Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2018

Nuno Miguel
Soares Silva

Anotação e Seguimento Semi-Automático de Alvos para Condução Autónoma

Semi-Automatic Labelling and Tracking of Targets for Autonomous Driving

Nuno Miguel Soares Silva

Anotação e Seguimento Semi-Automático de Alvos para Condução Autónoma
Semi-Automatic Labelling and Tracking of Targets for Autonomous Driving

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 e de Vitor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica.

o júri / the jury

presidente / president

Prof. Doutor António José Ribeiro Neves
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Doutor João Manuel Leite da Silva
Investigador Sénior, Altran Portugal (arguente)

Prof. Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar, Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

À minha família em especial à minha mãe e ao meu pai pela educação que me deram e por estarem comigo em todos os momentos. Aos professores Paulo Dias e Vítor Santos pela sua orientação, sugestões e dedicação ao projeto. Também ao professor Miguel Riem Oliveira pela sua prontidão a ajudar e disponibilidade.

Por último, mas não menos importante, aos meus amigos e pessoal do DETI que durantes estes anos me fizeram evoluir como pessoa e que contribuíram para a minha formação académica.

Palavras-chave

Navegação autónoma; Veículos autónomos; ATLASCAR; Agrupamento de dados; Calibração; Anotação de dados; Deteção de Objetos; Fusão de Dados; ADAS; LIDAR; Processamento de Imagem.

Resumo

No âmbito do projeto do ATLASCAR2, esta dissertação baseia-se no desenvolvimento (usando Robot Operating System (ROS)) de um sistema de assistência à condução que implementa uma interface para deteção, seguimento e anotação de alvos na estrada. A deteção e o seguimento são feitos a partir de dados de sensores Light Detection And Ranging (LIDAR) e de uma câmara. Numa primeira fase é implementado um algoritmo para ser usado na câmara baseado na aparência do alvo a ser seguido. Seguidamente, é desenvolvido um algoritmo baseado no alcance usando dados adquiridos nos sensores para seguir objetos num espaço tridimensional. Finalmente, como é possível detetar e seguir objetos usando a imagem e os lasers, a combinação dos algoritmos é feita projetando o que é capturado pelos sensores na imagem da câmara, sendo possível obter um seguimento mais preciso e robusto dos alvos na estrada. Para avaliar o algoritmo alguns "datasets" foram usados com a anotação dos vários alvos detetados e seguidos. Para efetuar este trabalho é necessário que os sensores e a câmara estejam devidamente calibrados. Para este efeito, a calibração entre os vários dispositivos é feita através de uma aplicação que, ao passar uma bola em frente dos sensores, é possível descobrir os valores das posições de cada sensor relativamente a um dispositivo dado como referência. Ao nível da calibração, esta dissertação inclui também uma melhoria do algoritmo de deteção da bola na imagem obtida pela câmara.

Keywords Autonomous driving; Autonomous vehicles; ATLASCAR; Data Clustering; Calibration; Data Labelling; Object Detection; Data Fusion; ADAS; LIDAR; Image Processing.

Abstract In the scope of the ATLASCAR2 project, this dissertation is based on the development (using ROS) of a driving assistance system that implements an interface to detect, track and label targets on the road. The detection and tracking are done using LIDAR sensor data and a camera. Firstly, an algorithm is implemented to be used in the camera based on the appearance of the target to be tracked. Next, a range based algorithm is developed using the data acquired from the sensors to follow objects in a tridimensional space. Finally, because it is possible to detect and track objects using the image and the lasers, the combination of the algorithms is done by projecting what is captured from the sensors in the camera image, being possible to obtain a more accurate and robust tracking. To evaluate the algorithm some datasets were used with the labelled data from the several detected and followed objects. To perform this work the sensors and the camera need to be properly calibrated. To do this, the calibration between the several devices was done using an application that, by passing a ball in front of the sensors, the position values of each sensor relatively to a given reference device are found. Within the calibration, this dissertation also includes an improvement of the ball detection algorithm in the image obtained by the camera.

Contents

List of Figures

List of Tables

Listings

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
LARTk	Laboratory of Automation and Robotics (LAR) Toolkit
MTT	Multi Target Tracking
HSV	Hue, Saturation and Value
RCA	Radio Corporation of America
ATC	Advanced Technologies Center
DARPA	Defense Advanced Research Projects Agency
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 growing in the past decades in the automobile industry and in the academic environment.

It is important in AD and ADAS to implement machine learning methods so that the vehicles recognize what objects are in their surroundings. Therefore, labelled data is necessary to train machine learning algorithms. Data labelling is the solution to create datasets that will serve as input to learning algorithms.

This thesis is focused on the research of methods to register data into labels using a camera and LIDAR sensors in ATLASCAR 2, so that the vehicle can later create a model of the objects it will detect and follow.

This dissertation will also improve some previous work, namely the calibration process in ATLASCAR 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 developed 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 (LARlabs 2018). Currently, ATLASCAR 2 is the new full sized prototype being used for research equipped with LIDAR sensors and a camera.

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.

As the project grew, it evolved into full-sized prototypes: the ATLASCARs. ATLASCAR1

Figure 1.1: ATLAS project small-sized prototypes LARlabs 2018

(figure 1.2) is the first full-sized platform and it is based on a Ford Escort Station Wagon. The ATLASCAR1 was equipped with several LIDAR sensors and cameras. Data about its environment is gathered by the scanners which is then processed building perception into the car allowing the car to actuate and move autonomously.

Figure 1.2: ATLASCAR1 based on the Ford Escort platform LARlabs 2018

The ATLASCAR1 brought successful results. In the end, the vehicle was able to move and execute maneuvers autonomously in small and controlled places. The ATLASCAR1 was then replaced by a more recent vehicle. The ATLASCAR2 (figure 1.3) 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 ATLASCAR2 is 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 LARlabs 2018

1.2 Motivation

AD and ADAS often make use of machine learning. Deep learning algorithms utilize neural and convolutional networks for images and range-based sensor data. In the fields of machine learning, images are often used as input templates to create object models. In an image sequence it is possible to track an object and obtain samples of the target's location. This way it is possible for an algorithm to automatically recognize an object.

While registering the position of the objects in an image sequence, it is important to tell what those objects are. Labelling is the act to classify objects in a certain group. As the annotation is done, when selecting an object, the user should enter a label that identifies the target making it easy for the learning algorithm to recognize to object afterwards.

The objective and importance of tagging samples of images with a label is to assign metadata in the form of a keyword to later allow an application to retrieve this information and easily construct a database. However, most labelling for images is currently done with non optimized manual procedures.

In the end of this dissertation, the ATLASCAR2 will have a labelling system that fuses images retrieved from the camera with the laser data, creating a semi automatic object tracking system that offers the possibility to retrieve various image template sequences and store metadata about them. This metadata contains, for instance, the label, object related identification markers and position of the object in the image and in the real world relatively to the ATLASCAR2.

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 ROS. Secondly, the development of another ROS package used for semi-automatic detection and labelling of objects in the field of view.

The calibration package was already developed by Vieira da Silva 2016. 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.

The detection, tracking and labelling system will combine the image data and the laser data from the sensors to create a semi-automatic tracking system that allows the user to associate an object category to the target and create datasets with metadata from the labelling.

1.4 Document Structure

This document is composed by seven chapters including the introduction. The second chapter describes the related work previously done on ATLASCAR 2 as well as a literature review detailing the most significant milestones on the history of autonomous driving and a research on image labelling datasets.

In the third chapter, the experimental structure of this dissertation will be described depicting the hardware (ATLASCAR2 and sensors) and the software (ROS, LAR Toolkit (LARTk) and Point Cloud Library (PCL)).

In the fourth chapter, the implementation of the calibration node will be explained presenting its features, the base algorithm and how the calibration package was modified in order to improve the ball detection.

In the fifth chapter, the detection, tracking and labelling node development will be explained, firstly by describing how the image tracking is done, then clarifying how to track objects using the LIDAR sensors and finally using both the image and laser sensor data. This chapter also features the created datasets and some extra tools used to aid in the labelling.

The sixth chapter presents the results of the ball detection and integration with the calibration package will be shown and the outcome of the labelling node will also be analyzed using some datasets produced by the same.

The final chapter the conclusions of this thesis are presented and some future work related to the scope of this dissertation is proposed.

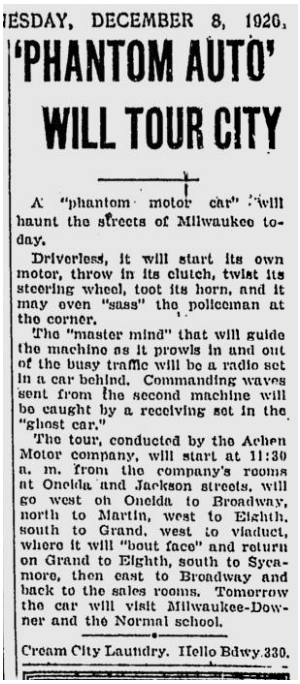
Chapter 2

Literature Review

This chapter presents some milestones in the history of autonomous driving and also some of the most relevant projects made previously in the ATLASCAR2. It also presents examples of ADAS datasets for labelled data.

2.1 Milestones on the History of autonomous driving

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 (Kröger 2016). 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 (The Milwaukee Sentinel 1926) in Milwaukee (see figure 2.1). 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. Hollo Bdwy 330.

Figure 2.1: The Milwaukee Sentinel 1926, 8 December - '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 demonstrated completing a test route of one mile (Kröger 2016).

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 (Pomerleau 1989).

Various autonomous vehicles competitions have been held. The first long distance competition for driverless cars was the Defense Advanced Research Projects Agency (DARPA) Grand Challenge (DARPA 2018). The event was open to teams and organizations from around the world. Teams have participated from high schools, universities, businesses and other organizations, bringing a wide variety of technological skills to the race. The challenge offered high value money prizes to the winners. Because the reward was so high, the contest brought various state-of-the-art autonomous vehicles that showcased the solutions implemented in the platforms featuring new ideas that used the most recent technologies (Montemerlo et al. 2006 and Thrun et al. 2007).

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 (Waymo 2018).

Figure 2.2: Waymo's Jaguar I-PACE Waymo 2018

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.

Figure 2.3: Ford Fusion Uber ATC car (Uber Advanced Technologies Group 2018)

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 (Audi MediaCenter 2018 and Andreas Herrmann, Walter Brenner 2018).

Figure 2.4: The new Audi A8 (Audi MediaCenter 2018)

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 (SCOT) vehicle (figure **??**), conducted by the Singapore-MIT Alliance for Research and Technology (SMART) (Singapore-MIT Alliance for Research and Technology 2018). 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 (Andreas Herrmann, Walter Brenner 2018). The SCOT vehicle also relies on LIDAR sensors similar to ATLASCAR 2 (Teo 2018).

Figure 2.5: SCOT - Shared Computer-Operated Transit vehicle (Singapore-MIT Alliance for Research and Technology 2018)

To resume this chapter, in figure 2.6 a timeline with the referred milestones is presented.

2.2 Related work on ATLASCAR2

In this section it is referenced previous work done at LAR relevant for this thesis.

2.2.1 Multisensor Calibration and Data Fusion Using LIDAR and Vision

The calibration procedure for data fusion of ATLASCAR was developed by Vieira da Silva 2016. 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 of 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 (see figure 2.7) that allows the user to configure the various sensors to be calibrated. The estimated positions between sensors are achieved with sensor data fusion.

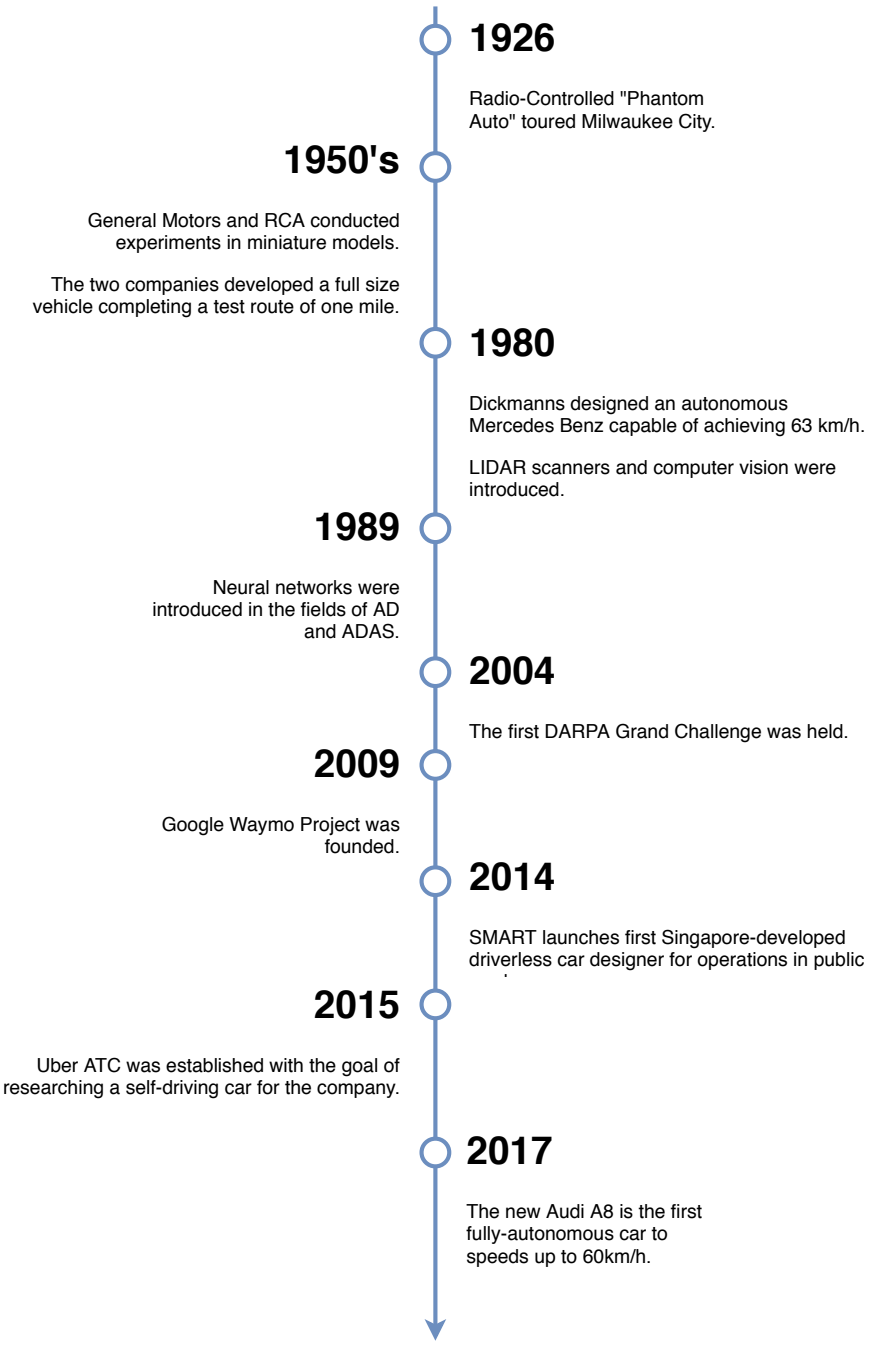


Figure 2.6: Timeline with some milestone of autonomous driving history

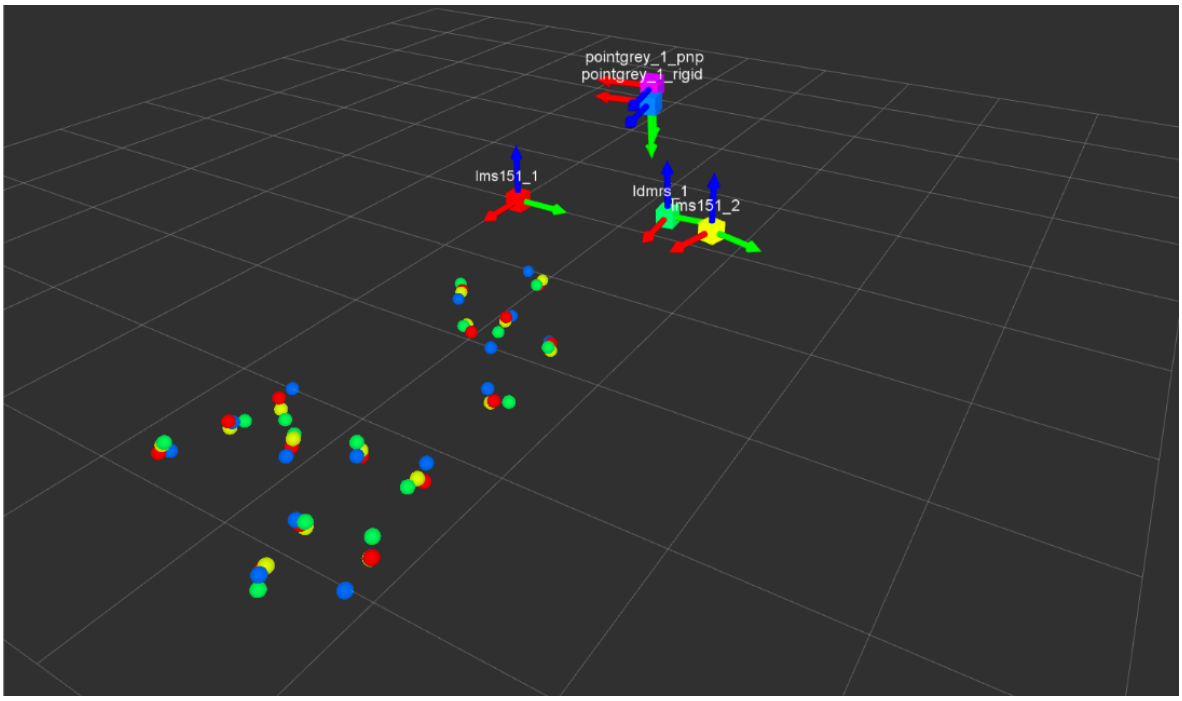


Figure 2.7: Calibration package result visualization (Vieira da Silva 2016)

2.2.2 Visual and Depth Perception Unit for ATLASCAR2

The infrastructure to build a visual and depth perception unit for the ATLASCAR was developed by Correia 2017. 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 Vieira da Silva 2016. 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 (see figure 2.8).

2.2.3 Active Tracking of Dynamic Multivariate Agents

Methods to detect and follow targets using the LIDAR on the ATLASCAR was developed by Soares De Almeida 2016. The thesis is based in the tracking of multiple targets 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.

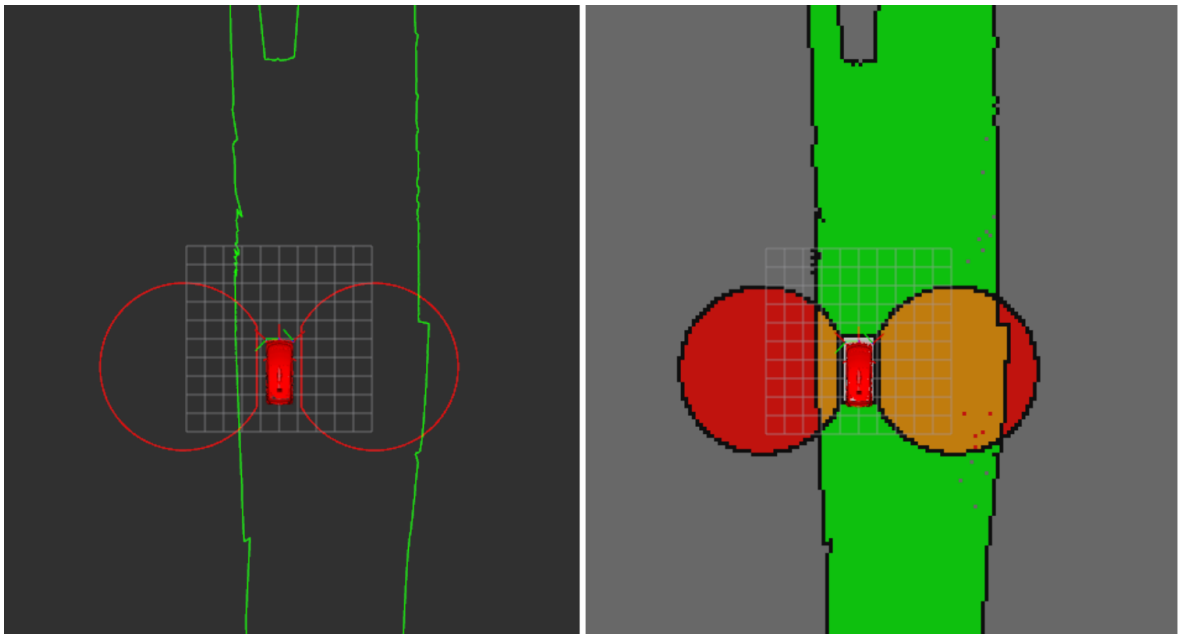


Figure 2.8: Free space represented by a polygon (left) and an occupancy grid (right) (Correia 2017)

2.3 ADAS Datasets - Examples of Labelled Data

This sections presents some relevant work and its results in public labelled datasets for ADAS projects.

2.3.1 KITTI Dataset

Probably the most well-known dataset in the fields of AD is the KITTI (Karlsruhe Institute of Technology 2018). The KITTI dataset was captured with a Volkswagen station wagon (figure 2.9) used in mobile robotics and AD research. The KITTI benchmark suite started in 2012 at Karlsruhe Institute of Technology with 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.

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 2.11) 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 in the 2D and 3D space (Geiger et al. 2013).

The development kit used for the KITTI database contains C++ and MATLAB code to read the sensor data and write dataset results. The data development kit used is provided on the KITTI Website (Karlsruhe Institute of Technology 2018). It contains a MATLAB demonstration code with C++ wrappers.

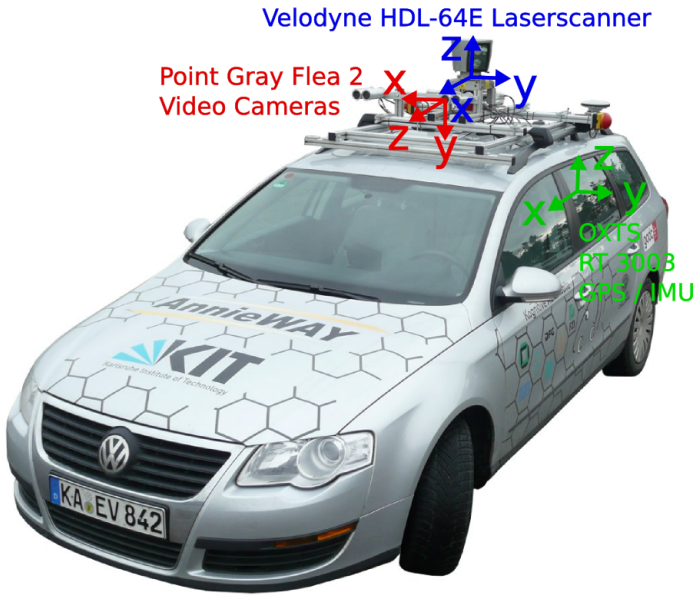


Figure 2.9: Volkswagen Station Wagon used in the KITTI Dataset (Karlsruhe Institute of Technology 2018)

The demonstration script of the KITTI development kit shows how 3D boxes can be read from the dataset files and projected into the image plane of the cameras (see figure 2.10).

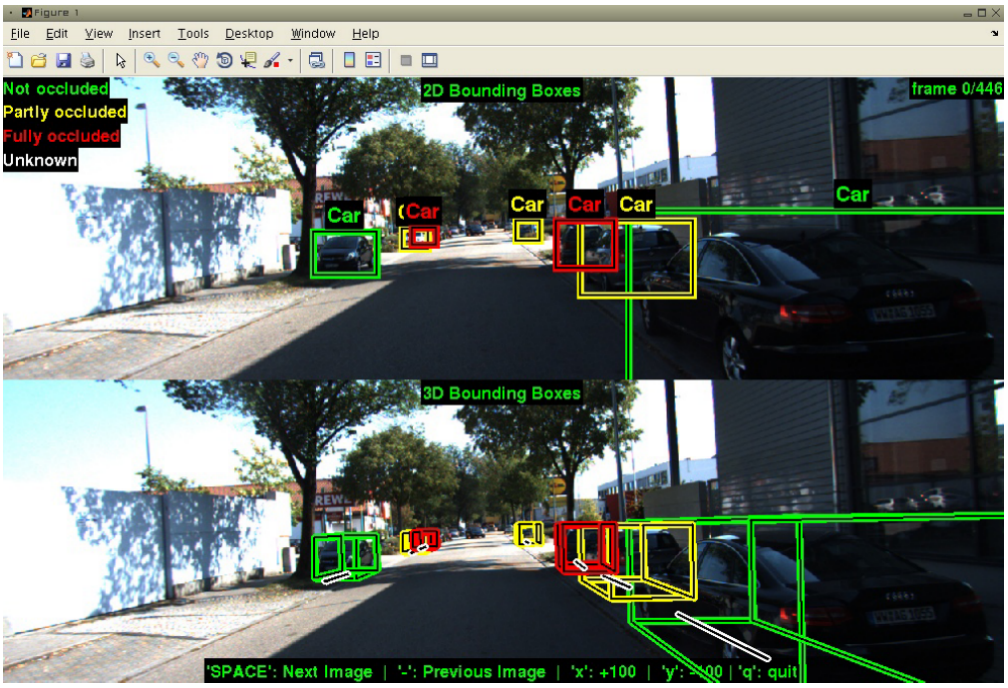


Figure 2.10: KITTI development kit showing 2D and 3D bounding boxes around several cars (Karlsruhe Institute of Technology 2018)

The data is processed and inserted into MATLAB structures and arrays. The KITTI

```
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 presenting frame_id, object_id, label, truncated and occluded flags, alpha, left top and right bottom coordinates, height, width and length, 3D coordinates (x,y,z) and rotation

database also uses the PCL to process and gather the pointclouds obtained from the LIDAR sensors.

Listing 2.1 shows 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 (Boston Didi Team 2018).

Analyzing the snippet, it is possible to locate 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.

Figure 2.11 depicts a result from an image sequence using the KITTI dataset.

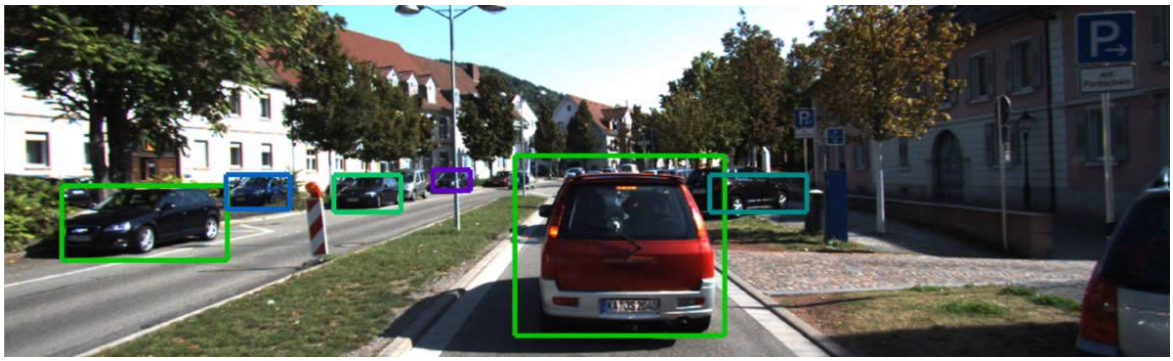


Figure 2.11: Sequence example of tracking by detection using the KITTI dataset (Karlsruhe Institute of Technology 2018)

2.3.2 Berkeley DeepDrive Dataset

The DeepDrive Dataset is a recent ADAS labelled dataset from the University of California, Berkeley.

The Dataset consists of box annotations, region annotations and detection of objects, lanes and drivable areas (Yu et al. 2018). The only source data for each dataset is a video captured with a camera on a vehicle. Their labelling system is a web-based tool (see figure 2.12).

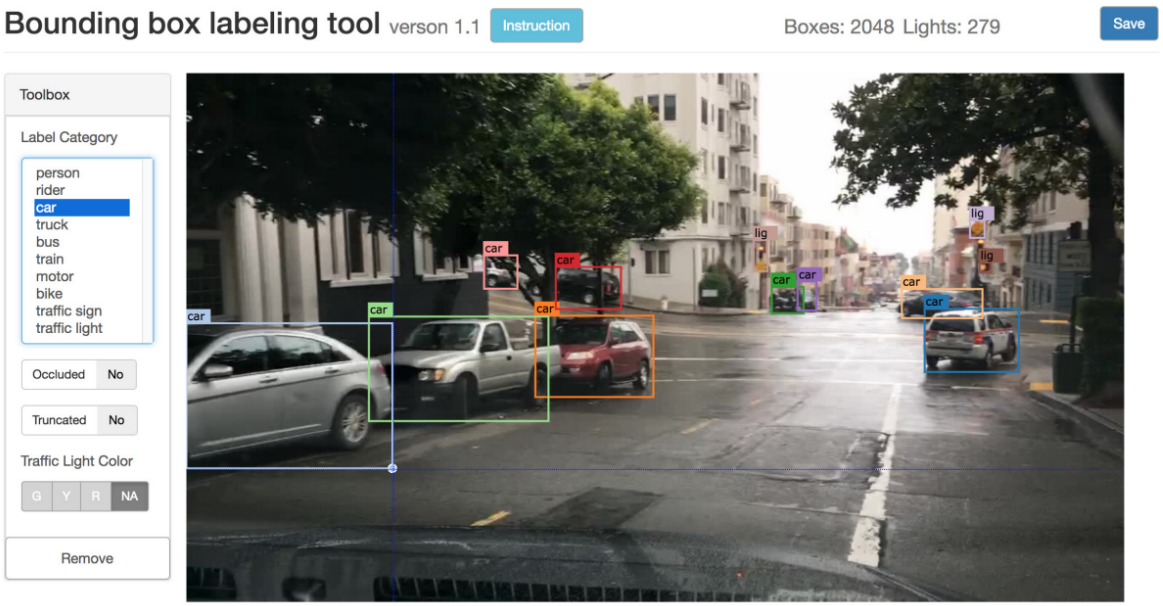


Figure 2.12: DeepDrive Dataset Annotation Tool (Yu et al. 2018)

The labelling has a semi-automatic and a manual mode. In this annotation application the objects of interest are suggested with a bounding box and a category (label). The size of the bounding box and the category can be edited if the system fails. The objects are detected using a previously trained object detection model.

In listing 2.2 a snippet from a file of the DeepDrive dataset is depicted. The dataset is

```
...
    "frames": [
    {
      "timestamp": 10000,
      "objects": [
      {
        "category": "car",
        "id": 0,
        "attributes": {
        "occluded": true,
        "truncated": false,
        "trafficLightColor": "none"
        },
        "box2d": {
        "x1": 555.647397,
        "y1": 304.228432,
        "x2": 574.015906,
        "y2": 316.474104
        }
      },
      {
        "category": "car",
        "id": 1,
        "attributes": {
        "occluded": true,
        "truncated": false,
        "trafficLightColor": "none"
        },
        "box2d": {
        "x1": 554.116689,
        "y1": 318.004813,
        "x2": 567.89307,
        "y2": 328.719775
        }
    },
...
```

Listing 2.2: DeepDrive dataset file snippet.

presented in the JSON format. For each frame there is a timestamp and a set of objects. Each object is identifies with a category, an ID, attributes that indicate if the object is occluded, truncated, and if the object is a traffic light which light color is on. The position is represented by the 2D bounding box position $(x1, y1, x2, y2)$.

2.3.3 HumanEva II Dataset

The HumanEva II Dataset from the Max Planck Institut Informatik (MPII) was also an interesting dataset, although it is used mainly for pedestrian detection.

This dataset appears with the need to represent information about detection and tracking of humans and their poses captured by a single image camera. The HumanEva dataset development kit includes several MATLAB modules, each one implementing a feature. Some modules refer to the body pose, others to image stream processing, writing the dataset results, and so on. Each script implements a chunk of the system that gathers the data and shapes it into MATLAB structures to be processed and to create the dataset.

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. For each frame in the image sequences there are several bounding boxes with the respective coordinates (Sigal et al. 2009).

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 are several bounding boxes with coordinates $(x1, y1, x2, y2)$, a score, silhouette, articulation and viewpoint id.

2.3.4 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 (Ess et al. 2009). Just like the previous datasets, its creation is based in MATLAB scripts and the data is gathered and stored in MATLAB structures. 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 (ETHZ (Eidgenössische Technische Hochschule Zürich) 2018). 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 (Biliotti, Antonini and Thiran 2015). The data from the cameras is captured and processed with MATLAB scripts and some algorithms in C++.

```
<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 2.13: Example image of the HumanEva dataset that generated the snippet in listing 2.3 (Max-Planck-Gesellschaft 2018)

```
...
"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 (ETHZ (Eidgenössische Technische Hochschule Zürich) 2018)

Figure 2.14: One of the images of the ETHZ dataset that generated part of the snippet in listing 2.4 (ETHZ (Eidgenössische Technische Hochschule Zürich) 2018)

```
...
1 80 45 99 98 9363 0 0 1 "PERSON"
1 80 45 99 98 9364 0 0 0 "PERSON"
1 77 45 96 98 9365 0 0 1 "PERSON"
1 74 45 93 98 9366 0 0 1 "PERSON"
1 71 46 90 99 9367 0 0 0 "PERSON"
2 81 45 110 126 0 0 0 0 "PERSON"
2 80 45 109 126 1 0 0 1 "PERSON"
2 80 45 109 126 2 0 0 1 "PERSON"
2 80 45 109 126 3 0 0 1 "PERSON"
2 80 45 109 126 4 0 0 1 "PERSON"
...
```

Listing 2.5: EPFL dataset file snippet (EPFL (École polytechnique fédérale de Lausanne) 2018)

Figure 2.15: One of the images of the EPFL dataset that generated part of the snippet in listing 2.5 (EPFL (École polytechnique fédérale de Lausanne) 2018)

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 (EPFL (École polytechnique fédérale de Lausanne) 2018). 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.

```
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 the 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.

2.3.5 Resume

To summarize this section, after analyzing these datasets are how they are created, it is important to look at what is stored in the data structures. Some datasets use well known data structures such as XML (HumanEva) or JSON (DeepDrive), and some datasets use their own set of data where each line corresponds to an entry.

To design a dataset for the ATLASCAR it is necessary to decide which construction to use. To simplify the complexity of the files, an adapted approach used in the KITTI dataset will be used, where each line is an entry for a different object.

Analyzing the datasets, the most common and relevant information in all files is the position of the targets, their classification and identification. So it is fundamental for the ATLASCAR dataset to contain, for each frame, 2D and 3D coordinates of the target, a label, and an identification number.

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 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.

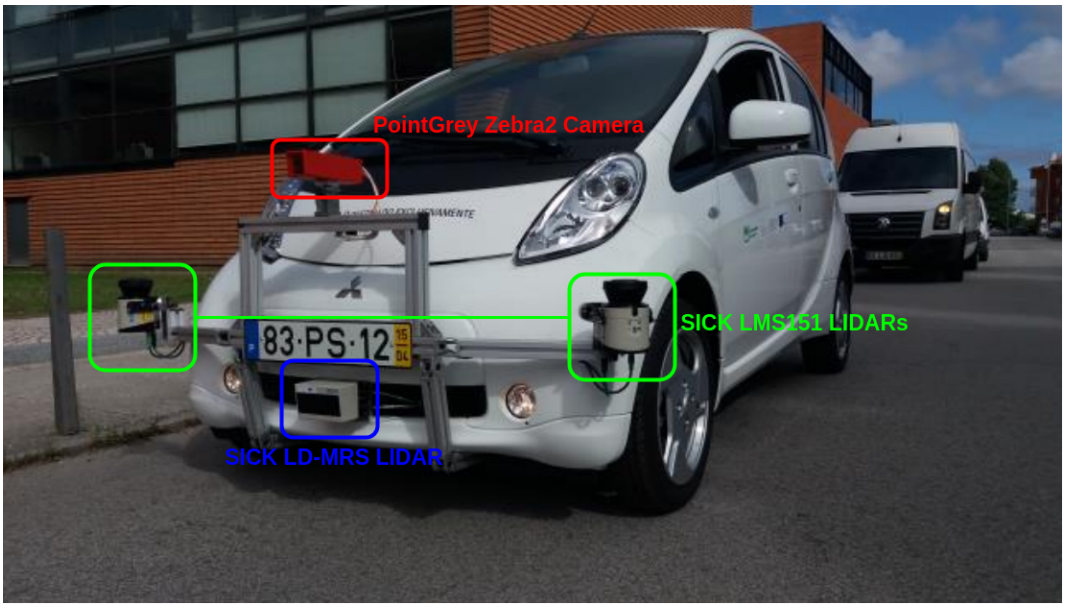


Figure 3.1: The ATLASCAR 2 based on the Mitsubishi i-MiEV platform equipped with a camera and several LIDAR sensors

Table 3.1: Mitsubishi i-MiEV technical specifications (MITSUBISHI MOTORS 2018)

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 LIDAR Sensors

The sensors equipped in the ATLASCAR 2 are two SICK LMS151 LIDAR, a SICK LD-MRS LIDAR and a PointGrey Zebra 2 Camera. The sensors have been mounted in the front of the car in an aluminum infrastructure designed by Correia 2017. These devices are connected to a network switch installed in the car to which a computer can be plugged to receive the data from the sensors.

3.2.1 SICK LMS151

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 (SICK 2018b).

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

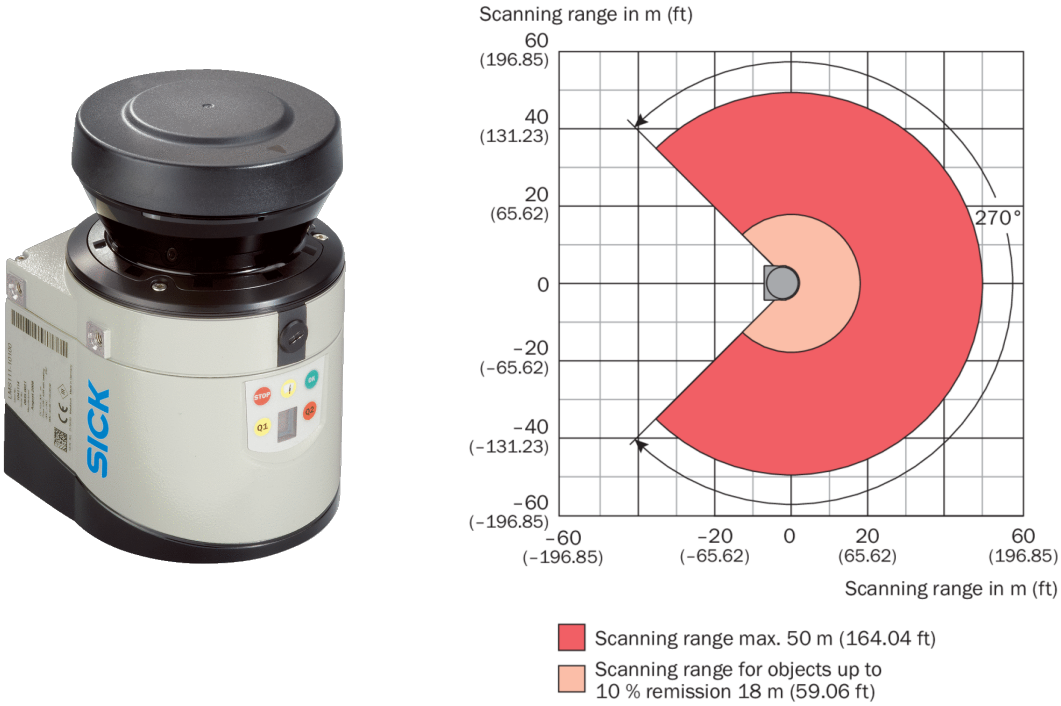


Figure 3.2: The SICK LMS151 LIDAR and its operating range

0.5° between readings. Each message will transmit a total of 540 points per scan in polar coordinates (r, θ). SICK 2018b

3.2.2 SICK LD-MRS

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 (SICK 2018a).

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

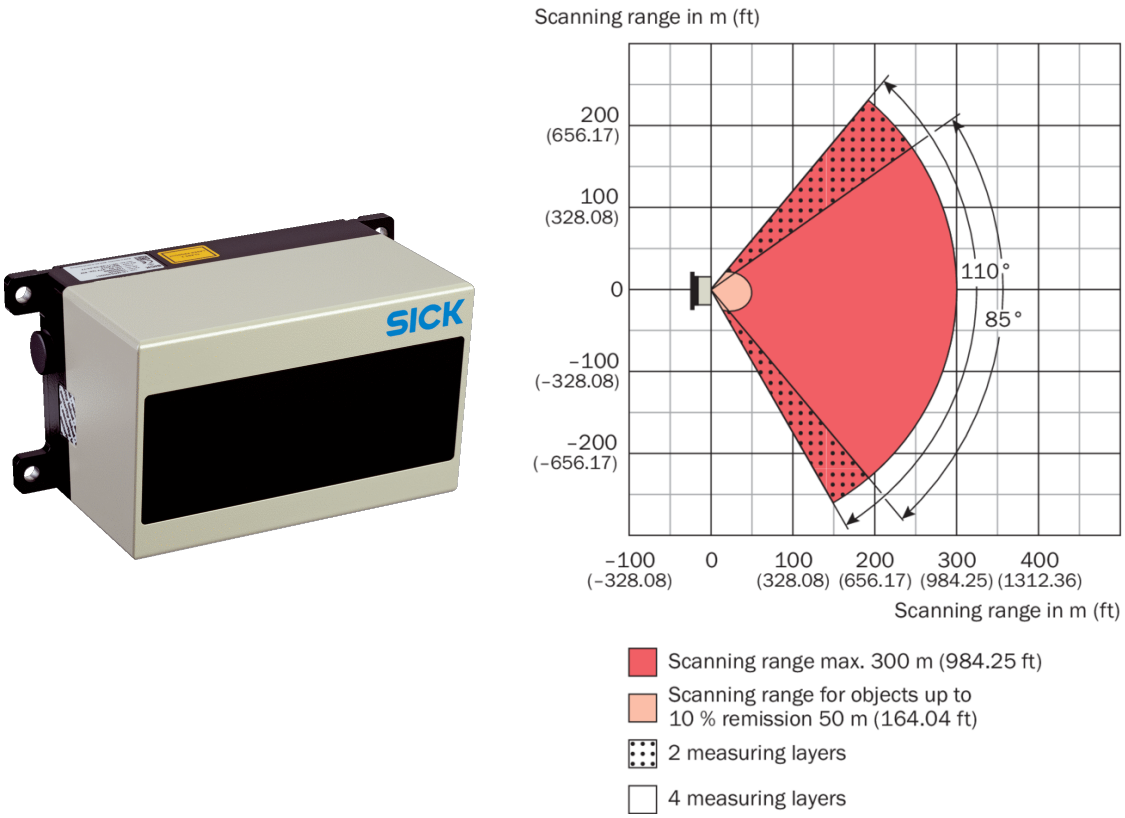


Figure 3.3: The SICK LD-MRS LIDAR and its operating range

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 (SICK 2018a).

3.3 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 (PointGrey 2018). The camera is inserted in a case made with 3D printing and designed by Correia 2017. 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 increase the network bandwidth and image processing times would be longer. 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.

3.4 Software

This section discusses 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 Vieira da Silva 2016) 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++.

3.4.1 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.

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 (ROS Wiki 2018b).

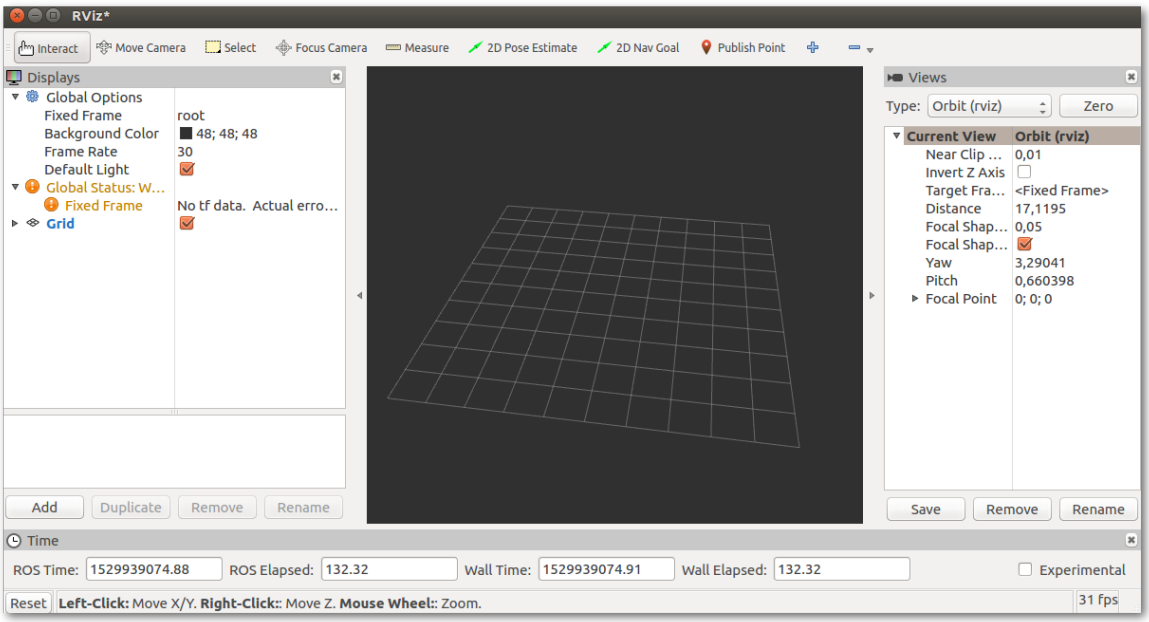


Figure 3.5: Rviz GUI

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 is to replicate the working environment of the ATLASCAR 2 off-line. 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 (ROS Wiki 2018a).

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 used 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.

Rqt_bag

The rqt_bag is a GUI to replay and display ROS bag files. In this program it is possible to show bag message contents, view image messages, plot message from selected topics, publish messages from selected topics and export messages to a new bag. The rqt_bag is useful for the labelling process as it presents a timeline in which the user can select the time by clicking on it.

3.4.2 LAR Toolkit

The LARTk is a software suite developed by members of LAR. The projects involved in the LARTk are based in the development of robotic solutions namely for the ATLAS project. The toolkit is constituted by a set of packages. Two packages in particular will be used for this dissertation: the multi-sensor calibration package and the MTT package.

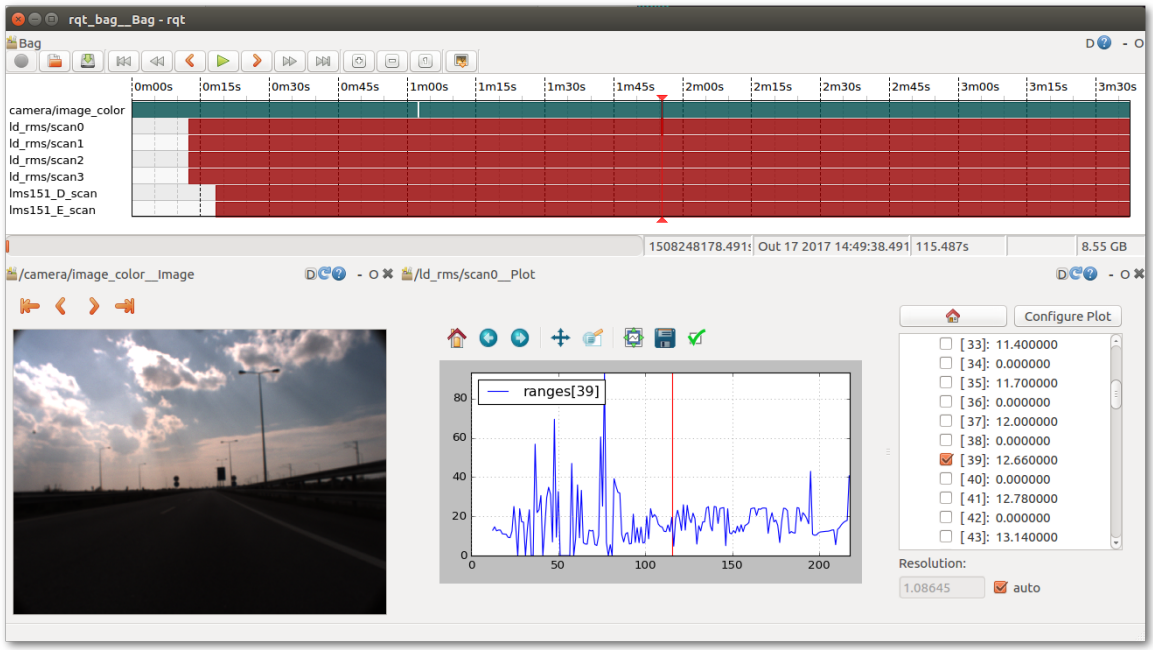


Figure 3.6: rqt_bag GUI with the several topics present in the bag, an image viewer and a plot with readings of the ranges.

Multi-sensor calibration package

The multi-sensor calibration package is a package developed by Vieira da Silva 2016. The package contains a 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.

To calibrate, the user should roll a ball (see figure 4.2) in front of the sensors in order to create a pointcloud of ball centers, and then align them. The user adds the sensors that will be calibrated and inserts the IP address of the sensor. One of the sensors will be defined as the reference sensor.

Some configurations can be done in the *Options* menu. The ball diameter can be defined here as well as the number of calibration points and distance between points. The calibration also has two acquisition types: automatic or user prompt.

In the automatic mode, while the ball rolls in front of the sensors the calibration programs automatically detects the ball center and publishes it. In the user prompt mode the user chooses when the ball should be captured.

In the end a file with the transformation matrices with the relative position of the sensors relatively to the reference sensor will be written.

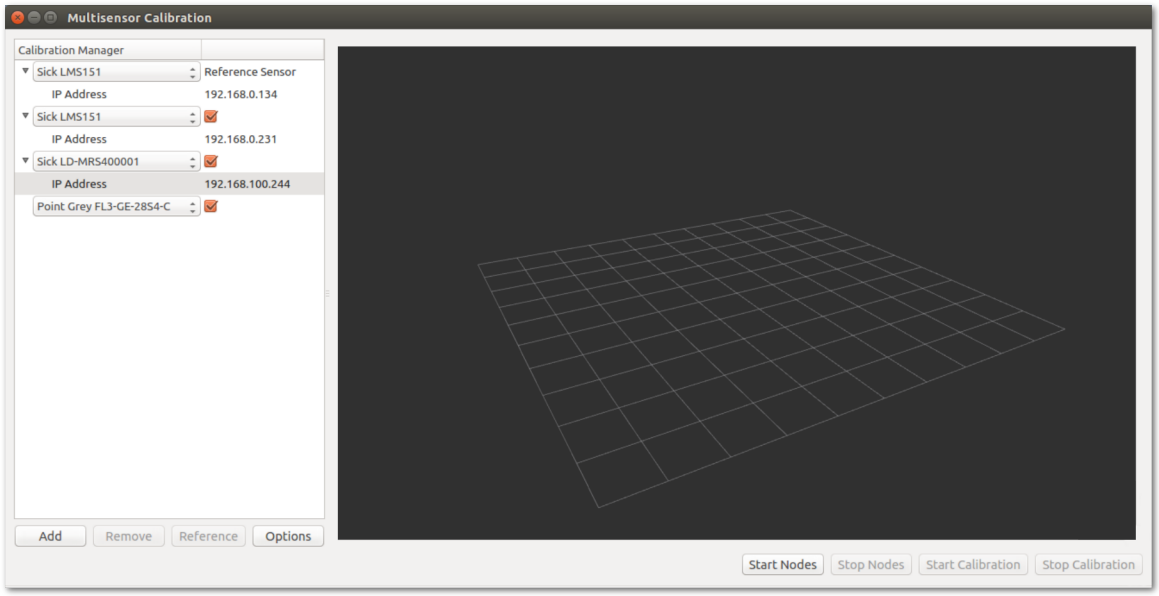


Figure 3.7: Calibration GUI

Multi Target Tracking (MTT)

The MTT library is a set of methods and strategies developed by Soares De Almeida 2016 specially designed for the ATLASCAR project with the goal to send perception parameters about objects captured in the LIDARs.

The MTT is capable of receiving LIDAR messages and break them in smaller groups. This process is called clustering. Clustering separates all objects found in a scan and returns their position. The clustering of data is an important step in the development of this dissertation as it facilitates the detection and tracking of objects in space.

The clustering algorithm of the MTT library is based on the Nearest Neighbor Clustering Algorithm (Luo, Habibi and Mohrenschildt 2016). It segments the received pointclouds using a predefined distance as threshold. Each set of points in the clusters are assumed as possible targets of interest.

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 be occluded during the sequence (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 allowing objects to be followed while they are out of the field of view but still in the surroundings.

The MTT defines targets that keep information about the object and its velocity plus their position and obstacle lines.

3.4.3 PCL

The PCL is a library that implements methods to process pointcloud data. The PCL framework contains numerous state of the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation (Point Cloud

Library 2018). Some of these algorithms are used throughout this thesis an were also used by the MTT library.

Chapter 4

Improvement of the Calibration Process in ATLASCAR2

Using sensors to recognize the environment is of most importance for vehicles to become fully autonomous. 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.

One of the main tasks for this dissertation is to be improve the extrinsic camera calibration process. Multi sensor calibration in ATLASCAR 2 is done using a ball as a target. One of the limitations of the approach in Vieira da Silva 2016 is the use of filtering the image using HSV values to detect the ball.

While moving the ball around the sensors, a point cloud of ball centers 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.

4.1 New Ball Detector Algorithm

This section describes the progress introduced in the development of the ball detector.

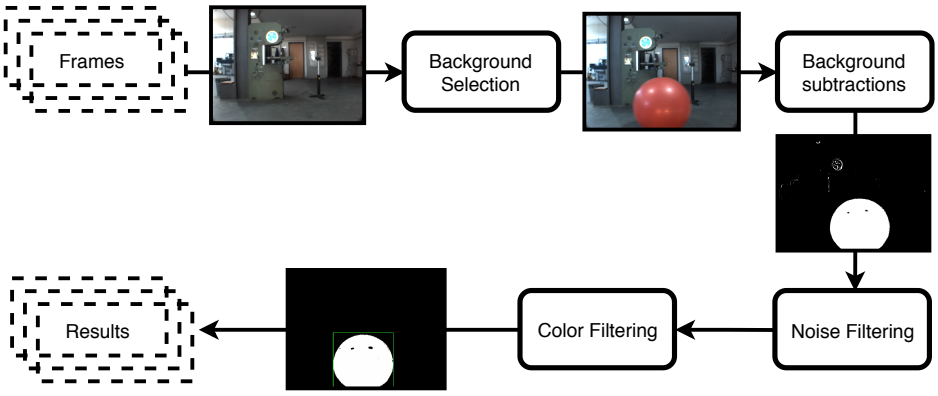


Figure 4.1: New ball detection algorithm simplified diagram

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 shown in figure 4.2.

Figure 4.2: Ball used for calibration testing

To acquire the images, the ball detector 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. The vehicle is supposed to be still when the calibration process is being done, so the background subtraction is a good method to apply.

Figure 4.3: Background in test rosbag used for testing

When the capture starts, the first received frame will be the default background (see figure 4.3). 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 (figure 4.4), by subtracting the background with the actual frame it is possible to detect easily where the object is (OpenCV 2018a).

Figure 4.4: Frame with ball rolling in front of the camera

In simple words, background subtraction extracts the foreground from the static background. Most times, the background is modified by applying Gaussian blur to it. When subtracting the foreground with the background, a value is obtained for each pixel.

In most cases the pixel values will not get to zero because of minor changes to the background (caused by shadows) which need to be ignored. In the resulting image a threshold is applied. 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. The result is a binary image as seen in figure 4.5.

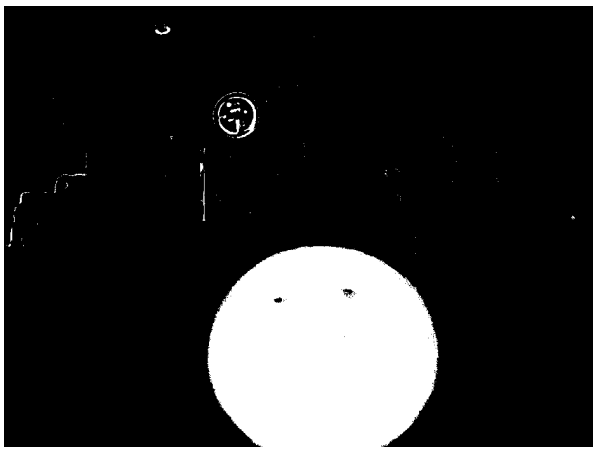


Figure 4.5: Background subtraction result with noise

4.1.2 Noise Filtering

After the background removal technique is applied, the resulting image may possibly contain some noise. Some pixels may be left white in the background due to lightning changes when the ball passes in front of the camera. These white pixels are noise and must be removed.

An erosion algorithm is implemented where for each pixel 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 (with value of 1).

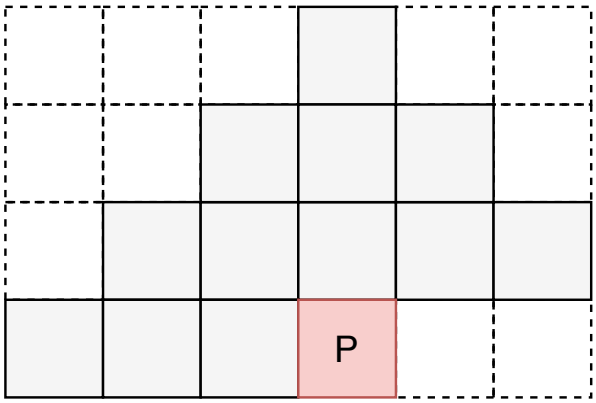


Figure 4.6: Neighbor pixels used in the algorithm where grey pixels represent the kernel and the pixel P represents the pixel being processed

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 (OpenCV 2.4.13.6 documentation 2018). The result is the image in figure 4.7.

4.1.3 Color Filtering

After the noise filtering, the ball stands out in the image. The next step is to separate the ball using color filtering. This step is necessary to make the detection usable when a person appears in the image to displace the ball. This way, the person will not appear in the resulting image.

The selected color by default is red. 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.

4.1.4 Bounding Box

A bounding box is drawn around the ball if there is a number of white pixels bigger than a predefined threshold. 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 in the image. 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 greater weight. Hence, the width and height of the bounding box are given by the maximum distance between the pixels with higher weight.

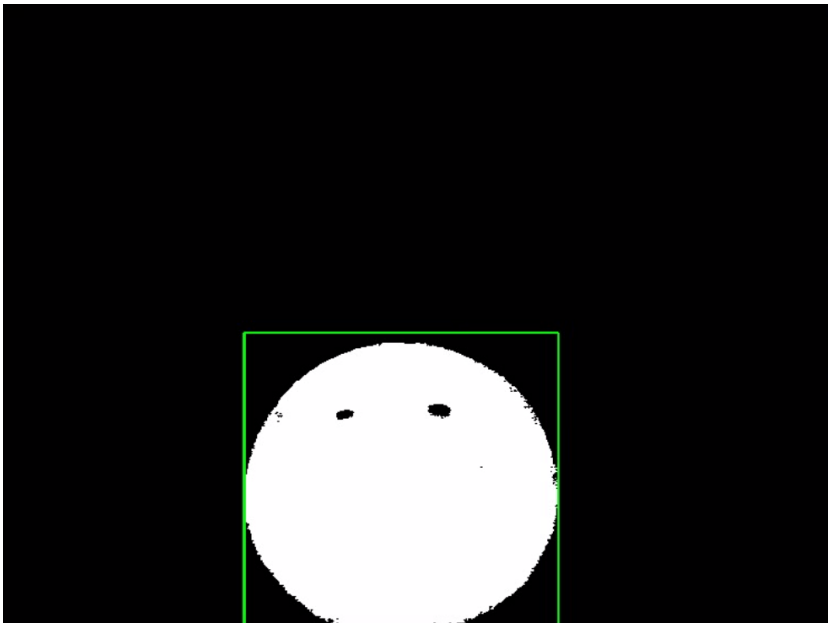


Figure 4.7: Ball detected with bounding box

4.2 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 in the previous 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`.

The 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.

Chapter 5

Object Detection, Tracking and Labelling

This chapter explains how the object detection, tracking and labelling application was implemented. First, it will be described how this is performed in the 2D image.

Secondly, the implementation of the tracking of targets using LIDAR sensors will be explained. Lastly, the combination of the 2D image and the LIDAR data is explained.

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 sensors.

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.

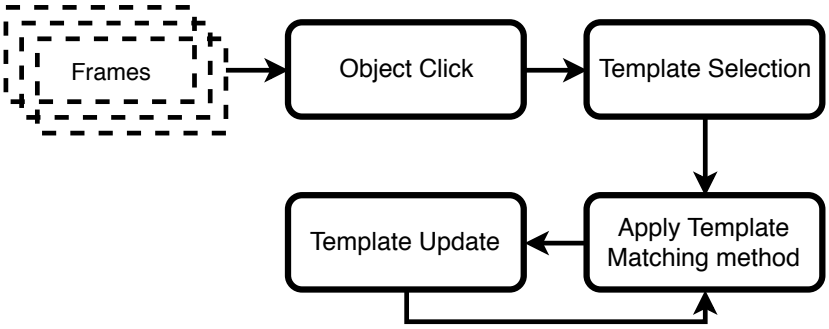


Figure 5.1: Image tracking algorithm diagram

Figure 5.1 presents an overview of the image tracking algorithm. It starts by receiving frames and by selecting a target on the images. Finally, template matching is applied to follow the object.

To obtain the image frames, this node subscribes the camera images through its rostopic. 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 5.2) 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.

Figure 5.2: Window view with image sequences appearing

When the user clicks on an object, a callback function is triggered to process the mouse event and a bounding box with a predefined size appears in the click position.

During this process, when the image is iterated the upper third-part of the image is ignored 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.

The bounding box follows the selected target. 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). A resulting image is calculated by iterating the source image and comparing the template with the area in that position (see figure 5.3). For each position

a score is assigned representing how good the match is in that position.

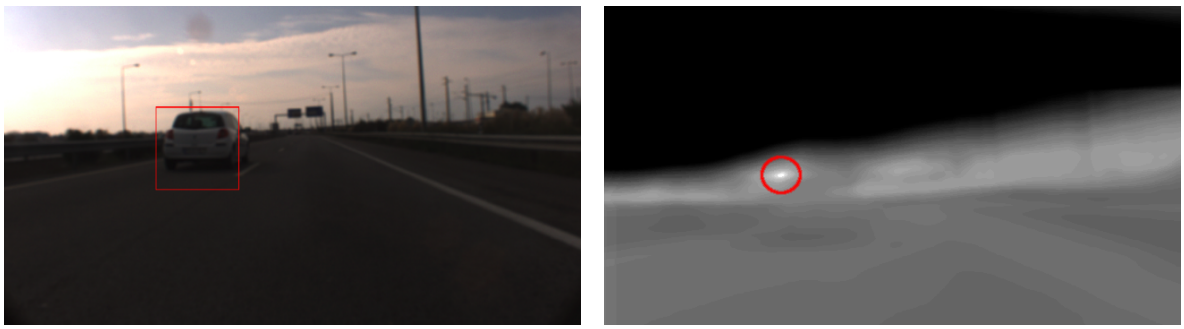


Figure 5.3: Template matching example result matrix

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.

This step concludes the front tracking. It is important to notice that this application features tracking of the objects not only for the next frames but also for the previous. The tracking for the next and previous frames is respectively the front and back tracking.

After the whole tracking is done, the back tracking is now performed. The reason why the back tracking is done after the front tracking is mainly because of processing times. If the back tracking began 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. In figure 5.4 an example of the tracking is presented.

5.2 Range Based Tracking

To improve the tracking, the image process is combined using the LIDAR scanners 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.

In figure 5.5 an overview of the range based detector algorithm is presented. It starts by receiving data from the LIDAR sensors and converting it to a pointcloud object. Finally, the detection is done by clustering the pointcloud where each cluster will represent a detected object. The clustering is performed with the MTT library developed by Soares De Almeida 2016.

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 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.

Figure 5.4: Example of back tracking and front tracking: the picture in the middle is the selected frame, the two upper frames show the back tracking and the two lower frames show to front tracking.

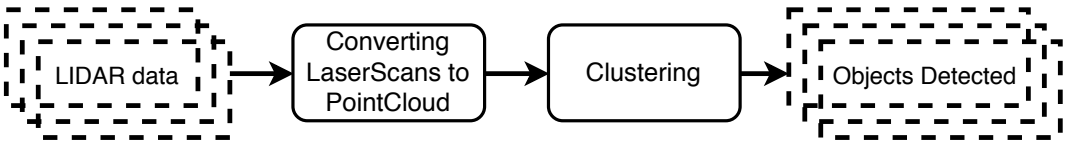


Figure 5.5: Range Based Detector Algorithm Diagram

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.

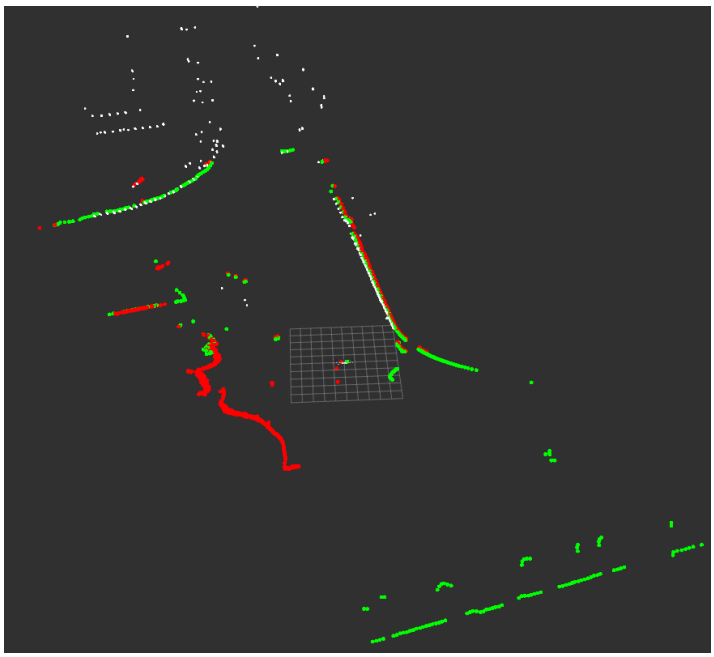


Figure 5.6: All the separated laserScans visualized with Rviz

In figure 5.6 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 concatenate pointclouds making it easy to merge them all together.

Finally, the pointcloud is filtered to a squared area in front of the car in order to avoid unwanted objects to appear such as roadsides on the highway. The pointclouds are converted from the ROS format to PCL format, concatenated, and ready to be processed by the MTT library.

The next step is to cluster the different objects found in the pointcloud. The clustering strategy is implemented by the MTT library and it is based on the Nearest Neighbor Clustering algorithm (Yu et al. 2018). 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. If the distance between points is larger than a given threshold, then a cluster is formed and an object is most likely to be there.

To make it easier to visualize, markers are created and placed in the location of 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.

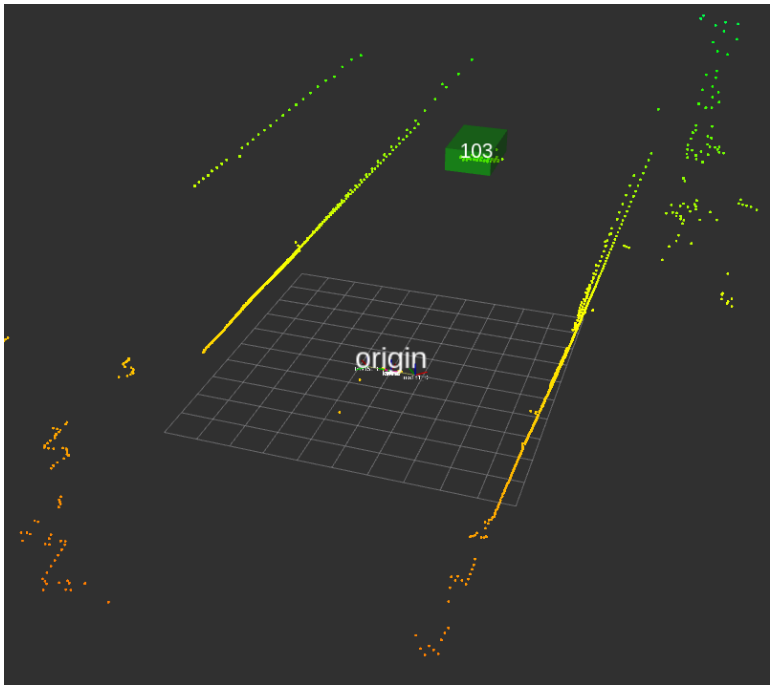


Figure 5.7: Visualization of a detected car with MTT in Rviz

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 with a predefined static size are created in the location of the objects.

The information in figure 5.7 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). The processed pointcloud can be seen, where the LIDAR `laserScans` messages are all merged and a green 3D bounding box with the ID 103 is shown meaning that an object was found at that location. The object was in fact a car traveling in front of the ATLASCAR 2. The roadsides are not detected since the pointcloud was filtered to a squared area in front of the car in order to avoid objects with no interest. Only part of the pointcloud in figure 5.7 is processed by the MTT.

5.3 Sensor Data Fusion

To accomplish sensor data fusion, a multi-modal approach was utilized, combining data retrieved from several ranged based and visual sensors.

Since the detection, tracking and labelling can be done independently with the 2D image and with the LIDAR data, this section will explain how the combination of both can be done.

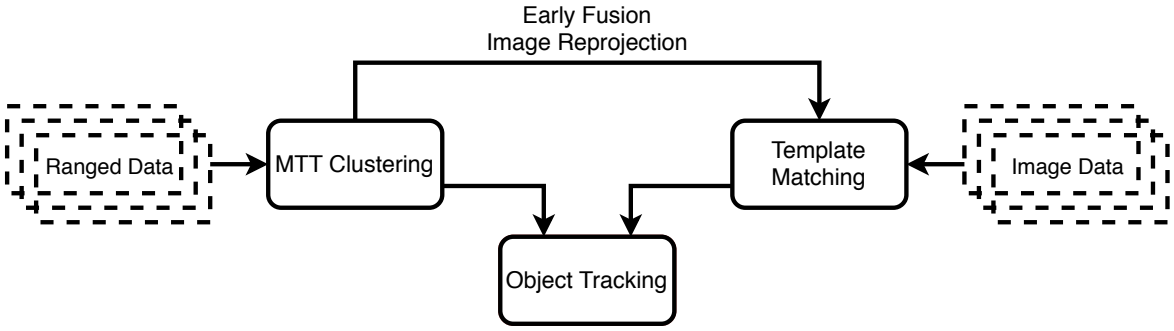


Figure 5.8: Overview of the multi-modal approach.

Figure 5.8 depicts the process of the multi-modal approach. In sensor fusion, ranged based information is combined with the image data allowing augmented perception of the surroundings. With the ability to have basic cognition of the environment, it is possible to detect and track objects in motion (Spinello, Triebel and Siegwart 2010).

5.3.1 Dynamic 2D Bounding Box Size

Previously, the bounding box size was static but using the 3D position given by the MTT it is possible to implement dynamic bounding box size.

The 2D 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 algorithm.

Figure 5.9: Example of near (left) and far (right) car with dynamic bounding box size

Figure 5.9 depicts an example of the dynamic bounding box as the target moves away.

5.3.2 Pointcloud Projection

By combining the sensor data with the image it is possible to check where the pointcloud is relatively to the camera position. To accomplish this it is needed to firstly make the

```
image_width: 1624
image_height: 1224
camera_name: 0
camera_matrix:
  rows: 3
  cols: 3
  data: [1454.423376687359, 0, 822.9545738617143, 0,
1458.005828758985, 590.5652711935882, 0, 0, 1]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.2015966527847064, 0.1516937421259596,
-0.0009340794635090795, -0.0006787308984611241, 0]
rectification_matrix:
  rows: 3
  cols: 3
  data: [1, 0, 0, 0, 1, 0, 0, 0, 1]
projection_matrix:
  rows: 3
  cols: 4
  data: [1379.264282226562, 0, 822.6802277325623, 0, 0,
1410.231689453125, 588.4764252277164, 0, 0, 0, 1, 0]
```

Listing 5.1: Intrinsic Calibration Result

intrinsic calibration of the camera, and then proceed to the implementation of the pointcloud projection. By using a ROS node in the package `camera_calibration`, the intrinsic values of the camera and the distortion coefficients are obtained (see listing 5.1).

In this file it is possible to see the image dimensions, the intrinsic values and distortions coefficients of the camera as well as the rectification and projection matrix. These values are used to reproject the points of the pointcloud into the camera's image. To do so, OpenCV implements a method called `projectPoints` (OpenCV 2018b) that is used in the labelling node.

Firstly, the labelling node reads the file in listing 5.1 to calibrate the camera. Then, the full pointcloud is taken and a vector of the points is retrieved. The `projectPoints` method takes the calibration file parameters and the points vector and generates another vector with the (x, y) coordinates of the points in the image.

The next step is simply to draw those points in the image using the function `circle` implemented by OpenCV. The final result is as seen on the left in figure 5.10.

With the pointclouds points projected in the image it is possible to associate the 3D clusters to the 2D image templates.

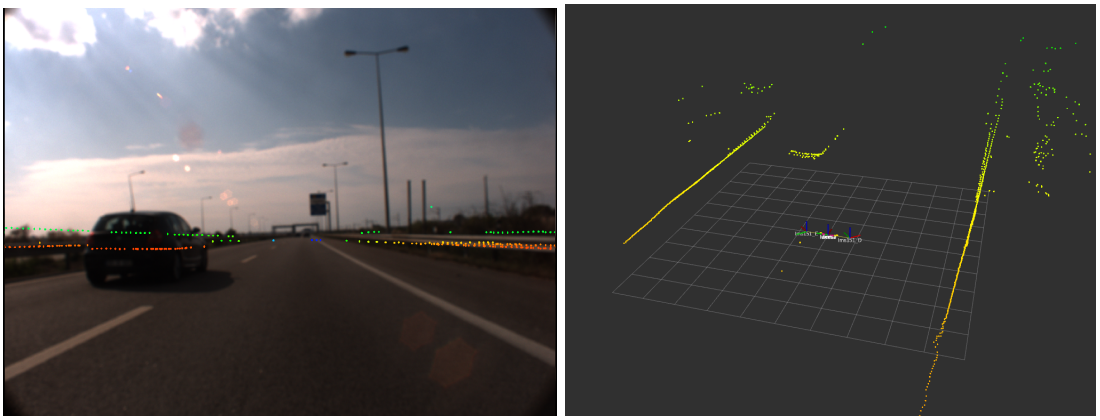


Figure 5.10: Example of the pointcloud projection (left) in comparison with the Rviz 3D view (right)

5.3.3 Suggestion of Objects of Interest

Using the algorithms previously developed it is possible to track objects in the 2D image using the position of the 3D clusters by picking up their points and following them.

A semi-automatic algorithm was developed where the tracking can be done manually by clicking while the semi-automatic system suggests objects of interest that may appear in the field of view. A simplified diagram of the algorithm is found in figure 5.11.

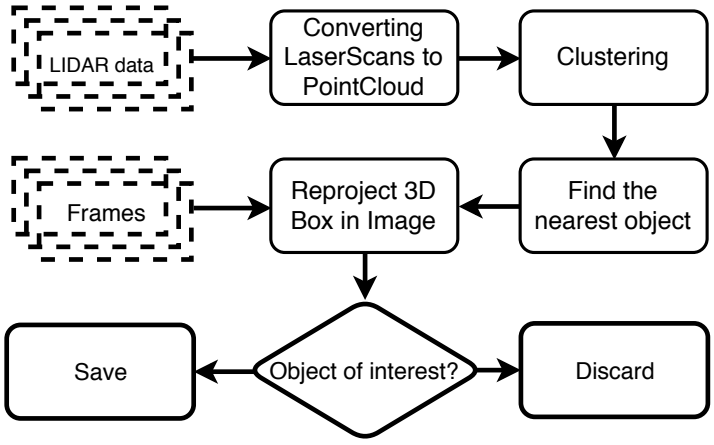


Figure 5.11: Simplified diagram of the tracking with sensor fusion

To begin, the full pointcloud data is gathered and passed into the MTT algorithm. The MTT library implements methods that handle detection and tracking of objects and outputs the coordinates of the found objects.

A function polls the MTT algorithm and raises a flag when an object is found. If more than an object is found, the nearest object is tracked. The coordinates of the objects relatively to the vehicle are found so the distances can be calculated. The object with shortest distance is chosen to track.

Having the coordinates of the tracked object, a cube with a fixed size is drawn centered in the point given by the MTT algorithm by setting the cube vertices and using the `line`

function implemented by OpenCV to join them.

Then, the cube is projected to the image using OpenCV's `projectPoints`. Figure 5.12 shows a 3D bounding box being projected around the detected car.

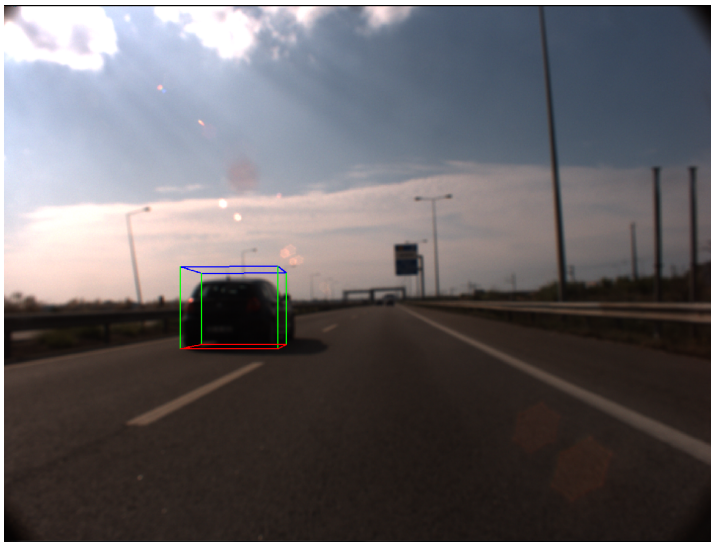


Figure 5.12: Example of 3D bounding box tracking a car

5.3.4 Improvement of the Manual Labelling

When the manual mode is active, the user selects the target to follow by clicking the image. In that location, the labelling node checks for an object found by the MTT algorithm.

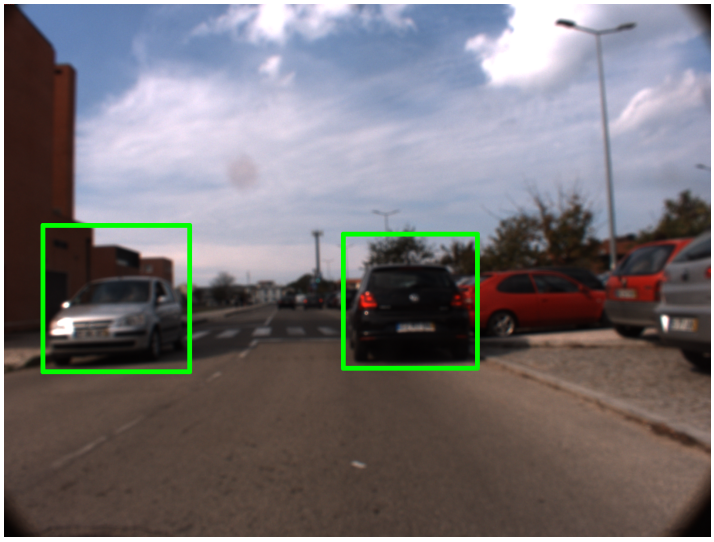


Figure 5.13: MTT exposing all found targets in the image

All targets found by the MTT are exposed in the image (see figure 5.13) and if the click is inside of one of the target's areas, the tracking is done using the MTT. Otherwise, the tracking is done using template matching.

```
struct BBox
{
  int x;
  int y;
  int width;
  int height;
  int id;
  string label;
};
```

Listing 5.2: BBox struct definition used for 2D datasets.

```
FRAME_ID
BOX_X BOX_Y WIDTH HEIGHT LABEL ID
...
1083
815 663 155 104 car 4
1084
816 662 155 104 car 4
1142
482 584 152 150 van 5
1143
512 589 152 150 van 5
...
```

Listing 5.3: 2D dataset example snippet

5.4 Output Datasets

5.4.1 2D Dataset

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 5.2. 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 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

```
struct BBox
{
  int x;
  int y;
  int width;
  int height;
  int id;
  string label;
  // 3D position
  double x3d, y3d, z3d;
};
```

Listing 5.4: BBox struct definition with 3D capabilities

```
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 5.5: Snippet of the dataset with 3D capabilities

and printed to a file similar to the snippet in listing 5.3.

5.4.2 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` (see listing 5.4) in which the 3D coordinates of the objects have been added.

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 5.5) 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 5.5, 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 forced to 0.5 (half a meter).

5.5 UI Tools for Labelling

In this section it will be described some extra tools and features implemented in addition to the detection, tracking and labelling of objects. Most features presented in this section are used to aid and complement what has been previously done.

5.5.1 Labelling Interface

The labelling node features a simplified GUI where the user can do several operations.

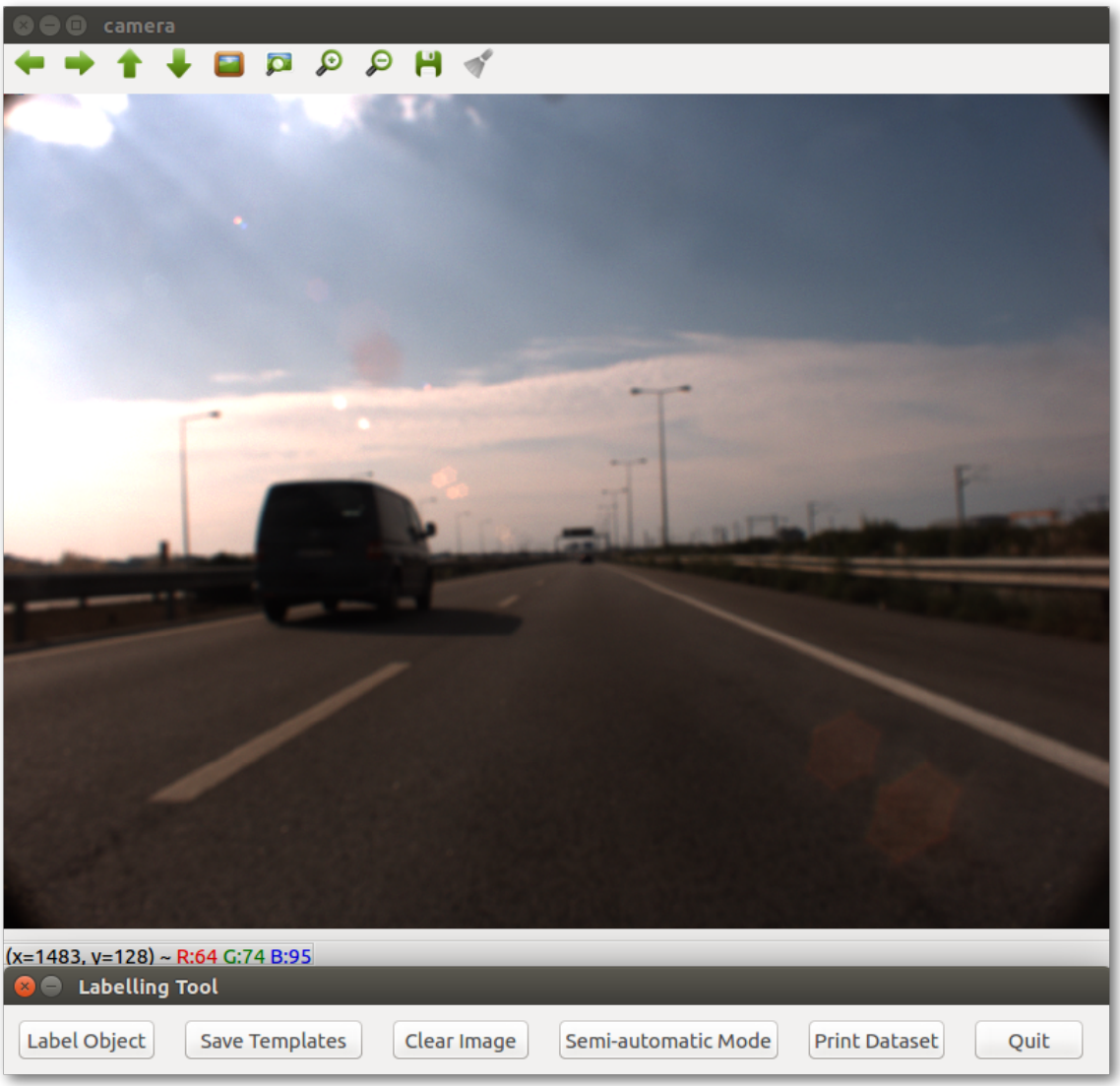


Figure 5.14: Labelling Node GUI

The GUI features an interactive window where the image sequences appear and the user

may select targets. There is also a window with six buttons where multiple actions can be performed.

- **Label Object** - Pauses the rosbag and asks the user for a label of the target. Ends the tracking.
- **Save Templates** - Pauses the rosbag and asks the user for a label of the target. Saves the image templates in all frames where the target appeared.
- **Clear Image** - End the tracking and clears the bounding box from the image.
- **Semi-Automatic/Manual Mode** - Switches between manual mode and semi automatic mode.
- **Print Dataset** - Saves the dataset created during the whole labelling process.
- **Quit** - Exits the program.

After successfully completing a full tracking, the user will be prompted with a window to enter a label for the target. The user may also discard the target instead of saving it.

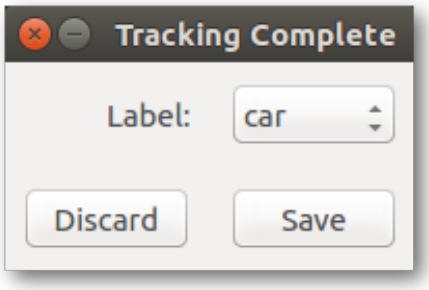


Figure 5.15: Prompt window to select a label and save or discard.

5.5.2 Automatic Pause for the rqt_bag

A problem encountered during the labelling process was that the execution of the ROS bag file continued when the tracking was complete. The user needed to insert a label to the followed target and after the object is labelled the ROS bag continued creating a gap (see figure 5.16).

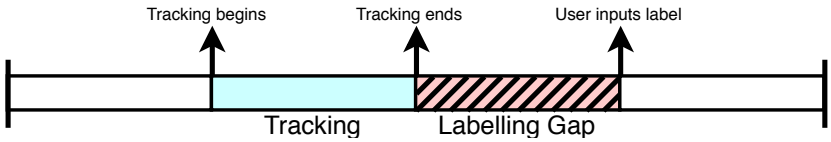


Figure 5.16: A gap is created while the user is prompted to input a label because the ROS bag execution does not pause

To solve this problem, the rqt_bag package was adapted. A ROS service was implemented in the rqt_bag node to receive messages from the outside. The labelling node sends a request

to pause the rqt_bag when the tracking ends. After the user inserts a label the labelling node sends a message to the rqt_bag node to resume the bag execution.

5.5.3 Playback

An extra rosnode was developed with the aim to play the rosbag and identify the objects in the images using the dataset created previously while the rosbag plays in the background.

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 5.17. The image is then converted again into the ROS format and published to a topic.

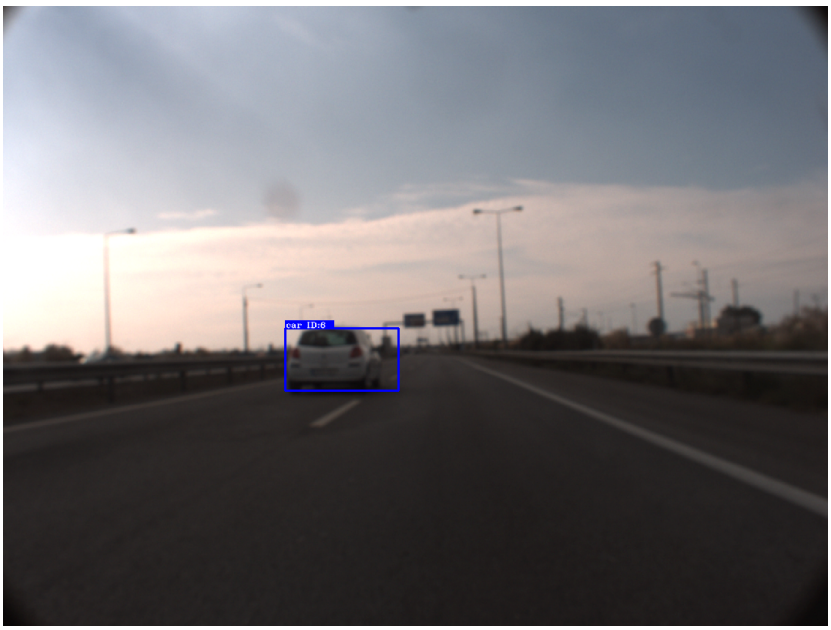


Figure 5.17: Playback example with a car

Chapter 6

Results

This chapter presents the results of the improvement of the calibration process and object detection, tracking and labelling system.

6.1 Camera Calibration

Figure 6.1 presents the multisensor calibration GUI. One of the SICK LMS151 LIDARs was used as point of reference for the purpose of demonstration and to have a comparator point to the camera.

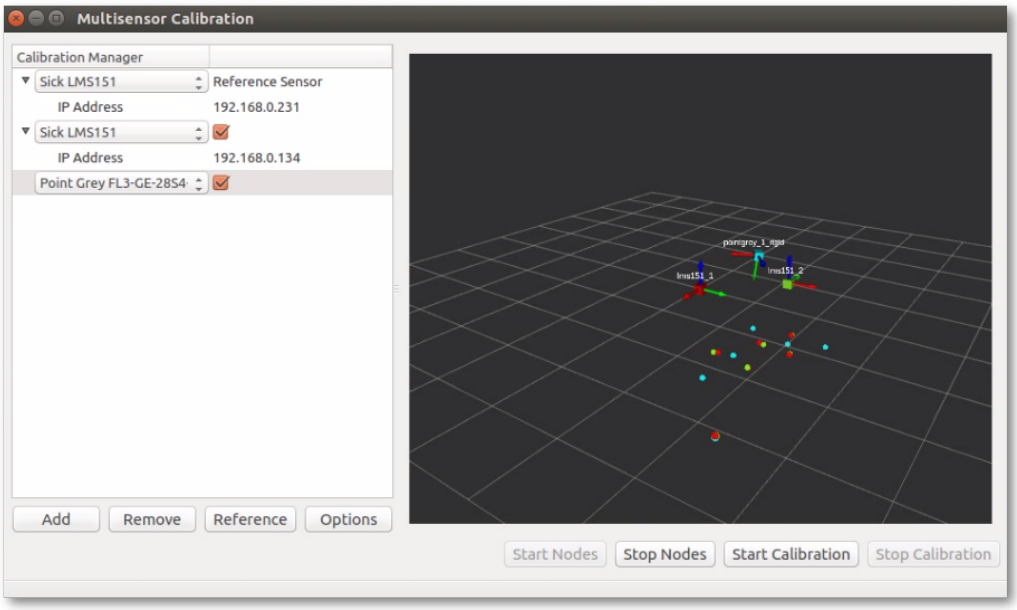


Figure 6.1: Calibration GUI with calibration result

During the calibration procedure, the ball rolls in front of the camera and sensors. The range 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.

```
-0.0563334   -0.998402 0.00440481    -1.07636
0.997797 -0.0561436   0.0353294     1.08224
-0.0350256 0.00638533    0.999366   0.0617893
0           0           0           1
```

Listing 6.1: Calibration output file.

In listing 6.1 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 its transformation matrix relatively to the reference sensor. The reference sensor does not create a transform because it is assumed that this sensor is in the origin, unrotated. This file is stored in a folder with the calibration results inside the calibration package directory.

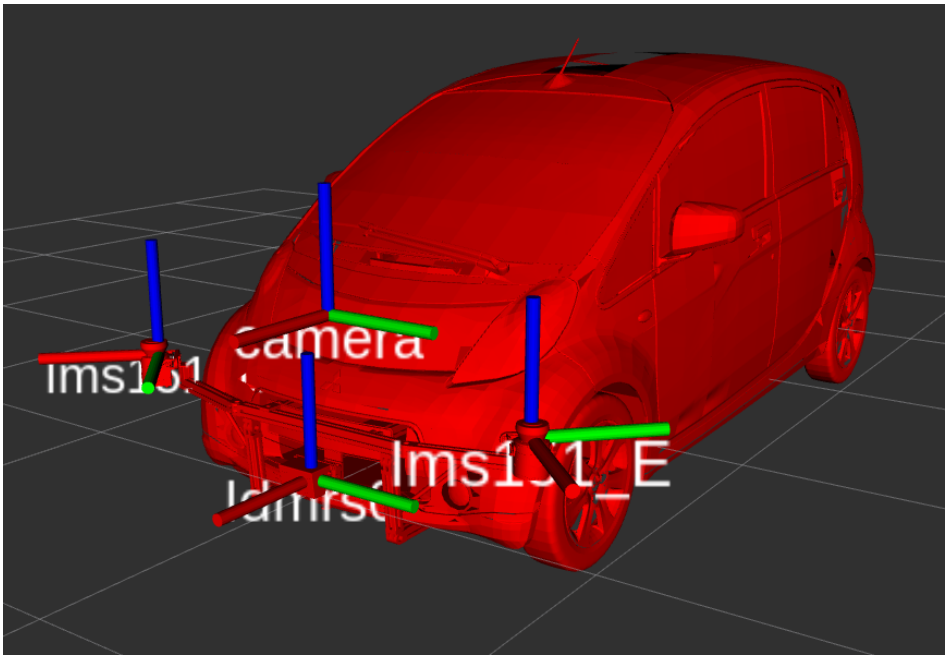


Figure 6.2: ATLASCAR2 model with sensors transformation frames

Figure 6.2 presents an ATLASCAR2 3D model with the several sensors and their respective transformation frames after applying the output file to a frame publisher.

6.2 Detection, Tracking and Labelling

This section will present the detection, tracking and labelling results with two rosbags recorded and used for the testing of the detection, tracking and labelling algorithms. Two rosbags were recorded in October 17, 2017, in the afternoon:

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

  - There are mostly cars in this rosbag and it was better to start with some tests in tracking objects.

- The second rosbag was recorded while leaving Universidade de Aveiro. The car traveled around the campus and visited the Alboi neighborhood.

  - In this bag there are cars, vans, cyclists 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.

Each rosbag produced a dataset where the objects found were registered. Looking into the datasets can show how the algorithm behaves. Several objects were found in the sequences. The discarded objects or the objects wrongly suggested by the semi-automatic methods are still saved but labelled as `DontCare`. The main labels used are `car`, `van`, `people`, `bicycle`, `sign` (street signs) and `misc` (miscellaneous objects in the streets like dumpsters, bushes and trees).

6.2.1 Dataset 1 - Highway / A25

The first dataset was produced from the highway rosbag. The rosbag duration is 3:37s (217s), has a size of 8.6 GB and includes 62352 messages.

The labelling was done with semi-automatic methods using sensor fusion and also manually by pointing and clicking on an object when the sensors don't find it. The semi-automatic methods suggest the user an object of interest and asks for a label from the user.

Table 6.1: Highway dataset annotation metrics

Metric	**Count**
Right Suggestions	12
Wrong Suggestions	6
Manual Entries Saved	1
Manual Entries Discarded	0
Maximum Objects in One Frame	1

Table 6.2: Highway dataset object amounts

Label	`car`	`van`	`people`	`bicycle`	`sign`	`misc`	`DontCare`	**total**
Count	7	1	0	0	4	1	6	19

In table 6.1 some statistics about the dataset are presented. Only one manual entry (figure 6.4) was saved because the target was too far and the semi-automatic methods in that place suggested the roadsides. As this sequence presents few objects at a time, most semi-automatic suggestions were accepted. The semi-automatic system works better in areas with reduced movement so the readings of the sensors are more accurate.

Figure 6.3 shows four examples in which three were right suggestions (`car`, `sign` and `misc`) and one was wrong (`DontCare`). Table 6.2 shows how many objects of each category appear in the sequence.

Figure 6.3: Targets acquired using the MTT labelled as `car` (top-left), `sign` (top-right), `misc` (bottom-left) and `DontCare` (bottom-right)

Figure 6.4: Example of `car` with manual entry.

6.2.2 Dataset 2 - Aveiro City Urban Area

The second dataset was generated from the rosbag at an urban area in Aveiro city. The rosbag duration is 7:06s (456s), has a size of 18.0 GB and includes 135379 messages.

The labelling in this rosbag was done using the semi-automatic methods and the manual methods separately. The reason for this is to compare the behavior of the semi-automatic suggestions and the manual method in a different environment.

Dataset 2.1 - Semi-Automatic Method

The dataset produced with the semi-automatic method the labelling metric listed in table 6.3.

Table 6.3: Urban area dataset with semi-automatic methods annotation metrics

Metric	Count
Right Suggestions	36
Wrong Suggestions	74
Maximum Objects In One Frame	1

The Urban area is a zone with high-traffic and because there are more objects (sidewalks, roadsides, walls, etc...), the semi-automatic method is more likely to mistake an object of interest with something with no significance in the dataset scope.

The metrics show that nearly 2 out of 3 suggestions are something that has no interest for the dataset. In table 6.4 the amount of objects found is presented and figure 6.5 shows

five examples of suggestions given by the tracking system where four are correct (`car`, `van`, `people` and `sign`). One of the example is a wrong suggestion (`DontCare`).

Table 6.4: Urban area dataset with semi-automatic methods object amounts

Label	`car`	`van`	`people`	`bicycle`	`sign`	`misc`	`DontCare`	**total**
Count	29	2	4	0	1	0	74	110

Dataset 2.2 - Manual Method

To test the manual method, one must click on the object and control the rosbag (pause, resume, go back and forward, etc...) to obtain maximum precision on the detection and tracking.

Table 6.5: Urban area dataset with manual methods annotation metrics

Metric	**Count**
Entries Saved	89
Entries Discarded	31
Maximum Objects In One Frame	4

In table 6.5 some annotation metrics are shown. Despite the high traffic zone, the metrics in the manual methods show that nearly 3 out of 4 clicks result in a good detection and tracking of the object.

Because the annotation was done manually, it is possible to enter more than one object at the same time whereas in the semi-automatic that is not possible because when the sensors suggest an object in sight, the algorithm always outputs the object that is closer to the vehicle. In table 6.6 it is possible to see the amount of each object in this dataset.

Table 6.6: Urban area dataset with manual methods object amounts

Label	`car`	`van`	`people`	`bicycle`	`sign`	`misc`	`DontCare`	**total**
Count	63	10	10	2	1	3	31	120

By analyzing table 6.6 it is possible to conclude that with manual methods one can freely enter with more detail the detected objects. In comparison to the semi-automatic methods, with the manual strategy it is easier to enter objects like people and bicycles as it seems to be difficult to cluster such small objects in the given sensor data.

Figure 6.6 shows six examples of manual entries (`car`, `van`, `people`, `bicycle`, `sign` and `misc`).

Figure 6.5: Example of suggestions given by the semi-automatic methods in dataset 2: `car` (top-left), `van` (top-right), `people` (middle-left), `sign` (middle-right) and `DontCare` (bottom)

Figure 6.6: Example of templates with the manual methods in dataset 2: `car` (top-left), `van` (top-right), `people` (middle-left), `bicycle` (middle-right), `sign` (bottom-left) and `misc` (bottom-right)

Chapter 7

Conclusions and Future Work

This chapter concludes the work done during this dissertation regarding the creating of a ROS package, improvement of the ball detection node and the labelling node. Having this conclusions as a foundation, a few related future works will be proposed in the scope of the ATLASCAR2 project and the AD and ADAS paradigm.

7.1 Conclusions

AD and ADAS fields are a research area prioritized by many car manufacturers. Building perception of the environment surrounding a vehicle is the solution to create anti collision systems, path planning systems, among other applications. Platforms are usually equipped with several sensors to gather information from the environment. With a fully equipped vehicle the data can be manipulated and several applications can be built. Before, any use of the sensors data can be done, an extrinsic calibration is necessary to obtain the several devices relative positions.

To improve the ball detection in the calibration process the major improvement made was based on the background removal technique. Erosion processes were made to filter noise and color filtering was done by selecting the color of the ball. In addiction to the previous work done on the calibration package, the ball detection was improved and the GUI was simplified. The old GUI featured sliders to choose the HSV values of the ball whereas now those values can be obtain by simply clicking in the ball on the image.

With the sensors fully calibrated the ATLASCAR2 is ready to gather data. Labelling of objects is an important subject the AD and ADAS since it is linked with the recognition of objects. It is important for an autonomous vehicle to know what object is in front of them and act according to them. To build the labelling system, it was first implemented ways to detect and track objects using the image and sensor data. Tracking using the visual information was done with template matching where an object is selected by clicking on the target and the tracking is made by iteratively updating the patch used in the matching method. Since the camera is mounted in a vehicle in constant movement, some methods such as optical flow were not appropriate and the template matching strategy was the one that produced better results. The range based tracking was accomplished thanks to the MTT library which implemented algorithms that used the pointcloud generated by the several sensors and applied data clustering to separate the found objects. The spacial coordinates of the found objects are outputted by the MTT algorithm and the 3D tracking is realized. To obtain maximum

accuracy with the tracking the sensor data and the images are merged. The data fusion is used to create detection and tracking applications and the system may also become semi-automatic with the aid of both information from the LIDARs and the camera. While tracking objects, it is important to classify them and this is why it is assigned to each object a label. In the end, datasets are created with a fully annotated sequence showing where objects are found and to what class they belong.

7.2 Future Work

The possibilities of future work on ATLASCAR2 regarding AD and ADAS are many. Some interesting and worth mentioning projects would be the development of a more automatic detection and tracking system used for labelling in order to create more robust suggestions and datasets as an extension to the work on this dissertation.

The Machine Learning is a essential study field for the AD and ADAS projects. So, the image templates and datasets produced by this dissertation can serve as input for a learning algorithm that may implement a full detection, tracking and recognition system for the ATLASCAR2.

The scope of this dissertation was based on objects such as cars, people and some miscellaneous objects on the streets. To expand this, a street sign detection system would be important for the ATLASCAR2 to become fully autonomous.

Using augmented reality glassed would also be an interesting project to develop a detection, tracking and labelling system.

References

Andreas Herrmann, Walter Brenner, Rupert Stadler (2018). *Autonomous Driving: How the Driverless Revolution will Change the World*. First edit. Emerald Publisher Limited. URL: https://books.google.pt/books/about/Autonomous%7B%5C_%7DDriving.html?id=JfxQDwAAQBAJ.

Biliotti, David, Gianluca Antonini and Jean Philippe Thiran (2015). "Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video sequences". In: URL: https://infoscience.epfl.ch/record/87317/files/Biliotti2005%7B%5C_%7D1412.pdf.

Correia, José (2017). "Unidade de Perceção Visual e de profundidade para o ATLASCAR2". In: pp. 1–98. URL: http://ria.ua.pt/handle/10773/22498.

Ess, A. et al. (2009). "Moving obstacle detection in highly dynamic scenes". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 56–63. ISBN: 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152884. URL: http://ieeexplore.ieee.org/document/5152884/.

Geiger, Andreas et al. (2013). "Vision meets Robotics: The KITTI Dataset". In: URL: http://ww.cvlibs.net/publications/Geiger2013IJRR.pdf.

Kröger, Fabian (2016). "Automated Driving in Its Social, Historical and Cultural Contexts". In: *Autonomous Driving*, pp. 41–68. DOI: 10.1007/978-3-662-48847-8_3. URL: http://link.springer.com/10.1007/978-3-662-48847-8%7B%5C_%7D3.

Luo, Zhongzhen, Saeid Habibi and Martin V Mohrenschildt (2016). "LiDAR Based Real Time Multiple Vehicle Detection and Tracking". In: URL: https://waset.org/publications/10004678/lidar-based-real-time-multiple-vehicle-detection-and-tracking.

Montemerlo, Michael et al. (2006). "Winning the DARPA Grand Challenge with an AI Robot". In: *Artificial Intelligence* 21.Gat 1998, pp. 982–987. ISSN: 03029743. URL: http://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:Winning+the+DARPA+Grand+Challenge+with+an+AI+Robot%7B%5C#%7D0.

Pomerleau, Dean A (1989). "An Autonomous Land Vehicle In a Neural Network". In: URL: http://repository.cmu.edu/cgi/viewcontent.cgi?article=2874%7B%5C&%7Dcontext=compsci.

Sigal, Leonid et al. (2009). "HUMANEVA: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion". In: *Int J Comput Vis*. DOI: 10.1007/s11263-009-0273-6. URL: http://files.is.tue.mpg.de/black/papers/ehumIJCV10web.pdf.

Soares De Almeida, Jorge Manuel (2016). "Seguimento ativo de agentes dinâmicos multivariados usando informação vectorial Active Tracking of Dynamic Multivariate Agents using Vectorial Range Data". In: URL: http://ria.ua.pt/handle/10773/16988.

Spinello, Luciano, Rudolph Triebel and Roland Siegwart (2010). "Multiclass Multimodal Detection and Tracking in Urban Environments". In: pp. 125–135. DOI: 10.1007/978-3-642-13408-1_12. URL: http://link.springer.com/10.1007/978-3-642-13408-1%7B%5C_%7D12.

Teo, Pauline (2018). "SMART launches first Singapore-developed driverless car designed for operations on public roads". In: URL: https://smart.mit.edu/images/pdf/news/2014/Driverless%7B%5C_%7DCar%7B%5C_%7DNR%7B%5C_%7D270114%7B%5C_%7DFinal.pdf.

The Milwaukee Sentinel (1926). "8 Dec 1926 - " 'Phantom Auto' will tour city"". In: URL: https://news.google.com/newspapers?id=unBQAAAAIBAJ%7B%5C&%7Dsjid=QQ8EAAAAIBAJ%7B%5C&%7Dpg=7304,3766749.

Thrun, Sebastian et al. (2007). "Stanley: The Robot That Won the DARPA Grand Challenge". In: pp. 1–43. DOI: 10.1007/978-3-540-73429-1_1. URL: http://link.springer.com/10.1007/978-3-540-73429-1%7B%5C_%7D1.

Vieira da Silva, David Tiago (2016). "Multisensor Calibration and Data Fusion Using LIDAR and Vision". In: p. 107. URL: http://ria.ua.pt/handle/10773/17967.

Yu, Fisher et al. (2018). "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling". In: arXiv: 1805.04687. URL: http://arxiv.org/abs/1805.04687.

On-line References

Audi MediaCenter (2018). *Audi at the IAA 2017: Autonomous driving in three steps*. URL: https://www.audi-mediacenter.com/en/press-releases/audi-at-the-iaa-2017-autonomous-driving-in-three-steps-9311 (visited on 17/04/2018).

Boston Didi Team (2018). *KITTI Github Respository*. URL: https://github.com/bostondiditeam/kitti.

DARPA (2018). *DARPA Grand Challenge*. URL: http://archive.darpa.mil/grandchallenge/ (visited on 29/05/2018).

EPFL (École polytechnique fédérale de Lausanne) (2018). *Multi-camera pedestrians video — CVLAB*. URL: https://cvlab.epfl.ch/data/pom (visited on 18/04/2018).

ETHZ (Eidgenössische Technische Hochschule Zürich) (2018). *Moving Obstacle Detection in Highly Dynamic Scenes*. URL: https://data.vision.ee.ethz.ch/cvl/aess/dataset/ (visited on 18/04/2018).

Karlsruhe Institute of Technology (2018). *The KITTI Vision Benchmark Suite*. URL: http://www.cvlibs.net/datasets/kitti/ (visited on 17/04/2018).

LARlabs (2018). *ATLAS project*. URL: http://atlas.web.ua.pt/.

Max-Planck-Gesellschaft (2018). *HumanEva Dataset*. URL: http://humaneva.is.tue.mpg.de/datasets%7B%5C_%7Dhuman%7B%5C_%7D2 (visited on 07/07/2018).

MITSUBISHI MOTORS (2018). *Specifications — i-MiEV*. URL: https://www.mitsubishi-motors.com/en/showroom/i-miev/specifications/ (visited on 20/04/2018).

OpenCV (2018a). *Background Subtraction*. URL: https://docs.opencv.org/3.1.0/db/d5c/tutorial%7B%5C_%7Dpy%7B%5C_%7Dbg%7B%5C_%7Dsubtraction.html (visited on 29/04/2018).

— (2018b). *Camera Calibration and 3D Reconstruction - OpenCV 2.4.13.6 documentation*. URL: https://docs.opencv.org/2.4/modules/calib3d/doc/camera%7B%5C_%7Dcalibration%7B%5C_%7Dand%7B%5C_%7D3d%7B%5C_%7Dreconstruction.html (visited on 05/06/2018).

OpenCV 2.4.13.6 documentation (2018). *Eroding and Dilating*. URL: https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion%7B%5C_%7Ddilatation/erosion%7B%5C_%7Ddilatation.html (visited on 29/04/2018).

Point Cloud Library (2018). *About - Point Cloud Library (PCL)*. URL: http://pointclouds.org/about/ (visited on 02/06/2018).

PointGrey (2018). *Zebra2 2.0 MP Color GigE / HD-SDI (Sony ICX274)*. URL: https://www.ptgrey.com/zebra2-2-mp-color-gige-vision-hd-sdi-sony-icx274-camera (visited on 20/04/2018).

ROS Wiki (2018a). *rosbag*. URL: http://wiki.ros.org/rosbag.

— (2018b). *rviz*. URL: http://wiki.ros.org/rviz (visited on 05/05/2018).

SICK (2018a). *LD-MRS400001 — Detection and ranging solutions — SICK*. URL: https://www.sick.com/de/en/detection-and-ranging-solutions/3d-lidar-sensors/ld-mrs/ld-mrs400001/p/p112355 (visited on 20/04/2018).

— (2018b). *LMS151-10100 — Detection and ranging solutions — SICK*. URL: https://www.sick.com/de/en/detection-and-ranging-solutions/2d-lidar-sensors/lms1xx/lms151-10100/p/p141840%20https://www.sick.com/de/en/detection-and-ranging-solutions/3d-lidar-sensors/ld-mrs/ld-mrs400001/p/p112355 (visited on 20/04/2018).

Singapore-MIT Alliance for Research and Technology (2018). *SMART - Singapore-MIT Alliance for Research and Technology*. URL: https://smart.mit.edu/ (visited on 17/04/2018).

Uber Advanced Technologies Group (2018). *Uber ATG - Car*. URL: https://www.uber.com/info/atg/car/ (visited on 06/07/2018).

Waymo (2018). *Waymo*. URL: https://waymo.com/ (visited on 17/04/2018).

Appendices

Appendix A

Augmented Perception Package

The Augmented Perception package is a ROS package built specifically for the ATLAS-CAR 2 with the aim to develop a better ball detection for the calibration package and a detection, tracking and labelling system.

The package features 3 main nodes:

- `ball_detection_node`
  - Node used for the development of the camera calibration ball detection algorithm improvement
- `labelling_node`
  - Core of the package. Contains all tools related to detection, tracking and labelling.
- `dataset_playback_node`
  - Extra feature that is used to check the datasets printed out by the `labelling_node`.

In order to visualize package functionalities and to , some `roslaunch` files were created:

- `ball_detection_playback`
  - Launches the Rviz and plays the rosbag used for calibration tests.
  - The `ball_detection_node` is executed and the detection of the ball can be seen in the Rviz.
- `calibration_playback`
  - Launches the Rviz and plays the rosbag used for calibration tests.
  - The `calibration_gui` from the `calibration_gui` package is executed and the ball in the rosbag is used to simulate calibration.
- `drivers`
  - Launches the drivers of all sensors and camera.
- `static_transform_publisher`

  - Launches the frame publisher

- `labelling`

  - Sets up environment for the labelling node. Launches the Rviz, the transform publisher and plays the rosbag passed by parameter using the improved rqt_bag.

A.1 Ball Detection Node Interface

This node is the core to the camera calibration's ball detection. The node is launched with the following command:

```
rosrun augmented_perception ball_detection_node
```

The interface uses keyboard and mouse inputs:

- **SPACEBAR** - Update the background frame
- **P** - Pause the actual frame
- **Mouse Left Click** - Picks the color used in the color fitlering
  - Used to click in the ball to get its color

A.2 Labelling Node Interface

This node is the core of the augmented perception package. The node is launched with the following command:

```
rosrun augmented_perception labelling_node
```

The interface uses keyboard and mouse inputs:

- **Q** - Quit / Close Application
- **C** - Clear Image
  - Used to clear the bounding box when tracking an object.
- **L** - Label object
  - When the tracking is done, the object is labelled.
- **S** - Save Templates
  - When the tracking is done, the templates of the objects tracked can be stored in the `labelling` folder.
- **M** - Manual Mode On / Off

  - Switch between manual mode and semi-automatic mode.

- **P** - Print the dataset
  - When the sequence of annotations is done a dataset can be printed out and stored in the `datasets` folder.

- **Left Mouse Click** - Choose the target to follow
  - When on semi-automatic mode, this switches the tracking to manual mode temporarily until the tracking of this object is done.

A.3 Playback

The playback node is used to preview the dataset while the rosbag runs in background. To execute the node run the following command:

```
rosrun augmented_perception dataset_playback_node <dataset
 file>
```

A.4 Dataset Stats Script

A small python script was implemented to check the stats of the datasets. To run it execute the following command

```
rosrun augmented_perception dataset_stats.py <dataset file>
```

The output of the script should present the possible labels and the amount of each one in the given dataset. In listing A.1 an example of the output can be seen.

```
('car: ', 63)
('van: ', 10)
('people: ', 10)
('bicycle: ', 2)
('sign: ', 1)
('misc: ', 3)
('DontCare: ', 31)
```

Listing A.1: Output of the Dataset Stats Script.