file sets up a roscore (os ROS master), sets parameters on the Parameter Server, and can also execute other roslaunch files.

It includes options to automatically re-spawn processes that have already died. The roslaunch takes in one or more XML configuration files (with the .launch extension) that specify the parameters to set and nodes to launch.

For example, a rosbag can be given as parameter to the roslaunch which opens the Rviz with a previous defined configuration to visualize the data from that rosbag.

In this project, roslaunch files are optimal to set up calibration values before launching the Rviz and the nodes that process the data from the LIDARs and the camera. The transformations between device frames are set up using roslaunch files and static transform publishers implemented by ROS.

The roslaunch files are also advantageous to set up the multiple drivers needed to bring up the several devices equipped in the ATLASCAR 2. Since the sensors are connected to a network switch, the parameters given in the driver's roslaunch file are the IP addresses of each of the devices. The drivers will then receive packets from the sensors and remap them into the ROS format.

3.4.4 Multi-sensor calibration package

The multi-sensor calibration package is a package developed by Silva [2]. The package contains a Graphical User Interface (GUI) used to calibrate the several sensors in the ATLASCAR 2. The package was initially developed for the ATLASCAR 1, and it was further used into the ATLASCAR 2 since the sensors were almost the same.

The calibration package features a graphical user interface in which several sensors can be selected to be calibrated. The available devices are the SICK LMS151 and SICK LD-MRS mounted in the ATLASCAR 2, PointGrey cameras, Microsoft Kinects, the SwissRanger SR4000 and the Velodyne VLP16 used in the ATLASCAR 1.

Chapter 4

Camera Calibration in ATLASCAR 2

One of the main tasks for this work was to develop and improve the ball detection for the calibration package used previously by Silva [2] and Correia [3]. In previous projects made with the ALTASCAR 2, the ball detection was done by only filtering the HSV values which was not enough to get optimal results.

4.1 Implementation

The ball detection upgrade begins by retrieving the video stream frames from the Point-Grey camera. The image obtained was worked in a rosbag used for testing in order for the development to be made outside the ATLASCAR 2. The ball used in the tests is the ball in figure 4.1.

Figure 4.1: Ball used for calibration testing

To acquire the images, the node creates a subscriber to get messages from the camera topic. ROS subscribers take the message in the topic and send it to a callback function in order to it to be processed. After obtaining information from the camera's message, it is needed to convert the RGB values into the HSV color space so that the image can be easily manipulated.

4.1.1 Background Subtraction

The first thing to do with the frames is to apply background subtraction. When the capture starts, the first received frame will be default background. To update the background to the actual frame in the stream the user presses the SPACEBAR (see figure 4.2). The background removal is a process used in many vision based applications with static cameras, in which a frame is captured and defined as the background of a sequence of images. Therefore, when an objects enters the scene, by subtracting the background with the actual frame it is possible to detect easily where the object is. [26]

Technically, background subtraction extracts the foreground from the static background. Most times, the background is worked out and a Gaussian blur is often applied to it. By defining a threshold, when subtracting the foreground with the background it is obtained a value for each pixel. It this value is higher than the threshold then (probably) an object entered that area.

Figure 4.2: Background in test rosbag used for testing

The background removal process in the calibration package for the ball detection is done with applying blur to the background frame and then subtracting the background with the actual frame. This will result in a frame with very low values in the pixels where no new objects are shown. In most cases the values will not get to zero because of minor changes to the background which need to be ignored. A value is then used to apply a threshold in the resulting subtracted frame. If a pixel presents a value under the threshold, then it is set to zero (black), otherwise it is set to one (white). In the end, it is obtained a binary image containing some noise due to shadows or other lightning changes, and potentially an area with more concentrated white values where new objects appear.

Figure 4.3: Frame with ball rolling in front of the camera

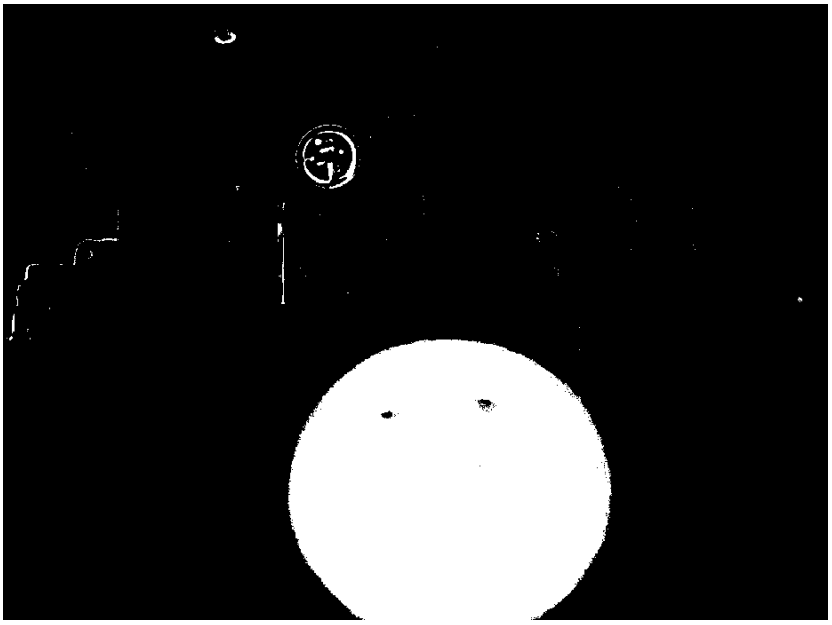


Figure 4.4: Background subtraction result with noise

4.1.2 Noise Filtering

Assuming that the ball is rolling in the ground in front of the camera and that the car is leveled to the ground, it is possible to ignore the upper half of the image. All pixel in the upper half are then neglected and set to zero (black). Removing the upper half will then remove a major part of noise that can be created in the background.

To remove the rest of the noise in the lower half, the neighbor pixels are counted. The image is processed left-to-right, top-to-bottom, so the matrix corresponding to the frame is iterated throughout its lines. A counter is added to check if the previous pixels contained white pixels (pixels with value of 1).

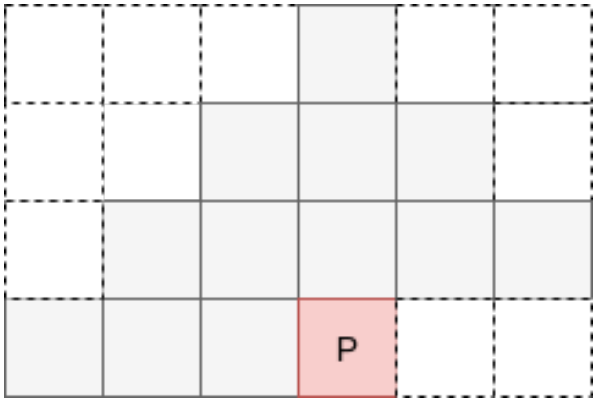


Figure 4.5: Neighbor pixels used in the algorithm

The pixel in red with letter P in figure 4.5 represents the pixel being analyzed at the moment. If the pixel is black (zero) then it is not noise and nothing needs to be removed, but if the pixel is white (one) it is needed to look at the neighbor pixels. Around pixel P there are pixels with gray color. These are the pixels to be analyzed in the algorithm. The strategy is simple: if all of these pixels are set to one (white) the pixel P probably represents an object in the frame, but if one of the pixels is set to zero (black) then the pixel P is most likely to be a noise pixel and it is also set to zero.

This process is similar to what is called an erosion. Erosion is a morphological operations that consists in a set of operations that process images based on shapes applying a structuring element to an input image and generate an output image. In erosion, the algorithm computes a local minimum over the area of a kernel. In figure 4.5 the kernel is composed by the gray pixels. As the kernel is scanned over the image, the algorithm computes the minimal pixel value overlapped by the kernel and replaces the image pixel under the anchor point with that minimal value. [27]

4.1.3 Color Filtering

After the erosion is done, the ball in the image is beginning to be extracted. The next step is to separate the ball using color filtering. To set the ball color the user needs to click in the ball to capture its hue. A mouse event is triggered and an handler function is called. The handler localizes the mouse click and retrieves the RGB values of the pixel in those coordinates. These values are converted to a hue value of the HSV color space. The selected hue is the color used to filter the ball from the rest of the image. By applying certain

thresholds it is possible to define an interval of color. The image will then be filtered by an interval centered in that hue value and the ball is now filtered from the rest of the image.

A bounding box is drawn around the ball. To draw this box it is needed to calculate the ball center point and box upper-left point. The bounding box will only be drawn in the binary image if there is an area of white pixels big enough. To do this, the image is iterated like in this erosion phase, left-to-right and top-to-bottom. The algorithm saves the coordinates of all white pixels. An average x and y are calculated, finding the ball center. The size of the bounding box is given by how much white is in an specific area. Supposing there is a possibility of a white pixel to appear as noise, that pixel will have low weight in the algorithm considering that the pixels around it are black. Therefore, white pixels gathered in a larger area have high weight. Hence, the width and height of the bounding box are given by the maximum distance between the pixels with higher weight.

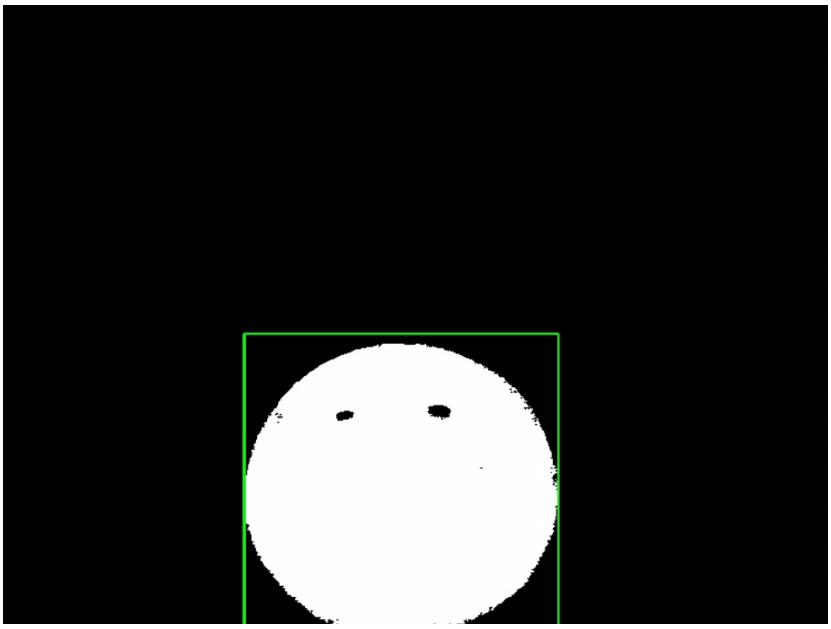


Figure 4.6: Ball detected with bounding box

4.1.4 Modifying the calibration package

The calibration package used to calibrate the several sensors in ALTASCAR 2 uses multiple nodes, one for each kind of sensor. The camera node in the package is called `point_grey_camera` and it uses the source file `point_grey_camera.cpp`. This file was modified in order to add the implemented features into the calibration package.

The `point_grey_camera` node receives the camera image by subscribing to the camera topic with ROS. The camera sends images in a ROS message format to be interpreted and processed by a callback function. The image processing algorithm follows the steps explained before in this section. After performing the ball detection, the center of the bounding box matching the center of the ball is retrieved. The ball center is published into the calibration package main node `calibration_gui`.

To publish the ball center, a Canny image is retrieved from the binary image. The Canny algorithm is often used in visual computation applications to detect the edges of an object with

low error rate, good localization and minimal response. OpenCV implements this function optimally by filtering any noise in the image using Gaussian filters. [28] Then it finds the intensity gradient of the image with a procedure analogous to Sobel. The Sobel derivatives uses discrete differentiation operators that compute an approximation of the gradient of an image intensity function combining Gaussian smoothing and differentiation. [29]

With this, all pixel that are not considered to be part of an edge are removed and only thin lines (the edges) will remain. An upper and lower threshold is applied. This is called the hysteresis procedure and it consists in removing the pixels with a gradient below to the lower threshold. If the pixel has a gradient higher than the upper threshold then it is accepted as part of an edge.

The next step is to find the contours in the image edges. OpenCV implements a function for structural analysis and shape descriptors called `findContours`. The method retrieves contours from the binary image using the algorithm of Suzuki, S. and Abe, K. [30]. The algorithm performs a point-in-contour test determining whether the point is inside a contour, outside, or lies on an edge.

When the contours are found, it is needed to find to centroid of the ball. The radius of the ball is calculated with the size of the given bounding box. The real radius of the ball is passed as an argument so it is possible to calculate the distance from the camera to it. The centroids are obtained with a method based on the circle radius and they are published as `PointStamped` ROS points to be presented in the calibration GUI.

4.2 Results

In figure 4.7 the multisensor calibration GUI is shown with a test result. The SICK LMS151 LIDARs were used as point of reference for the purpose of demonstration and to have a comparator point to the camera.

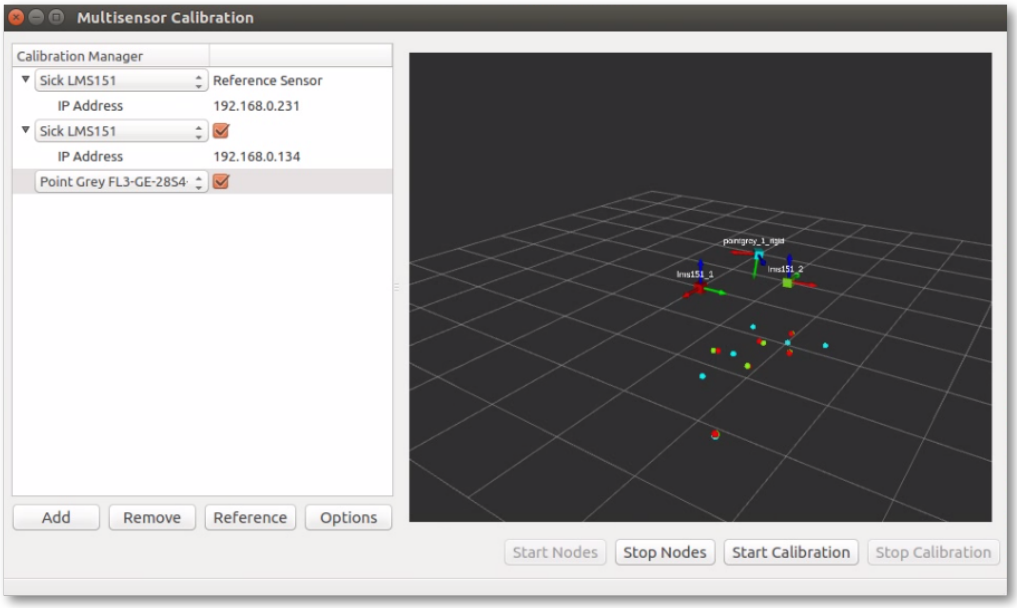


Figure 4.7: Calibration GUI with calibration result

```
1  −0.0563334   −0.998402  0.00440481   −1.07636
2   0.997797  −0.0561436   0.0353294    1.08224
3  −0.0350256  0.00638533   0.999366  0.0617893
4           0           0          0          1
```

Listing 4.1: Calibration output file.

During the calibration procedure, the ball rolls in front of the camera and sensors. The rage based sensors and the camera obtain a pointcloud of centroids of the ball during this activity. In the end, the transforms for each sensor are calculated, aligning the pointclouds of the several devices with each other.

In listing 4.1 an example output file of the calibration can be observed. The file contains a 4x4 matrix that indicates the transforms for the given sensor. Each sensor in the calibration will output its own file containing the transformation matrix. The reference sensor does not create a transform because it is assumed that this sensor is in the origin, unrotated.

Chapter 5

Object Detection, Tracking and Labelling

The main task of this dissertation is focused on the detection, tracking and labeling of objects in motion found in the field of view of the several sensors equipped in ATLASCAR 2. In this chapter it will be succinctly explained how these features were implemented.

Firstly, it will be described how the detection and tracking in the image is performed. In the image tracking phase happens also the labelling phase and the creation of dataset files and image templates.

Secondly, the implementation of the tracking of multiple targets with ranged based sensors will be explained. To make this step possible, the MTT library designed by Almeida [4] was used. The MTT library contains methods of perception and planar object detection.

Lastly, the multi-modal approach will be used where both data from the images and the LIDARs will be assembled and a single perception unit will be created. With this conceptualization, it is possible to detect, track and label objects easily in the ATLASCAR 2.

The image sequences and laser scan data obtained for the development of this stage of the dissertation were recorded into rosbags using the ATLASCAR 2 using the sensors it has. The rosbags were recorded in October 17, 2017, in the afternoon. Two rosbags were recorded:

- The first rosbag was recorded while leaving Departamento de Engenharia Mecânica at Universidade de Aveiro. The car travelled around the campus and visited the Alboi neighbourhood.

  In this bag there are cars, vans, cyclist and pedestrians. It is a bag where the car also runs into slopes. It is a rosbag with more detail which was used later in the project.

- The second rosbag starts at Alboi where the first rosbag stopped. The car follows a path into the A25 highway until the first exit.

  There are mostly cars in this one and it was a good rosbag to start with some tests in tracking objects.

5.1 Image Tracking

The development of object detection, tracking and labelling starts by processing and analyzing the image sequences. A labelling node was created in ROS where the features

in this chapter were implemented. This node subscribes to the camera images through its rostopic. In resemblance to the calibration, the image comes encapsulated in a ROS message to be processed in a callback function. In this function the image is analyzed.

5.1.1 Template Matching

The image is converted from the ROS message format into an OpenCV format so it can be easily manipulated. OpenCV treats images as matrices of pixel with (x, y) coordinates and RGB values. As the image sequences arrive, they are stored into a queue. This queue will be used later to look back to the previous frames and back-track the object.

When the node starts, a window opens (see figure 5.1) for the user to view the video stream recorded in the bag. The node also functions in real-time. In other words, the node can be executed by connecting the computer directly to the car, obtaining the images in immediately. In this window, the user can click on objects that may appear. To obey the scope of the dissertation, most of the objects captured were cars, vans and people. Some street signs were also tested. When the user clicks on an object, a callback function is triggered to process the mouse event and a bounding box appears around the selected target.

Figure 5.1: Window view with image sequences appearing

This callback function gets the (x, y) coordinates of the mouse click. The bounding box size is given by the distance to the object. If the distance is greater, the bounding box will be smaller and vice-versa. The distance to the object is given by the tri-dimensional spacial coordinates given by the MTT library explained further in the dissertation. This library uses the LIDAR sensors of the ATLASCAR 2, and since it was developed specifically for the ATLASCAR it is the optimal tool to detect and track objects in the field of view.

The upper third-part of the image is ignore as it is considered to be irrelevant content as most of it will be tall objects like trees or objects in the sky, like clouds. Also, it is meant for the image to be fused with the laser data, and since the ATLASCAR 2 is equipped with planar based sensor, the objects off the ground will be out of the scope. The bounding box tracks the object inside the bounding box. To accomplish this, template matching techniques are used. Firstly, the previous frames are saved. The node will check the queue of previous frames and store them to use them later.

The template matching strategy is used to track the selected target in the next frames. It begins by copying the source image to display to another OpenCV matrix and also creates a result matrix. The matching is now performed using a method implemented by OpenCV called `matchTemplate` which takes the source image, the patch (which is the Region of Interest (ROI) inside the bounding box), the result matrix and a matching method.

Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch). It is needed a source image which will be the actual frame and a template image which will be the ROI in the bounding box. The template matching purpose is to detect the highest matching area. to identify the matching area, the algorithm compares the template image with the source by iterating it through the source image. In other words, the patch will move on pixel at a time in the source image (left to right, up to down). At each cycle, a score is calculated. This score represents how good the match is in that position of the source image (or how similar the patch is to that particular area in that location).

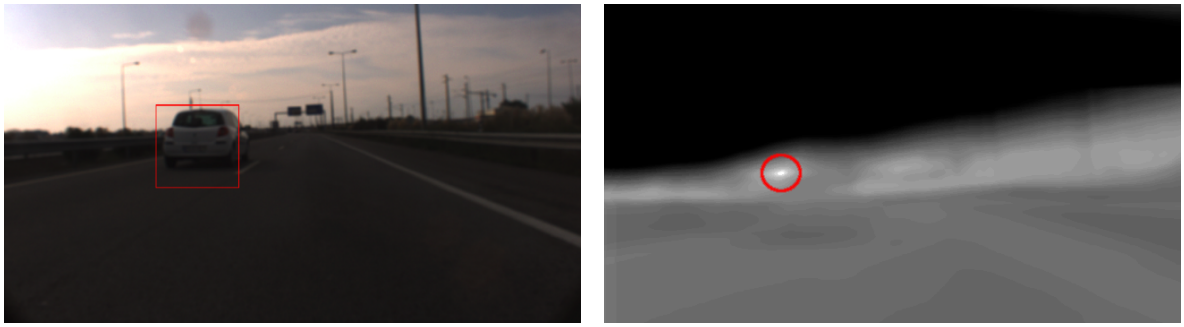


Figure 5.2: Template matching example inverted result matrix

For each location, the score is stored in the result matrix. Each location contains a metric score. The brightest points represent the positions where the matching was more similar. In figure 5.2 two images are displayed: on the left there is an example of a car being selected with a bounding box around it, and on the right is the inverted result matrix from the template matching algorithm. A red circle is around the brightest spot which indicates the location where the matching score was higher. The matching method used for this project is an equation based on the squared difference. The following equation is described by the the following formula

$$R(x,y) = \sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2$$

where R is the result matrix, T is the template (patch, or ROI in the bounding box) and I is the source image. The result is obtained by calculating the squared difference between the pixels in the template and in the area in that location in the source image.

```
1  struct BBox
2  {
3    int x;
4    int y;
5    int width;
6    int height;
7    int id;
8    string label;
9  };
```

Listing 5.1: BBox struct definition used for 2D datasets.

After the result matrix is filled, it is needed to locate the highest or lowest value in the result matrix depending on the matching method used. OpenCV implements a function called `minMaxLoc` which finds the global minimum and maximum in an array or matrix. Because it was used the squared difference as matching method, the locations with highest score are the ones where the difference is minimal.

In other words, the result matrix shows that the place with more similarities is the one where the values are lower. In figure 5.2 on the right, the result matrix is inverted for demonstration purposes. So in reality, the result matrix for this project show that the location with more similarities is in the position with the lowest value.

It is important to note that after each frame is received, the patch used for the next template matching cycle will be the ROI acquired in the previous frame. This means that the patch is update when a new frame is received to obtain better accuracy of the object's pose.

After the whole tracking is done, the back tracking is now performed. The reason why the backward tracking is done after the forward tracking is because if the back tracking would begin when a target is selected in the image before the front tracking, some time would be wasted to process the previous frames, losing some of the next frames used for the front tracking.

With a queue, the last frames are saved and at the moment of object selection those frames are copied and saved to be processed posteriorly. After the front tracking, the node gets the frames stored before the target selection and applies template matching. With this, the tracking is done in both directions.

5.1.2 2D Dataset and Playback

While tracking objects, the user is prompted to insert a label to the object. The user enters a class to which the object belongs to. The node saves several bounding boxes for each frame. To accomplish this, a bounding box data structure called `BBox` was implemented.

The `BBox` struct definition is presented in listing 5.1. The `BBox` presents its x and y coordinates, its width and height, an id, and a label. While the tracking is performed, several instances of `BBox` are created and stored in a map that relates the frame with the `BBox`.

When the tracking is complete, the user can opt to save the results or to discard them. If the frames are to be saved, the users enters the object label and a folder will be created with patches of the frames where the object appears. The user can also choose to label the

```
1  FRAME_ID
2  BOX_X BOX_Y WIDTH HEIGHT LABEL ID
3  ...
4  1083
5  815 663 155 104 car 4
6  1084
7  816 662 155 104 car 4
8  1142
9  482 584 152 150 van 5
10 1143
11 512 589 152 150 van 5
12 ...
```

Listing 5.2: 2D dataset example snippet

objects without saving the templates. In the end, a set of `BBox` instances are created and a dataset file can be created.

The dataset contains, for each frame, a set of bounding boxes that are defined by their coordinates, size, label and object ID. The map where the set of `BBox` is stored is iterated and printed to a file similar to the snippet in listing 5.2.

The dataset can be used later to playback the rosbag. Another node was developed with the aim to play the rosbag and identify the objects in the images using the dataset created previously. This rosnode starts similarly to the previous one, by subscribing to the camera ROS topic and send the image message to a callback function where it is processed. The message is converted into OpenCV format and the dataset file is read. A map similar to the previous is also created to be filled with the dataset information where it relates the objects in the bounding boxes to the frames in the sequence.

When a frame is received, its frame ID is retrieved. This ID is used to check which boxes in this frame. If this frame contains objects, a rectangle is draw in the image representing the bounding box acquired before. The box also features a legend with the object label and its ID. The color of the bounding box is randomly assigned, depending on its label. In other words, objects with the same label will have the same box color, making it easy to identify objects if the image has several different boxes. The resulting image is presented in figure 5.3. The image is then converted again into the ROS format and published to a topic.

5.2 Range Based Tracking

After the image processing, the detection, tracking and labelling continues using the LIDAR scanners equipped in the ATLASCAR 2. To develop this part, a continuation to the previous labelling node is added where the capabilities of the laser scans will be explored. The range based object tracking is performed with the MTT library developed by Almeida [4].

The MTT library was designed specially for the ALTASCAR making it the best tool to be used for this project. The MTT works with planar scanners to obtain perception although it receives a pointcloud as input. The MTT library supposes that the objects are all at the

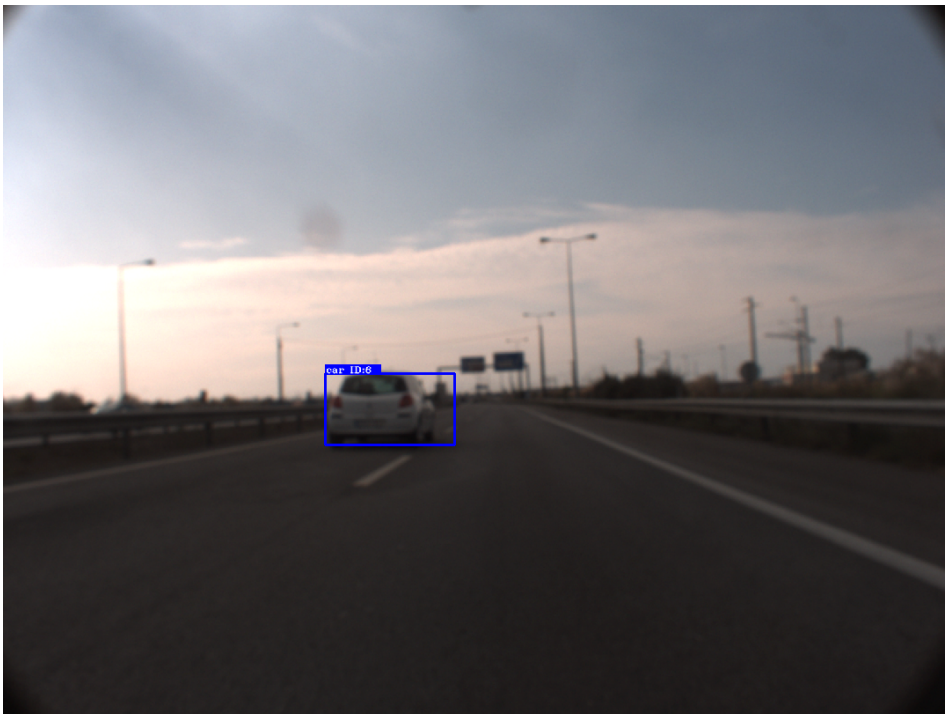


Figure 5.3: Playback example with a car

same height so the pointclouds are flattened. For the scope of this project this is no problem as most of the scanners used are planar except for the SICK LD-MRS. Assuming that the readings of this LIDAR are at the same height does not influence the results as the difference of the measures are minimal.

5.2.1 Multi Target Tracking

The node starts by subscribing to all topics where `laserScans` can be found. There are two SICK LMS151, one on each side of the ATLASCAR 2, giving two planar scans with a 270 degree aperture. The SICK LD-MRS features four planar scans. Each scan is submitted into a ROS topic totaling six topics, one for each SICK LMS151 and four topics for the SICK LD-MRS. The six topics are subscribed and the sensor messages are given to a callback function in the ROS format.

This callback functions gets the laser frame ID. This frame is the transformation frame, not to be misunderstood with an image frame. The frame ID is used to identify the laser scan in the callback function. The callback function converts the `laserScan` in to a pointcloud.

In figure 5.4 the `laserScans` can be observed individually. The readings from the central SICK LD-MRS are given by the white points. The two SICK LMS151 are distinguished by the red and green colors for the left and right respectively. The colors are chosen regarding nautical and aeronautical navigation lights for port and starboard positions.

In the image callback function, when an image frame arrives it creates a full pointcloud by merging the pointclouds of all `laserScans`. To do this, a transform listener is created to calculate the transforms at that given time between the two SICK LMS151 and the SICK LD-MRS. The PCL library is used here to blend the different `laserScans`. The PCL can

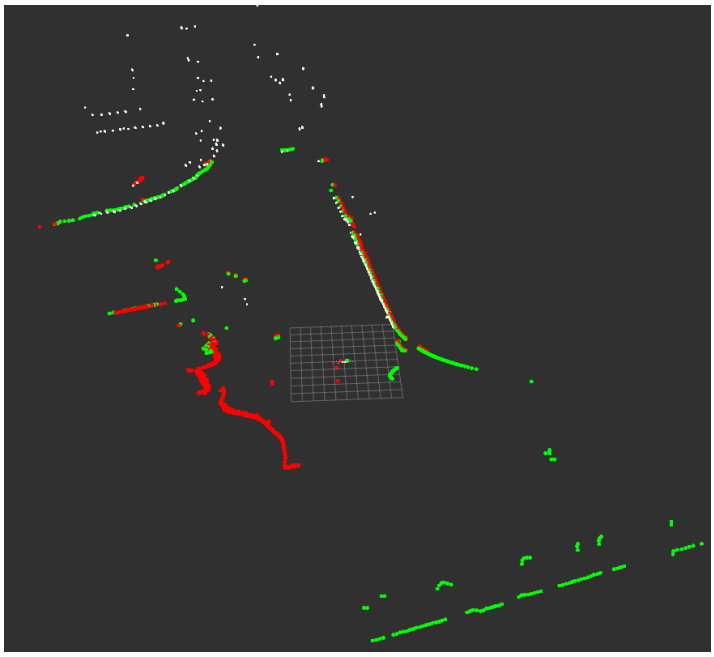


Figure 5.4: All the separated laserScans visualized with Rviz

concatenate pointclouds with the operator + making it easy to merge them all together. The pointclouds are converted from the ROS format to PCL format, concatenated, and ready to be processed by the MTT library.

Firsty, the final pointcloud is converted to the MTT format. The MTT library works in a data structure that has all points coordinates called `t_data`.

The pointcloud is iterated and a `t_data` instance is initialized and filled with the pointcloud information. The next step is to cluster the different objects found in the pointcloud. The clustering strategy is implemented by the MTT library. The MTT takes the full data and initializes a vector of clusters. Clusters are formed by points and its structure contains an id of the cluster, start and end points, total number of points, length of the cluster, among other variables and flags used for occlusion detection.

```
1    typedef struct
2    {
3      double x[2160],y[2160];
4      double r[2160],t[2160];
5      bool flag[2160],flag2[2160];
6      bool occlusion_data;
7      int initial_position[2160];
8      int n_points;
9    }t_data;
```

Listing 5.3: t_data struct definition

```
struct st_object
{
  vector<t_linePtr> lines;
  double cx,cy;

  double rmin,tm;
  double size;

  int id;

  bool object_found;
  bool partialy_occluded;

  double min_distance_to_existing_object;
};
```

Listing 5.5: st_object struct definition

```
  struct st_cluster
  {
    ///id of the cluster
    int id;
    ///start point
    int stp;
    ///end point
    int enp;
    ///total number of points
    int n_points;
    double rmin,tm;
    double cx,cy,cxp,cyp,cxe,cye;
    double r;
    bool partialy_occluded;
    double lenght;
  };
```

Listing 5.4: st_cluster struct definition

The clustering algorithm breaks the data into small groups (clusters) of points. A distance threshold is defined depending on the size of the pointcloud data and the distance between the points is calculated. If the distance between points is greater than the given threshold, then an object is most likely to be there. A set of points near that position will be grouped to form a segment. The clusters are then converted into objects. Objects are defined as a set of lines connecting the points that form the cluster.

The lines that form an object are defined by the polar coordinates and the Cartesian coordinates of the initial and final line points.

```
struct st_line
{
  ///Polar coordinates
  double ro,alpha;
  ///Initial and final line points
  double xi,yi,xf,yf;
};
```

Listing 5.6: st_line struct definition

The MTT then associates the targets found to a linked list of objects where the full description of the objects are stored. This list is later used to create a motion model where the estimated position of the objects and its velocity is calculated.

Some objects may become occluded, for example when a car passes in front of another. The MTT estimates the position of the occluded objects by creating a motion model using their velocity and so it is possible to track objects while they are out of the field of view but still in the surroundings.

The MTT operates around targets. The objects are converted into targets. Targets keep information about the object and its velocity plus their position and obstacle lines. A target list is published to a ROS topic.

For the sake of visualization, markers are created and placed in the location fo the objects. For each object in the target list, a marker is created with the ID of the object. The ID increments by one as a new object is found.

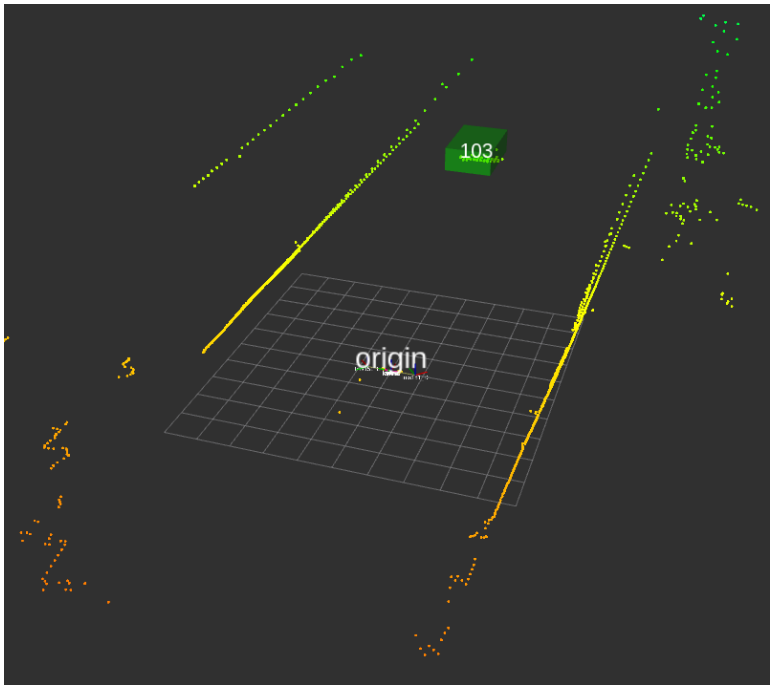


Figure 5.5: Visualization of a detected car with MTT in Rviz

In the position of the object, a visual marker is placed. There are also markers in the form of line strips in order to visualize the connection between the points of the pointcloud. An addition to the marker creation method was made, where 3D bounding boxes are created in the location of the objects.

The information in figure 5.5 was visualized using the Rviz tool. The MTT creates by default a marker at the origin where usually the front of the ATLASCAR 2 is (depending on the transformations).

In the figure, the processed pointcloud can be seen, where the LIDAR `laserScans` messages are all merged into. It can also be seen a green 3D bounding box with the ID 103, meaning that an object was found at that location. The object was in fact a car traveling in front of the ATLASCAR 2.

5.2.2 3D Datasets

To develop datasets and include the 3D information of the tracked targets, some changes have been made to the structure of the `BBox` in which the 3D coordinates of the objects have been added.

```
1   struct BBox
2   {
3     int x;
4     int y;
5     int width;
6     int height;
7     int id;
8     string label;
9     // 3D position
10    double x3d, y3d, z3d;
11  };
```

Listing 5.7: BBox struct definition with 3D capabilities

While tracking an object in the image, an object in the MTT of ranged based sensors is selected and its ID is retrieved. This object is followed and its position is given to the labelling node to print a dataset file (see listing 5.8) with the full information about the objects whereabouts.

The dataset header was updated to contain the 3D information and the contents now present the coordinates in space of the object regarding the ATLASCAR 2 position. Analyzing the snippet in listing 5.8, the `3D_Z` values can be seen set to 0.5. The explanation for this is that the MTT library implements perception for planar sensors which only give x and y coordinates. For this reason, the height for all objects is estimated as 0.5 (half a meter).

```
FRAME_ID
BOX_X BOX_Y WIDTH HEIGHT LABEL ID 3D_X 3D_Y 3D_Z
...
1063
693 600 218 218 car 1 19.1706 1.64176 0.5
1064
692 597 218 218 car 1 19.5359 1.61985 0.5
1144
570 597 145 145 van 2 25.7349 2.61821 0.5
1145
590 602 145 145 van 2 25.7349 2.61821 0.5
...
```

Listing 5.8: Snippet of the dataset with 3D capabilities

Chapter 6

Results

6.1 Ball detection improvement

6.2 Image Object Tracking

6.3 Sensor Object Tracking

6.4 Dataset creation

Chapter 7

Conclusions and Future Work

Bibliography

[1] LARlabs. ATLAS project.

[2] David Tiago Vieira da Silva. Multisensor Calibration and Data Fusion Using LIDAR and Vision. page 107, 2016.

[3] José Correia. Unidade de Perceção Visual e de profundidade para o ATLASCAR2. pages 1–98, 2017.

[4] Jorge Manuel Soares De Almeida. Seguimento ativo de agentes dinâmicos multivariados usando informação vectorial Active Tracking of Dynamic Multivariate Agents using Vectorial Range Data. 2016.

[5] Fabian Kröger. Automated Driving in Its Social, Historical and Cultural Contexts. In *Autonomous Driving*, pages 41–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[6] The Milwaukee Sentinel. 8 Dec 1926 - " 'Phantom Auto' will tour city".

[7] Dean A Pomerleau. An Autonomous Land Vehicle In a Neural Network. 1989.

[8] Waymo. Waymo.

[9] Audi MediaCenter. Audi at the IAA 2017: Autonomous driving in three steps.

[10] Singapore-MIT Alliance for Research and Technology. SMART - Singapore-MIT Alliance for Research and Technology.

[11] Pauline Teo. SMART launches first Singapore-developed driverless car designed for operations on public roads.

[12] Karlsruhe Institute of Technology. The KITTI Vision Benchmark Suite.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset.

[14] Boston Didi Team. KITTI Github Respository.

[15] Leonid Sigal, Alexandru O Balan, Michael J Black, A O Balan, and M J Black. HUMANEVA: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *Int J Comput Vis*.

[16] ETHZ (Eidgenössische Technische Hochschule Zürich). Moving Obstacle Detection in Highly Dynamic Scenes.

[17] EPFL (École polytechnique fédérale de Lausanne). Multi-camera pedestrians video — CVLAB.

[18] Jorge Manuel Soares De Almeida. Multi target tracking.

[19] Luciano Spinello, Rudolph Triebel, and Roland Siegwart. Multiclass Multimodal Detection and Tracking in Urban Environments. pages 125–135. Springer, Berlin, Heidelberg, 2010.

[20] MITSUBISHI MOTORS. Specifications — i-MiEV.

[21] SICK. LMS151-10100 — Detection and ranging solutions — SICK.

[22] SICK. LD-MRS400001 — Detection and ranging solutions — SICK.

[23] PointGrey. Zebra2 2.0 MP Color GigE / HD-SDI (Sony ICX274).

[24] ROS Wiki. rviz.

[25] ROS Wiki. rosbag.

[26] OpenCV. Background Subtraction.

[27] OpenCV 2.4.13.6 documentation. Eroding and Dilating.

[28] OpenCV 2.4.13.6 documentation. Canny Edge Detector.

[29] OpenCV 2.4.13.6 documentation. Sobel Derivatives.

[30] Satoshi Suzuki and Keiichi A. Be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing*, 30(1):32–46, 1985.