Aalto University
School of Electrical
Engineering

Michael Strohmeier

Development and Evaluation of a Human-Robot Interface for an Autonomous All-Terrain Transportation Vehicle in order to Localize, Track and Follow a Human.

School of Electrical Engineering

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Technology.

Espoo, 18.08.2014

Thesis supervisors:

Prof. em. Aarne Halme

Prof. Thomas Gustafsson

Thesis instructor:

Dr. Sami Terho

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Michael Strohmeier

Title: Development and Evaluation of a Human-Robot Interface for an Autonomous All-Terrain Transportation Vehicle in order to Localize, Track and Follow a Human.

Date: 18.08.2014 Language: English Number of pages: 11+82

Department of Electrical Engineering and Automation

Professorship: Automation Technology Code: AUT-84

Supervisors: Prof. em. Aarne Halme, Prof. Thomas Gustafsson

Instructor: Dr. Sami Terho

Full solutions for automatic transportation vehicles in unstructured environments are restricted to military applications, although possible mission scenarios exist for civil and space applications. In this thesis, a user-friendly, human-robot interface is implemented for an all-terrain electric vehicle. The interface allows the vehicle to follow a moving user autonomously by combining 3D LIDAR measurements with intensity images. The method detects clusters within the LIDAR scan that match the human appearance and resolves potential ambiguities based on image tracking results. The interface can be controlled using different input devices. A detailed evaluation compares the input devices, analyzes the implemented algorithms, and validates the overall system behavior outdoor under real-world conditions. The developed system is able to follow its user as long as he is completely visible.

Keywords: Automatic Transportation, Human Detection and Tracking, Human Following, Car-like Steering, 3D Perception, Image Perception

Preface

In 2009, I first heard about the international Master's degree in Space Science and Technology, SpaceMaster. At that time, I started my studies in Würzburg, Germany. During my Bachelor, all my efforts focused on becoming a part of the SpaceMaster program. Being part of this program, I was able to study in three different countries together with fellow students from all around the world. This Master's thesis is the result of the journey that I started five years ago.

First of all, I wish to express my gratitude to my supervisor Prof. em. Aarne Halme for his guidance throughout my work.

I would also like to thank the ESA Directorate of Human Spaceflight and Operations for the awarded scholarship and their great financial support.

Special thanks go to my instructor Dr. Sami Terho for numerous, fruitful discussions and continuous feedback, and to Lic.Sc. Tomi Ylikorpi for all the help he provided during my stay at Aalto University.

I sincerely thank Alex Battiston and Ville Toiviainen for their work on the vehicle as well as my fellow SpaceMaster colleagues for their suggestions and tips.

Finally, I would like to thank my family and my girlfriend Rebecca, for all their love and support, especially during the last year.

Otaniemi, 08.08.2014

Michael Strohmeier

Contents

List of Symbols

α	Hyper-parameter for GPB ground segmentation
B	Minimum distance for initial seeds
b_n^m	Bin n in segment m
d	Maximum size along the car x-axis for a valid person cluster
d_{max}	Maximum distance between adjacent points in a valid person cluster
$\delta(t)$	Calculated control steering angle applied to vehicle
δ_{ij}	Kronecker delta
$e(t)$	Distance error for the vehicle speed control
F	LIDAR–Camera projection matrix
h	Maximum size along the car z-axis for a valid person cluster
h_c	Maximum height of the center of gravity for a valid person cluster
H_n^m	Set of all points p_i' in segment m and bin n
K	Matrix describing the ground model estimated by GPB segmentation
$K(r_i, r_j)$	Covariance between pairs of seeds r_i and r_j based on Gaussian process
K_{XB3}	Intrinsic parameters of the Bumblebee XB3
l_i	Length-scale parameter for point p_i
N_{max}	Maximum number of points for a valid person cluster
N_{min}	Minimum number of points for a valid person cluster
p_i	2 D coordinates of point i: $(r_i, z_i)^T$
p_i'	3 D coordinates of point i: $(x_i, y_i, z_i)^T$
PG_m	Set of points with minimum height z_i in each bin n for segment m
$P_{b_n^m}$	All points in bin n in segment m
r_i	Radial distance of point i in the x-y plane
R_{max}	Maximum distance for GPB segmentation
S_m	Segment m
σ_f^2	Hyper-parameter for GPB ground segmentation
σ_n^2	Hyper-parameter for GPB ground segmentation
T_s	Maximum height for initial seeds
$v(t)$	Calculated control speed
v_k	Kalman Filter measurement noise
$v_{lim}(t)$	Limited control speed applied to vehicle
w	Maximum size along the car y-axis for a valid person cluster
w_k	Kalman Filter process noise
x_i	Distance of point i along car x-axis, origin in LIDAR
$x(t)$	Distance of human user along car x-axis, origin in LIDAR
X_k	Kalman Filter state
y_i	Distance of point i along car y-axis, origin in LIDAR
$y(t)$	Distance of human user along car y-axis, origin in LIDAR
z_i	Distance of point i along car z-axis, origin in LIDAR
$\bar{z}_*$	GPB estimation of mean height for tested bin distance r_*
Z_k	Kalman Filter measurement

List of Abbreviations

AATTV	Autonomous All-Terrain Transport Vehicle
AdaBoost	Adaptive Boost
ATV	All-Terrain Vehicle
CAN	Controller Area Network
CMU	Carnegie Mellon University
CMT	Consensus-based Matching and Tracking
C-HOG	Circular Histograms of Oriented Gradients
DARPA	Defense Advanced Research Projects Agency
DRATS	Desert Research and Technology Studies
EVA	Extra-Vehicular Activities
fps	frames per second
FOV	Field of View
GPB	Gaussian-Process-Based
GPS	Global Positioning System
GPU	Graphics Processing Unit
HOD	Histograms of Oriented Depth
HOG	Histograms of Oriented Gradients
HRI	Human Robot Interaction
HSV	Hue Saturation Value
ICR	Instantaneous Center of Rotation
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
LRV	Lunar Roving Vehicle
LS3	Legged Squad Support System
NDVI	Normalized Difference Vegetation Index
NN	Nearest Neighbour
OpenCV	Open Source Computer Vision Library
PCL	Point Cloud Library
PS3	PlayStation 3
PTZ	Pan-Tilt-Zoom
RANSAC	Random Sample Consensus
R-HOG	Rectangular Histograms of Oriented Gradients
RGB	Red Green Blue
RFID	Radio Frequency Identification
ROS	Robot Operating System
SEV	Space Exploration Vehicle
SMSS	Squad Mission Support System
SOI	Sector of Interest
SVM	Support Vector Machine
TLD	Tracking Learning Detection
TOF	Time of Flight
UDP	User Datagram Protocol
UGV	Unmanned Ground Vehicle

List of Figures

List of Tables

1 Introduction

Since the last Apollo mission in 1972, no human has set foot on the lunar surface again [1]. During that last mission, a Lunar Roving Vehicle (LRV) was used to transport the astronauts, their tools and communication equipment as well as collected samples [2]. Figure 1.1 shows the LRV.

Figure 1.1: Gene Cernan aboard the Lunar Roving Vehicle during the Apollo 17 mission [3].

The rovers used in the Apollo missions were operated manually by the astronauts. Therefore, the LRVs could only provide limited storage space, since two astronauts had to have space aboard the vehicle, too. This storage problem could have been solved by a vehicle that was able to drive on its own or follow an astronaut. Unfortunately, vehicles were not able to drive autonomously 40 years ago. However, this has drastically changed within the last decade and the increasing interest in Unmanned Ground Vehicles (UGVs).

The announcement of the first DARPA Grand Challenge[1] in 2004 triggered a research boom in the field of fully autonomous UGVs. While none of the competitors managed to drive more than 12 km of the rural track in the first challenge [4], only three years later, a fully autonomous 96 km drive through an urban environment was completed as part of the DARPA Urban Challenge [5]. In 2011, Google published details about its hitherto secret project, the Google car, creating a new state of the art for autonomous cars [6]. By that time, the Google car had already managed to drive more than 225.000 km completely autonomous in real-world traffic situations [7]. In March 2014, Google introduced the first car that operates completely autonomous. The first prototype was built from scratch without steering wheel, accelerator pedal or brake pedal [8].

However, UGVs are not only able to operate in urban environments. Porting the sensors and intelligence of cars to All-Terrain Vehicles (ATVs) creates a new range of applications for UGVs. They may, for example, be used for transportation tasks in

[1]The US Defense Advanced Research Projects Agency (DARPA) has sponsored three challenges for autonomous car since 2004.

unstructured environment. UGVs that are used for transportation in unstructured and rough environment will be referred to as Autonomous All-Terrain Transport Vehicles (AATTVs) within this thesis report. Although, AATTVs are not used in concrete civil or space applications yet, multiple scenarios are conceivable. Conversely, AATTVs are already frequently supporting military squads and often combined with a human-following interface. In order to follow a human user, three subproblems have to be solved: The identification of the user, the localization and tracking of the user and finally the user-following itself. Especially, the localization and tracking of the user is immensely important for the system's reliability in real-world applications.

The goal of this thesis project is the development, implementation and evaluation of a human-robot interface that allows an AATTV to identify, track, and follow a human user autonomously.

Chapter 2 reviews existing AATTVs and interface solutions that are currently available for military, civil and space applications. Furthermore, current mission scenarios and possible applications are described.

Chapter 3 gives an introduction to the Human Robot Interaction (HRI) problem in general and describes HRI demands for an AATTV in particular. Additionally, the challenges and benefits of a human-following interface are described.

Chapter 4 states the background of different localization and tracking approaches. Furthermore, several state-of-the-art examples for human detection and tracking based on the utilized sensor types are explained.

After summarizing the major system requirements from the current state-of-the-art, Chapter 5 introduces the overall system concept and describes the available hardware and the software that are used in this project.

Subsequently, Chapter 6 gives a detailed description of the system implementation. Thereby, each implemented subsystem is presented and put into context with respect to the overall system.

Chapter 7 extensively evaluates each implemented subsystem on its own. Furthermore, the performance of the overall system is assessed.

Finally, the achievements of this thesis project are summarized in Chapter 8. Furthermore, the development stages of the system are reviewed and system limitations are discussed. Possible solutions to overcome the system limitations are outlined as part of possible future work.

2 Autonomous All-Terrain Transportation Vehicles

Depending on the field of application, AATTVs can be divided into different categories. Therefore, this chapter reviews state-of-the-art AATTVs within three different scopes. The first scope focuses on military applications and illustrates modern systems based on three examples: The R-Gator, the Squad Mission Support System (SMSS) and the Legged Squad Support System (LS3) AlphaDog. Although AlphaDog's locomotion is not wheel based, it is nevertheless an excellent example of autonomous transportation. In the second scope, civil AATTVs are described and exemplified by two state of the art research platforms: Grizzly and Seekur. The last scope outlines AATTVs in space applications and describes the Space Exploration Vehicle (SEV), a current research project for manned surface explorations in space.

2.1 Military AATTVs

Logistics are a central aspect in modern warfare. Weapons, ammunition as well as necessary supplies have to be moved to where they are needed the most. This commonly means to transport critical goods over long distances through hostile environments. However, AATTVs are able to complete these dangerous and dirty tasks, without setting human life at risk. Furthermore, soldiers accompanied by an autonomous transport vehicle are able to carry more supplies, thus increasing their efficiency and time of march. Additionally to the provided transportation support, an AATTV can also offer protection during combat.

Military AATTVs can not afford high error rates. Consequently, strict requirements have to be met, even in very hazardous settings. Besides a very high reliability, the vehicles have to master rough terrain and different weather conditions. They have to provide sufficient armor, while being agile at the same time. Furthermore, they have to offer enough storage space. Finally, they have to provide a very easy and safe-to-use human interface, especially if operated during combat.

The following subsections give each an example of state-of-the-art transportation vehicles in military applications.

2.1.1 R-Gator

The R-Gator is a UGV based on the John Deere XUV utility vehicle [9]. Figure 2.1 shows the vehicle. The R-Gator has a 4 wheel drive and Ackermann steering. It reaches a maximum speed of 50 km/h, carrying a payload of at most 500 kg. With a ground clearance of 28 cm, the R-Gator is suitable for rough terrain. The vehicle is equipped with two color cameras, one infrared camera, one PTZ camera, one NDVI camera, two tilting SICK laser scanners and an automotive radar unit. A GPS receiver, an IMU and wheel encoders are mounted, too.

Figure 2.1: The R-Gator based on the John Deere XUV utility vehicle [9].

Human-Robot Interaction The R-Gator can be operated in six different modes: Manual operation, teleoperation, waypoint driving, direction driving, playback and silent sentry.

Using the R-Gator's steering wheel and pedals, it can be controlled like a regular car in the *manual operation* mode. In the *teleoperation* mode, the vehicle can be driven using a commercial game controller. Hereby, the obstacle avoidance system is limiting the maximum speed in order to avoid collisions. The *waypoint driving* mode allows the user to send waypoints to the vehicle. Once a waypoint is received, the vehicle navigates autonomously to the desired location. The waypoints can be selected using a map user interface or any camera view. In the *direction driving* mode, the user specifies a certain heading. Subsequently, the robot drives into the given direction until the user interacts with the robot again. Directions are given in the same way as waypoints. The *playback* mode allows to record a certain path in any of the previously described modes. Once a track is recorded, it can be replayed in this mode. Conversely, the *silent sentry* mode implements a surveillance mode. Upon mode entry everything is turned off except for the cameras. Thus, the vehicle can be used as a static surveillance platform.

2.1.2 Squad Mission Support System

The SMSS is a multi-mission UGV produced by Lockheed Martin [10, 11]. In 2011, it was successfully evaluated in combat for various transportation tasks during a five month test period in Afghanistan [10]. Different vehicle versions are available, each optimized for a specific task. Figure 2.2 shows the transport and logistics vehicle version. It is able to transport a payload of up to 682 kg. Its wheels are configured as a 6×6 drive with skid steering, allowing in place rotations. Since the vehicle has a vertical step of 41 cm and overcomes gaps of 80 cm, it can easily handle off-road and rough terrain.

In order to interact with a user and to perceive its environment, different sensors are equipped to the vehicle[2]. A 3D 360° LIDAR, a 2D laser scanner, and a camera

[2]The list of sensors is based on observations made from videos on the developers web page [11].

system are mounted in the front. In the back of the vehicle, there is another camera system installed. Additionally, a GPS receiver is attached.

Figure 2.2: The Squad Mission Support System [12].

Human-Robot Interaction The user can interact with the SMSS using a hand-held device with a touchscreen, several buttons, and a joystick. The device allows the user to switch between three modes. The vehicle can be commanded using simple voice commands in the *Voice Control* mode. In the *Plan* mode, the vehicle can be steered by giving GPS goals. The goals are sent using a map that is displayed on the hand-held device. The position of the user relative to the vehicle is also displayed on the device [11]. In addition to that, the vehicle is able to follow a person in the *Follow Me* mode.

2.1.3 LS3 - AlphaDog

The Legged Squad Support System (LS3), also known as AlphaDog, is a quadruped robot manufactured by Boston Dynamics [13]. It is mainly based on its predecessor, the BigDog. Thus, AlphaDog is not a vehicle in the traditional sense. Nevertheless, AlphaDog is an excellent example of automated transportation. The leg based locomotion allows the robot not only to perform in place rotations, but also to master very rough and step terrain. Furthermore, the robot is able to walk through high grass, on snow, as well as on icy and wet terrain. Moreover, the robot is able to recover from a loss of balance, even if it falls over completely.

The LS3 utilizes a tilting laser scanner and two stereo vision systems, in order to obtain a model of its environment. One of the stereo vision systems is mounted on a rotating platform, while the second system has a fixed orientation pointing towards the floor. Additionally, an integrated GPS receiver allows a precise localization [14]. In its latest version, a sensor is attached to the user, which allows the robot to localize and follow him [15].

In contrast to other solutions like the SMSS, the LS3 was not designed to offer protection during combat. The robot's only purpose is to work as a transportation system in very rough and step terrain, which is not suitable for wheeled solutions. Figure 2.3 shows the robot in such a difficult environment. The LS3 is able to carry a payload of 180 kg during a 24 hour mission time. The maximum distance that can be covered without refueling is about 32 km.

Figure 2.3: The Legged Squad Support System during military tests on Hawaii in 2014 [16].

Human-Robot Interaction Using voice commands, the LS3 can be commanded to follow a leader either in the *Follow Tight* or the *Follow Corridor* mode [13]. While the user is responsible to choose a suitable path in the follow tight mode, the LS3 is able to navigate completely on its own in the corridor mode. Additionally, the robot can be controlled by sending GPS positions. It is also possible to command the robot using a game controller.

2.2 Civil AATTVs

Although, driver-less transport systems are commonly used for various logistic tasks in indoor applications, they are mostly restricted to structured environments, like warehouses [17, 18]. An autonomous forklift for a semi-structured environment is described in [19]. However, AATTVs are very rare with concrete civil applications. In fact, there are only commercial research platforms available without a specific application. The two most recent platforms are described in Subsection 2.2.1. Both platforms support the open-source Robot Operating System (ROS), offering an easy way to implement complex autonomous tasks [20, 21].

Different applications range from support in agriculture and mining to hunting and forestry. A possible mission scenario is described in Subsection 2.2.2.

2.2.1 Research Platforms

Grizzly The Grizzly is an autonomous all-terrain vehicle research platform developed by Clearpath Robotics [22]. Figure 2.4 shows the vehicle. It can transport a

payload of up to 600 kg with a maximum speed of 19 km/h. Grizzly has 4×4 skid steering drive with a ground clearance of 20 cm. An internal GPS, an IMU and high precision wheel encoders allow a precise localization [23]. Additionally, the Grizzly can be interfaced with a SICK laser scanner, a Velodyne 3D LIDAR, and a stereo camera for more detailed perception of its environment.

Figure 2.4: The autonomous all-terrain vehicle research platform Grizzly [22].

Human-Robot Interaction In the standard version, a PlayStation 3 (PS3) game controller can be used to navigate the robot. However, based on ROS, plenty of different solution can be developed.

Seekur Jr. The Seekur Jr. is developed by Adept MobileRobots [24] and based on its predecessor Seekur. Seekur Jr. is available in three different configurations: Laser Mapping and Navigation, Outdoor, and Outdoor Manipulator. Figure 2.5 shows the Seekur Jr. in the outdoor configuration. The maximum payload is 50 kg. Similar to Grizzly, Seekur Jr. has also a 4×4 skid steered drive. However, its maximum speed is limited to 1.2 m/s. Optional sensors include a SICK laser scanner, an IMU, stereo vision and a PTZ camera.

Figure 2.5: The autonomous all-terrain vehicle research platform Seekur Jr. [21].

Human-Robot Interaction The Seekur Jr. can be controlled via joystick. Furthermore, the robot can be teleoperated using a graphical user interface. Again, similar to Grizzly, other solutions can be easily implemented using ROS.

2.2.2 Possible Mission Scenario

A possible mission scenario for AATTVs at the size of Grizzly is the use in hunting and forestry. Since forests are naturally highly unstructured, and cluttered environments, they are difficult to cross, especially for big vehicles like off-road cars or trucks. This is aggravated by different soil conditions like sand, mud, branches, rocks and scree. However, AATTVs are designed to cope with this kind of terrain. Thus, a hunter could be supported in carrying his hunting kit or prey. The AATTV could also be used to transport equipment that is needed for cutting down and chopping a tree. A user friendly interface could utilize voice recognition to receive and execute simple voice commands. If the robot is able to follow the user autonomously, the user is free to focus on his path and his environment instead of the robot control.

The requirements for civil AATTVs differ slightly from military requirements, since there is no need for heavy armor. However, similar to military, civil AATTVs should be able to handle rough terrain, different soils as well as changing weather conditions. Moreover, human safety plays a very important role in designing a system for any civil application. Since civilians are usually untrained users, an easy to use human-robot interface is required, too.

2.3 Space

Robots have been used extensively in space exploration. Yutu, the most recent moon-exploration robot, launched in December 2013 [1], while NASA's most popular rover, Curiosity, is exploring the martian surface for more than 2 years [25]. Since, the last manned Apollo mission to moon was in 1972 [1], there are currently no human-supporting space-exploration vehicles in use. However, this topic is intensively researched.

This chapter outlines the current research on extra-terrestrial vehicles in Subsection 2.3.1 and describes two possible mission scenarios in Subsection 2.3.2.

2.3.1 Current Research

NASA's Space Exploration Vehicle (SEV) is a concept vehicle for future in-space and surface missions [26, 27]. The pressurized SEV cabin is designed to hold a crew of two for 14 days. However, it provides enough space to host four people in case of emergencies. Furthermore, the cabin offers a suit-port that allows the crew to get into their spacesuit and out of the vehicle easily. Figure 2.6 shows the SEV surface concept during the Desert Research and Technology Studies (DRATS) in 2009. The concept vehicle has a maximum payload capability of 1000 kg. It can travel with a speed of up to 10 km/h using six pivoting wheels, which allow the vehicle to move in any direction.

Human-Robot Interaction The SEV can be controlled using a joystick. Since joysticks are located in the cockpit of the cabin and at the suit-port, the vehicle can be driven from the in- and outside of the SEV. Displays provide information about the current heading, the SEV's internal state and from external cameras.

Figure 2.6: The Space Exploration Vehicle during DRATS in 2009.

2.3.2 Possible Mission Scenarios

An AATTV in space could accompany astronauts during Extra-Vehicular Activities (EVA) on lunar or planetary surface missions. In one scenario, an AATTV could be utilized to transport different mineral and soil sampling equipment needed for geological explorations. Furthermore, storage space for collected samples could be provided easily. In this scenario, the AATTV could also supply energy for drilling tools.

In another scenario, AATTVs could replace the large spacesuit backpacks that are currently needed to supply astronauts with oxygen. Oxygen tanks or bottles could be transported with an AATTV, which is autonomously following an astronaut on his exploration. In that way, empty oxygen bottles on the astronauts back could just be filled up with the tank or replaced by a full bottle at any time. Currently, the MIT and the NASA are working on BioSuit, a spacesuit designed for EVAs on other planets with the goal to provide increased mobility and the possibility for quick bottle changes [28].

The requirements for an AATTV in space are similar to military requirements. The vehicle has to provide a very high reliability, while operating in extremely hazardous environments. Additionally, all components have to be radiation hardened.

3 Human Robot Interaction

In order to develop such a system, different possibilities and requirements for the Human Robot Interaction (HRI) with an AATTV have to be analyzed. Therefore, Section 3.1 gives an introduction to the HRI problem in general. Section 3.2 outlines the specific requirements interfacing a human with an AATTV and emphasizes the importance of solutions that allow an AATTV to follow a human autonomously. In Section 3.3, subproblems that have to be solved in the implementation of a human-following interface are outlined.

3.1 HRI Problem

The goal of the HRI is *"the design, understanding, and evaluation of robotic systems, which involve humans and robots interacting through communication"* [29]. According to [30], the HRI problem can be divided into five constituents, which can be each influenced at a certain level by the system designer.

Firstly, there is the robots level of autonomy. It defines primarily the frequency of human robot interaction that is required to accomplish a task.

The second component is the nature of information exchange, which includes the choice of a mission suitable information medium. Common media types are categorized into visual displays, movement-based signaling, natural language, non-speech audio, as well as physical interactions and haptics.

Another part tackles possible team structures and considers one or more users interacting with one or more robots.

A fourth part that needs to be considered is the adaption, learning and training capability of the robot and the user. Easy to use interfaces come with high demands towards the robot's intelligence, while complex interfaces might require highly trained operators.

The last part to be considered is the shape of the task itself and how it should be executed.

3.2 HRI for AATTVs

Interacting with field robots in unstructured environment, traditional HRI approaches consider humans to be spectators or bystanders and restrict the collocation of humans and robots to service robots in structured environments [31].

Nevertheless, the combination of field robots, which are designed to operate in a rough environment, and service robots, whose purpose it is to provide support and assistance for humans, interdicts to treat the operator merely as a spectator. Consequently, specific requirements result towards the HRI for AATTVs. Especially, HRI awareness has to be considered for AATTVs, since they are operated in the vicinity of humans [32]. Above all, this includes plausibility checks of received commands in order to avoid humans getting harmed by the robots.

Since AATTVs are both, field and service robots, two operational modes are considered to be advantageous: A manually operated and semi-autonomous mode.

Manually Operated Mode In the manually operated mode, the robot is fully controlled by the user. This operation mode requires the user's complete attention, while the autonomy demands towards the robot are rather low. A manually operated mode enables the AATTV to perform maneuvers with high speed and accuracy. In this mode, a HRI could be realized using direct physical interaction with the AATTV. The use of joysticks or other game controllers is conceivable, too.

Semi-autonomous Mode The semi-autonomous mode allows the user to focus on other tasks, while still having full control over the robot. Consequently, a higher level of autonomy is required compared to the manually operated mode. As can be seen from the previous examples in Subsection 2.1.2 and 2.1.3, interfaces utilizing movement-based signaling and natural language provide a preferable solution. Additionally, touchscreen devices are used to send GPS goals. State of the art interfaces are based on simple voice commands and gestures, as well as solutions that allow a vehicle to follow a user autonomously.

Comparison Comparing both modes, the semi-autonomous mode seems to offer more advantages, especially under rough circumstances. All interfaces described for this mode offer an easy and intuitive control of the vehicle. Nevertheless, not all interface solutions are suitable for military or space missions. During combat, it is unfavorable for a soldier to use a touchscreen device in order to navigate a vehicle. The same applies for astronauts, who might be limited in their mobility by their spacesuit. Therefore, solutions based on voice, gesture or the ability to follow a human should be preferred. Gesture and voice based interfaces provide an optimal solution for commanding a vehicle. In contrast, a human-following ability offers full navigation control, while the user can still focus on other task. Thus, an ideal solution can be achieved by combining an easy-to-use interface with a human-following ability (see Figure 3.1). In this case, gestures, voice commands, touchscreen devices as well as other simple button-based input devices can be used to turn the human-following interface on or off.

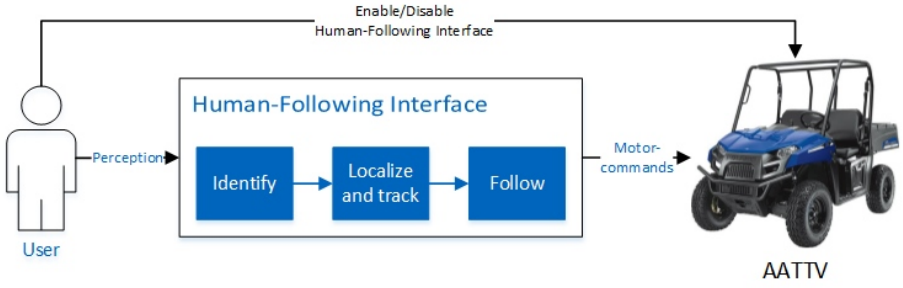


Figure 3.1: Human-robot interaction with a human-following AATTV: The human-following mode can be enabled using voice or gesture commands. In order to implement the human-following ability, a detection and tracking interface is required.

3.3 Human-Following Interface

The robot has to solve at least three subproblems when implementing a human-following interface as shown in Figure 3.1: It has to be able to identify the user, continuously localize and track him, and control the vehicle in a way that allows the robot to follow the user.

3.3.1 User Identification

First, the user has to be identified by the system. This can be done by either creating a unique user characteristic that allows an identification at any time or by defining an initial user position.

Approaches using unique characteristics are usually based on the user's appearance or beacons. Special color coded clothes or patterns are popular solutions to create a unique appearance [33–35]. Conversely, beacons are implemented by transmitting light [36] or radio signals. Radio Frequency Identification (RFID) tags are often used in this context [37, 38].

Approaches that require an initial user position can be implemented in many different ways. The simplest way to obtain an initial position is to start the human-following interface only if the user is located at a fixed position relative to the robot. Another approach determines the user's positions by calculating the sound source direction of voice commands [39]. In [40], the user and therefore the initial position is identified through specific gestures that are used to start the human-robot interaction. Face recognition can be used in a similar way, too [37, 41]. However, other solutions like GPS-based localization are also possible.

Figure 3.2: User identification Workpartner: The robot Workpartner identifies the human user based on color coded clothes.

Comparing both approaches, unique user characteristics allow a continuous user identification, while approaches based on an initial position require some other kind of user tracking in order to localize the user at any time.

As long as the user appearance stays unique, the first approach is by far more robust towards occlusion. In space, for example, dust and dirt could cover or distort unique characteristics, such as color or specific patterns. In addition to that,

solutions that depend on added user characteristics are not feasible within all applications. In military applications for example, color coded clothes and striking patterns make soldiers easy targets for the enemy. Similar, transmitted light or radio signals could be located by enemy troops.

The second approach, however, allows a great flexibility with respect to the user appearance. Thus, there is no need to change clothes in order to use the interface. Furthermore, no additional devices for locating the user are required.

3.3.2 User Localization and Tracking

Once the user is identified, the robot has to be able to localize and track the user continuously. If the user identification is based on an initial position and therefore does not require augmented human tags, independent tracking and localization methods are needed. Since the detection and tracking of humans is a very complex problem, a detailed description of possible solutions is given in Chapter 4.

3.3.3 User-Following

Once the user's position is known, the robot has to calculate a suitable path in order to follow the human. Main difficulties result from the robot's locomotion, its size and environment. For example, a small holonomic robot can use an approach that is a lot simpler than the one needed for a car-sized robot with Ackermann steering. Similarly, a robot in an unstructured but static environment has to consider less obstacles than the same robot in an unstructured and highly dynamic environment. In [42], Gockley et al. suggest two ways of user-following: Direct path-following and direction-following. While the direct path-following allows the robot to follow his user directly, the robot chooses its own path based on the user's direction in the direction-following approach. A similar behavior was outlined for the LS3 in Subsection 2.1.3.

Depending on the available level of autonomy, either solution proposed by Gockley et al. might be more suitable.

4 Human Detection and Tracking

Object tracking in general and people tracking in particular are elementary challenges within machine perception and therefore in mobile robotics. Traditionally, object tracking describes the process of determining an object trajectory within a sequence of two-dimensional intensity images. However, in order to obtain three-dimensional information about the object of interest, range information has to be utilized during the tracking. Depth data can be gathered using Time of Flight (TOF) cameras, cameras in a stereo setup and 2D or 3D Light Detection and Ranging (LIDAR) scanners. Depending on the acquired data, different object recognition and tracking methods are beneficial. As a complete review on the variety of methods presented in different surveys [43–46] is beyond the scope of this work, this chapter focuses on the most common used methods and principles. Therefore, different approaches are classified into vision-based, laser-based, and hybrid approaches and are presented in the subsequent sections.

4.1 Theoretical Background

In order to find and track a person, at least the following three steps have to be performed: Feature selection (4.1.1), object classification (4.1.2), and the tracking itself (4.1.3). In the first step, characteristic features that describe the object are extracted from the image. Subsequently, the classification step tries to identify objects based on the selected features. Usually, the selected features are therefore compared to templates using supervised learning algorithms. In the tracking step, useful information about the object movement is extracted. This information can be used to improve the object detection. Other important steps include image preprocessing, foreground segmentation, and the verification of detected objects [43].

4.1.1 Feature Selection

Selecting unique features from intensity images is crucial for successfully detecting people. Traditional features are color, edges, texture, and optical flow [45]. Very popular feature descriptors for human detection are Haar-like features and Histograms of Oriented Gradients (HOG) descriptors [47]. While traditional features are mostly limited to intensity images, feature descriptors can theoretically be applied to both, intensity and range images.

Color Since color is directly encoded in images, it is one of the simplest features to detect. Different color spaces, like Red Green Blue (RGB) and Hue Saturation Value (HSV) can be utilized (see Figure 4.1).

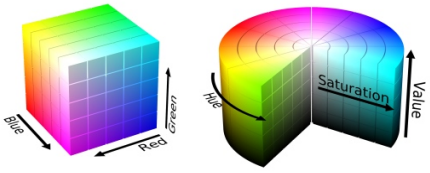
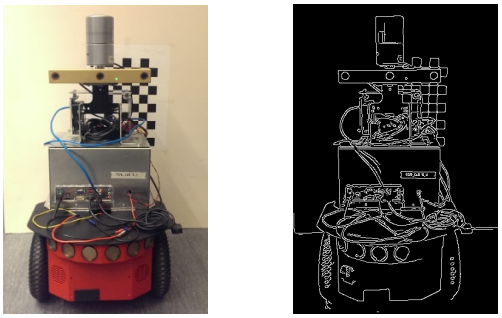


Figure 4.1: Comparsion of different color spaces: RGB and HSV

In contrast to RGB, HSV is approximately a perceptual uniform color space [45]. This allows a simple selection of color regions of interest in the HSV space. Nevertheless, features based on color are light sensitive and thus not robust against changes in illumination.

Edges In contrast to color, edges are features that are relatively robust against illumination changes. An edge detection algorithm, which is still very popular today, was introduced by Canny in 1986 [48]. His algorithm is based on the detection of image gradients with a large magnitude. Figure 4.2 shows the output of the Canny Edge Detector.



(a) Input Image (b) Canny edges

Figure 4.2: Canny edge detection.

Optical Flow A third feature is the observed motion in a sequence of intensity images. The object motion in a picture can be estimated using optical flow, a vector field which describes the apparent velocity of brightness patterns in images [49]. Therefore, this feature is especially useful for detecting moving objects in otherwise static scenes. Early, but still popular approaches were described by Horn and Schunck [49], and Lucas and Kanade [50].

Texture The texture of an object is another feature that can be extracted from images. It characterizes the objects intensity surface. Thus it can describe different patterns and structures on the surface. However, it can not be extracted as simple as image intensities. Special feature descriptors are required in this case [51]. Texture-based features are beneficial if characteristic a priori information about the objects surface is available.

Haar-like features Haar-like features are based on Haar wavelets and were first introduced by Papageorgiou and Poggio [52]. The authors proposed to describe an object class by a new representation based on orientated intensity differences between adjacent image regions. Three 2-dimensional Haar wavelets are outlined in their work and shown in Figure 4.3.

Figure 4.3: Three different Haar wavelets: Vertical, horizontal and diagonal.

In order to use Haar-like features for object detection, for each wavelet an object descriptor has to be obtained and a classification algorithm has to be trained. The descriptors are usually based on a large database containing both, images of the desired object and images of other scenes without the object. Figure 4.4 shows the averaged human descriptors for the three wavelets.

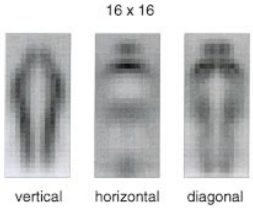


Figure 4.4: Averaged Haar-like feature descriptors [52].

Histograms of Oriented Gradients Histograms of Oriented Gradients (HOG) are another set of feature descriptors. In [53], Dalal and Triggs introduced a feature descriptor based on normalized, local histograms of image gradient orientations in a dense grid. Similar to the Haar-like features, first, a descriptor has to be formed and, subsequently, a classifier to be trained.

The HOG descriptor is formed in three steps. First, the image gradients are calculated using a gradient filter. In the second step, the image is divided into cells and an orientation histogram is computed for each cell. Subsequently, the previously calculated histograms are contrast-normalized in the third step. This increases the invariance to illumination changes and shadowing. The image is therefore tiled into overlapping blocks. Depending on the block geometry, one can distinguish between rectangular (R-HOG) and circular (C-HOG) Histograms of Oriented Gradients. For rectangular blocks, the cells are also divided into rectangles, while for circular blocks each cell has a polar log fashion. Furthermore, the C-HOG center cell can be considered as one or be split up into four segments. The possible block configurations are shown in Figure 4.5.

For a reliable descriptor, these steps have to be applied to a large number of training images and the calculated descriptors have to be averaged. Once the final

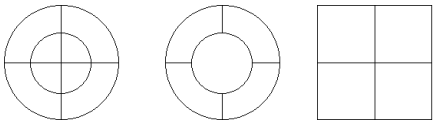


Figure 4.5: HOG block forms: C-HOG, C-HOG with one center cell, R-HOG.

descriptor is obtained, a classifier can be trained. In the original publication, a Support Vector Machine (SVM) classifier is utilized.

Figure 4.6 shows the averaged gradients of a training set, positive and negative SVM weights, a sample picture, the R-HOG descriptor for the sample picture, and R-HOG descriptor weighted with the positive and negative SVM weight.

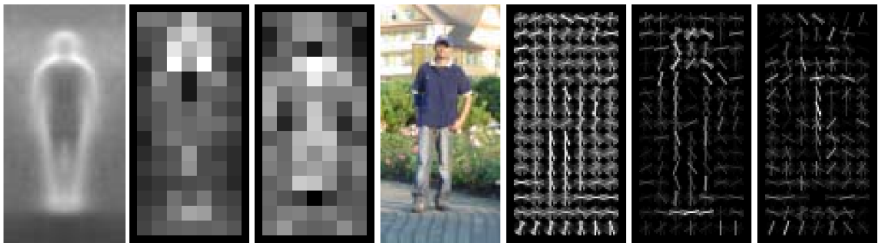


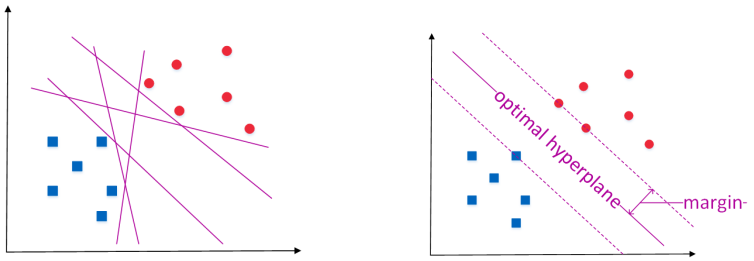
Figure 4.6: From left to right: Averaged gradients of a training set, positive and negative SVM weights, a sample picture, the R-HOG descriptor for the sample picture, and positive and negative weighted descriptor [53].

4.1.2 Object Classification

Object classifiers are based on supervised learning methods. Their task is to determine whether a certain subset of extracted features matches an object description or not. Depending on their error rate, classifiers can be categorized into weak and strong. A one-level decision tree is a simple example for a weak classifier. However, classifying a set of image features is a very complex and challenging task, especially for nonrigid objects. Therefore, strong classifiers like Support Vector Machines (SVMs) are needed. Additionally, it is also possible to combine several weak classifiers.

Support Vector Machines Support Vector Machines (SVMs) are a class of algorithms used for data analyzing. The most popular implementation was published by Cortes and Vapnik in 1995 [54].

The goal of SVMs is to categorize a labeled dataset into two groups: One group that fits a certain description and one that doesn't. Therefore, SVM fits a hyperplane in such a way that new, unlabeled data can be categorized correctly. Figure 4.7 shows a labeled dataset of squares and circles as well as possible hyperplanes. For each hyperplane, a margin can be defined as the minimum distance towards a data sample. Thus, the best-choice hyperplane provides a maximum margin towards both categories.



(a) Possible hyperplanes for classification.

(b) Best fitting hyperplane with maximum margin.

Figure 4.7: Classification based on Support Vector Machines: Dataset consisting of squares and circles separated by different hyperplanes.

Boosting Boosting allows to combine several weak classifiers in order to generate a strong one. The idea was first introduced in 1989 by Schapire [55]. Figure 4.8 exemplifies how two weak decision tree stumps are combined into a stronger classifier. Adaptive Boost (AdaBoost), the most popular boosting method, was described by Freund and Schapire [56]. AdaBoost, assigns weights to classified data based on an initial classification. Thus, depending on the previous classification, weights for data points are either in- or decreased depending on whether they were misclassified or correctly classified, respectively. The adapted weights are then considered during the next classification step.

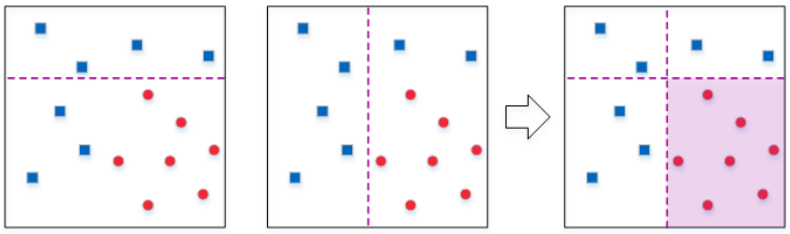


Figure 4.8: Classification based on Boosting: Combination of two weak classifiers into a strong classifier.

4.1.3 Object Tracking

Object tracking allows to predict the future position of a tracked objects. Therefore, it can simplify the detection process and correct detection errors. Furthermore, object tracking can resolve occlusion problems. Different approaches can be divided into three categories depending on the object representation: Point tracking, Kernel tracking and Silhouette tracking [45]. The most popular approach is point tracking and often realized using Kalman and Particle Filters [43].

Another common approach to realize point trackers is based on optical flow. Today, various tracking algorithms are based on optical flow trackers such as the one described by Lucas and Kanade [50] or the Median Flow tracker [57]. Two very

recent trackers that are based on this principle will be described in this section: The Tracking Learning Detection (TLD) algorithm and the Consensus-based Matching and Tracking (CMT) approach.

Tracking Learning Detection Tracking Learning Detection (TLD) is a real-time tracking algorithm that is able to track any object within an image sequence based on a single bounding box in the first frame. The algorithm was originally developed for `MATLAB` by Kalal [58]. With OpenTLD, a `C++` implementation was published by Nebehay [59]. The strong performance of the TLD tracker results from the feedback loop used for online training. Figure 4.9 gives an overview of the TLD framework.

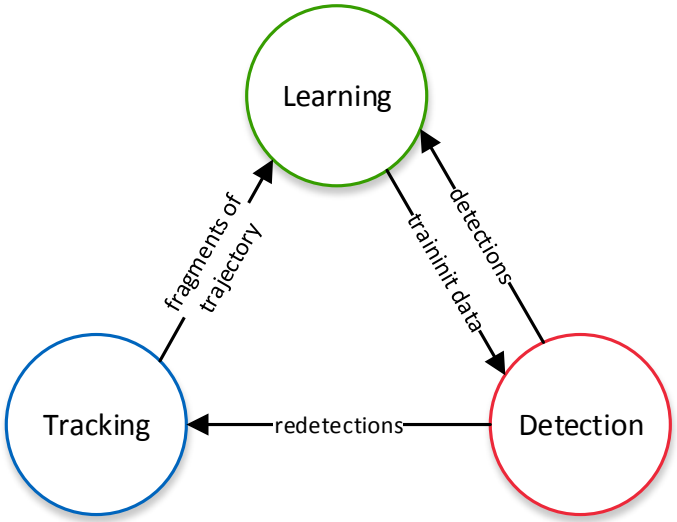


Figure 4.9: The Tracking Learning Detection algorithm [58].

As the name TLD indicates, the algorithm consists of three main parts: The tracking, learning and detection of an object.

The tracking part is based on a median-flow tracker which utilizes a Lucas-Kanade optical flow tracker. Additionally, the tracker is able to detect tracking failures if the object of interest is not visible in the processed frame. The information obtained from the tracker is forwarded to and utilized by the learning component.

The learning component consists of two major parts: A P- and an N-expert. The P-expert gathers positive object descriptors, allowing to obtain a more general model of the tracked object. Conversely, the N-expert creates negative descriptors based on the image background. The information of both experts is then provided to the object detector.

The object detector is based on the learned model. It is initialized by the learning component. The detection itself is realized using a sliding window approach and cascaded classifiers. The cascade consists of three stages: A patch variance classifier, an ensemble classifier and a 1-NN classifier. The detection results are fed into the tracker if an object had been lost and re-detected, or detected for the first time.

Consensus-based Matching and Tracking The CMT tracker is a real-time tracking algorithm that allows long-term and model-free tracking. The approach was presented by Nebehay and Pflugfelder in 2014 [60]. Open-source implementations are available for `Python` and `C++`. The main idea of the tracker is to detect keypoints, which each vote for the center of the object. Based on the votes, outliers can be detected and the 2D pose of the object in the image estimated.

The CMT tracking algorithm consists of three major steps. The first step is responsible for the matching and tracking of keypoints. Therefore, BRISK keypoints are extracted in each frame. The extracted keypoints are then matched with the keypoints extracted from the first frame. The matching is based on the distances of each keypoint towards its neighbors. In order to keep track of the matched keypoints, a Lucas-Kanade optical flow tracker is used.

In the second step, each keypoint casts a vote towards the object center. This step can take translational and rotational changes as well as possible changes in scale into account.

In the final step, the object's pose is determined by removing outliers and finding a consensus for the cast votes. Therefore, keypoints with similar votes are clustered together using a hierarchical agglomerative clustering. The biggest subset of keypoints is defined as the new consensus.

The voting and consensus finding process are illustrated in Figure 4.10.

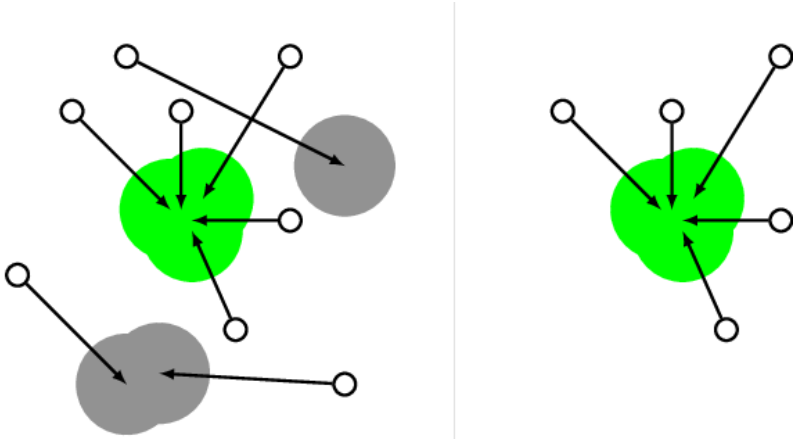


Figure 4.10: The CMT tracker voting and consensus finding: Each point casts a vote towards the object's center. The cluster with the biggest number of keypoints is defined as consensus [60].

4.2 Vision-Based Approaches

Vision-based approaches utilize intensity images for the detection and tracking of people. Difficulties in these approaches result from object occlusion, illumination changes, nonrigid and articulated objects, complex object motions, real time requirements, ego motion, and measurement noise [45]. Depth information can be obtained using stereo vision or TOF cameras. Vision-based approaches are widely used among different authors.

In [61], Sataka and Miura present a person following robot utilizing stereo vision. The human detection is based on depth templates generated from person shapes. A SVM-based verifier is used in order to eliminate false detections. The person tracking is realized with an Extended Kalman Filter.

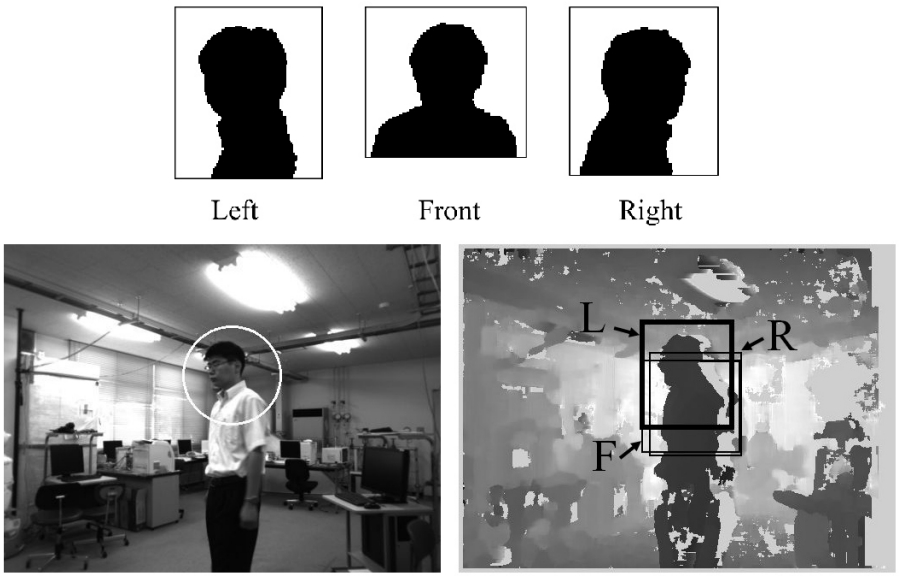


Figure 4.11: Depth templates and human detection result in depth images [61].

Spinello and Arras [62] describe a human detection system based on HOGs and stereo vision. Therefore, the authors introduce the concept of Histograms of Oriented Depths (HODs). The HOD concept is similar to HOG, except that it is applied to depth instead of intensity gradients. Furthermore, they evaluate a combined approach using HOG and HOD feature descriptors for RGB-D data. The algorithm is able to detect humans with a correct classification rate of 85% at a distance of more than 8 m. However, in order to detect people at 30 frames per second (fps), the algorithm has to be implemented on a Graphics Processing Unit (GPU).

Conversely, Munaro et al. [63] present a RGB-D based solution that runs at 26 fps without the need of a GPU. After removing the ground and 3D-clustering the data, a HOG-based people detection is performed on the remaining clusters. A SVM classifier is trained in the same way as described by Dalal and Triggs in [53]. Their tracking approach is based on an Unscented Kalman Filter.

4.3 Laser-Based Approaches

Laser-based approaches utilize range data for the detection and tracking of people. In contrast to vision-based approaches, laser-based approaches provide very accurate range information, while lacking intensity information. Resulting difficulties are similar to vision based approaches: Object occlusion, articulated nature of objects, complex object motions, real time requirements and ego motion. This is aggravated by the so called LIDAR shadowing. However, laser measurements are not influenced by illumination changes.

4.3.1 2D LIDAR Approaches

2D LIDAR approaches utilize one or multiple line scanners. Therefore, common features as described in Section 4.1 can not be extracted. Instead, depending on the sensor configuration, other features have to be found.

Taipalus and Ahtiainen [64] use a knee-high mounted 270° Field of View (FOV) LIDAR on a mobile robot in order to detect and track a human. After clustering the received laser scans, a heuristic is used to identify human legs within the clusters. A similar approach was described by Arras et al. in [65].

Another approach described by Carballo et al. [66] uses two layers of laser line scanners covering a 360° FOV. The lower layer detects legs, while the upper layer is used to detect the human chest (see Figure 4.12). Thus a more reliable detection is possible by combining both layers.

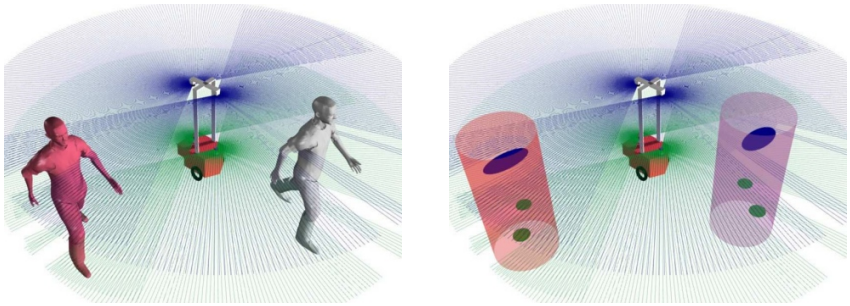


Figure 4.12: Two humans in multi-layered approach and corresponding detection result from [67].

Furthermore, Carballo et al. introduced a novel feature for laser based detection in [67]. Utilizing laser intensities as features, they improved their multi-layered approach significantly.

4.3.2 3D LIDAR Approaches

3D LIDAR approaches are based on 3-dimensional range measurements. The measurements can be collected with 3D LIDARs or moving 2D LIDARs. Again, not all of the features described in Section 4.1 can be extracted. Nevertheless, similar principles are applied in different publications.

Spinello et al. introduce a novel person detection method for 3D point clouds in [68]. The point clouds are collected with a Velodyne HDL-64E S2 laser scanner. Their detection approach is based on several highly specified AdaBoost classifiers based on multiple co-planar layers that are subdividing a human at different heights. Figure 4.13 illustrates the approach. The authors claim to achieve a classification rate of up to 96% based on a single 3D scan.

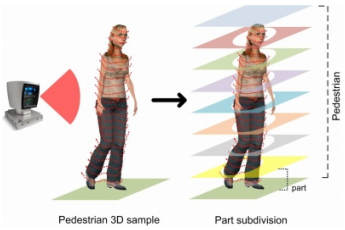


Figure 4.13: A 3D scan of a person is divided into K layers. For each layer an AdaBoost classifier is trained [68].

Teichman et al. suggest to identify objects in dense 3D data based on previously recorded and labelled object tracks [69]. Again, the data set is obtained by a Velodyne HDL-64E S2 laser scanner. For classification, a boosting framework is utilized. The suggested method runs in real-time with an accuracy of 98,5%. Furthermore, the system is able to classify between different objects: Pedestrians, bicyclists and cars.

In [70], Bohlman et al. implement a mobile robot platform that is able to track and follow a walking person. The authors utilize a 3D LIDAR with a resolution of 59x29 points. The human detection is based on ground segmentation and the extraction of human leg features based on cylindrical models. A Particle Filter is utilized for the tracking task.

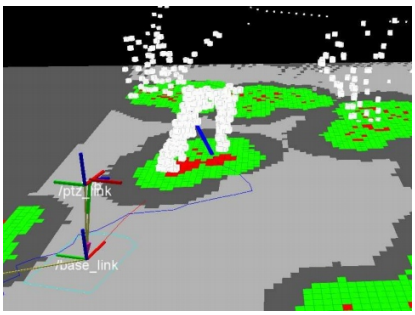


Figure 4.14: 3D LIDAR scan and detected human legs [70].

4.4 Hybrid Approaches

Hybrid approaches combine vision- and LIDAR-based approaches. The most popular setup is to use 2-dimensional laser scanners in combination with cameras.

In [47], Schiele et al. implement and evaluate a system based on a 2D SICK laser scanner and a camera. The camera is calibrated and used for human detection based on HOGs. For classification, a SVM classifier is used. The laser scanner is mounted about 30 cm above the floor and used for leg detection. Fusing both detection methods improves the overall correct classification rate. Figure 4.15 shows a typical false classification by the visual system and the corrected classification.

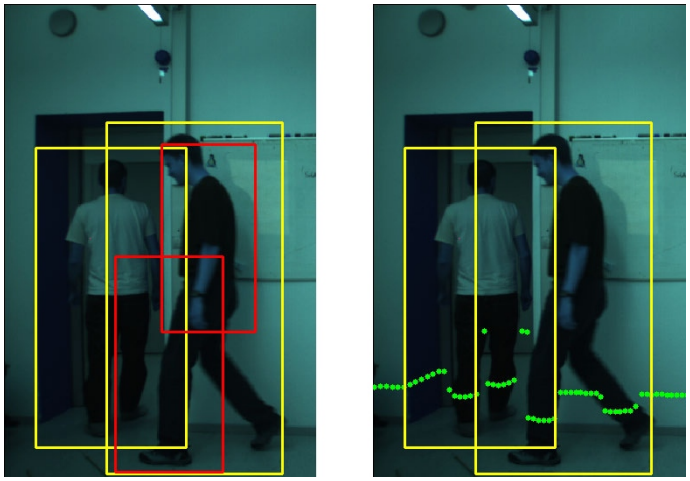


Figure 4.15: Hybrid approach for human detection: Misclassification by the visual system and the corrected classification by sensor fusion [47].

Bellotto and Hu combine a 2D SICK laser scanner with a Pan-Tilt-Zoom (PTZ) camera in order to follow a human with a mobile robot [41]. Similar to [47], the laser is used for leg detection. The camera, however, is mounted on the robot at approximately 1.5 m above the floor and used to detect human faces based on a set of Haar-like features. Both detection results are then combined using a sequential implementation of an unscented Kalman Filter.

Premebida et al. [71] combine a 2D SICK laser with a camera, too. Their goal is to detect and track pedestrians and vehicles. Therefore, entities of interest are extracted from the laser scan and classified using a Gaussian Mixture Model classifier. In addition to that, the position of each extracted entity is used to define a region of interest within the image. Subsequently, each region of interest is classified using an AdaBoost classifier based on Haar-like features. The classification results of both, the lidar and the camera classifier, are then combined using an Bayesian Classifier.

5 System Concept

The purpose of this chapter is to introduce the conceptional ideas of the developed system and its components. Therefore, first, principle system requirements which result from current state-of-the-art vehicles are described. Subsequently, the overall system design is presented. Finally, the individual hardware components of the system and their tasks, as well as important software frameworks that were frequently used for implementing the system are explained.

5.1 System Requirements

Comparing state of the art vehicles for different applications shows that full solutions for AATTVs are restricted to military applications. Nevertheless, various civil and space application scenarios could also benefit from AATTVs. This applies in particular to vehicles that provide easy-to-use interfaces.

A popular, user-friendly approach allows the robot to follow a human user autonomously. A major challenge implementing such an interface is to detect and track the user continuously using the robots perception. The perception can be based on visual information or laser measurements. A combination of both information sources is also possible.

However, most of the existing detection and tracking solutions are not suitable for the operation in rough terrain. Pure vision-based approaches suffer from illumination changes, while 2D laser-based approaches are not advisable in steep terrain due to their limited the vertical FOV. Consequently, 3D laser-based approaches seem to be most promising. They are independent of illumination changes and provide usually a vertical FOV of at least 30°. Furthermore, they have a large horizontal FOV.

Nevertheless, 3D LIDAR approaches lack the possibility of a reliable user identification. To overcome this problem, LIDAR data should be combined with a vision-based information. Combining various information sources is usually linked to a high computational effort. However, in order to follow a human, the robot's perception and control has to operate at a speed that is reasonable compared to the moving speed of the user. Consequently, the system has to operate in real-time.

All in all, the system that is going to be developed has to meet the following requirements:

1. Identify and track the human user
2. Follow the human user
3. Operate in a real-world environment
4. Operate in real-time

5.2 System Overview

Figure 5.1 gives an overview of the complete AATTV system.

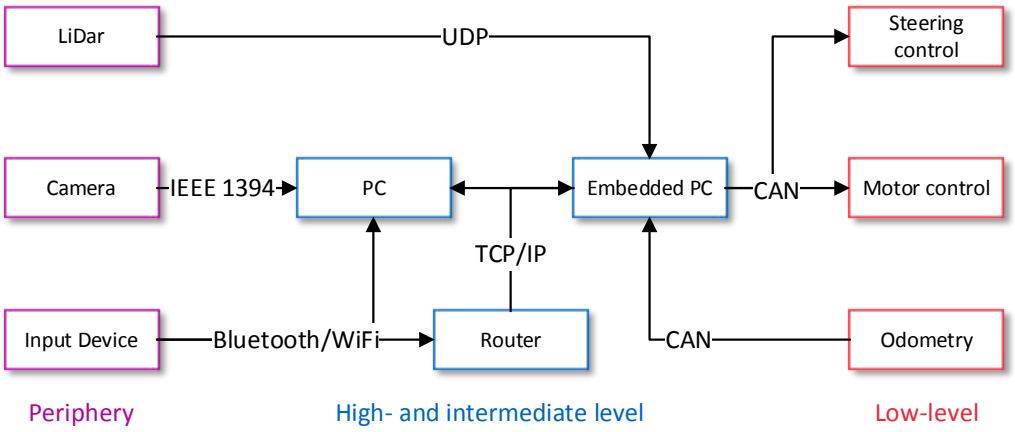


Figure 5.1: System overview and used communication protocols between periphery, high-, intermediate-, and low-level components.

In order to implement a human tracking and following robot, a camera, a 3D LIDAR, and other input devices like a game controller, a Bluetooth headset or a smart phone are required.

Thereby, the camera and the 3D LIDAR are utilized to determine the user location relative to the robot, while the other input devices can be used to activate the tracking and following behavior of the robot. The information provided by the peripheral devices is gathered and subsequently processed by both PCs, increasing the overall computation time. Once the perceived data is processed and the user location is determined, speed and steering commands are computed and sent to the according control units of the vehicle using the embedded PC's Controller Area Network (CAN) interface.

Both computers, as well as the user input devices communicate over an internal network. A standard network setup based on a router allows to connect additional hardware to monitor the system.

Both, the low-level components as well as the CAN interface between high- and low-level components were developed in other projects. The focus of this thesis project is therefore on the high-level system implementation. This includes to interface all peripheral devices, the setup of network structures as well as the implementation and evaluation of a suitable tracking and control algorithms.

5.3 Hardware

This section describes the hardware components used to implement the system: The Velodyne HDL-32E LIDAR scanner, the Bumblebee XB3 stereo camera, the ATV, and additional hardware used to create an interface between the robot and the user.

5.3.1 Velodyne HDL-32E

As 3D LIDAR scanner, the Velodyne HDL-32E shown in Figure 5.2 is used [72]. The HDL-32E emits 32 laser beams providing a vertical coverage from +10° to -30° with a horizontal FOV of 360°. The operating frequency can be selected by the user sampling up to 10 scans per second. Hereby, each scan consists of approximately 70.000 points with an error of less than 2 cm per measurement. The scans are transmitted using User Datagram Protocol (UDP) packets over a 100 Mbit/s Ethernet connection . The sensor additionally provides a Global Positioning System (GPS) receiver interface and an integrated Inertial Measurement Unit (IMU). The information of both can be collected using a separate UDP port.

Within this system, the Velodyne sensor is used to determine the 3D location of the user.

Figure 5.2: The Velodyne HDL-32E LIDAR scanner.

5.3.2 Bumblebee XB3

The camera used is the Bumblebee XB3 from Point Grey [73] as shown in Figure 5.3. The Bumblebee XB3 is a stereo camera with two different base lines. It operates at a frequency of up to 16 Hz and a maximum data rate of 800 Mbit/s using the IEEE-1394b interface (Firewire). It can capture and transfer three Bayer pattern encoded images simultaneously with a resolution of 1280×960 pixels.

Although the camera was originally intended to compute 3D images, only 2D images from one lens are used to distinguish the user from other persons. This is mainly because of the high computational effort required to calculate 3D depth images. Furthermore, the Velodyne LIDAR provides more accurate depth data.

Figure 5.3: The Point Grey Bumblebee XB3 with two different base lines.

5.3.3 Polaris Ranger EV

The ATV platform is based on the Polaris Ranger EV – an electric vehicle with Ackermann steering and 4 wheel drive. It has a vertical step of 25 cm and can transport a payload of up to 450 kg. Currently, the ATV can be operated in manual and automated mode. While the manual mode allows the ATV to be driven in the same manner as a regular car, it can be controlled in the automated mode using speed and steering angle commands. In order to control the electric vehicle in automated mode, two control units can be interfaced using CAN. One unit is responsible for controlling the steering motor as well as for reading the steering encoder and the four wheel encoders. The other control unit is used to command the four wheel motors. Additionally, a standard WiFi router, an embedded computer as well as a regular computer for high-level tasks are mounted. Figure 5.4 shows the ATV with attached sensors.

Figure 5.4: The Polaris Ranger ATV with mounted Velodyne HDL-32E, Bumblebee XB3 and perception computer.

5.3.4 Additional Periphery

An additional peripheral device is required for two reasons. First, an additional input device provide another level of control when activating and using the human-robot interface. Therefore, it is above all a security feature. Second, the device can be used to assure that only the user activates the interface, since only he has access to this hardware. Thus, it is also a way of identifying the user.

A PS3 controller, a standard Bluetooth headset and an Android OS smart phone were used as additional periphery as shown in Figure 5.5. The devices can be combined or used separately. All devices provide a full interface solution. However, it is also possible to use them in any user desired combination.

(a) PS3 Controller (b) Samsung HM1100 (c) HTC One

Figure 5.5: Additional input devices.

5.4 Software

Different software frameworks and libraries were used to implement the system. The three main contributors are the Robot Operating System (ROS), the Point Cloud Library (PCL), and the Open Source Computer Vision Library (OpenCV).

5.4.1 Robot Operating System

The developed software system is based on the open-source Robot Operating System (ROS), which is currently developed by Willow Garage [74]. ROS is not an operating system in the traditional sense, but rather a meta-operating system that implements a communication layer which allows collaboration within a heterogeneous computer cluster. Nevertheless, ROS also provides typical operating system services such as hardware abstraction, low-level device control and inter-process messaging. Furthermore, ROS contributes a great number of software frameworks implementing commonly used functionality for robots such as path planning, obstacle avoidance, machine perception, localization and mapping, and data visualization.

ROS is language- and platform independent. It incorporates three main and various experimental client libraries. The main client libraries provide `C++`, `Python`, and `Lisp` support and are targeted to UNIX-like systems. The most spread, experimental library provides a pure `JAVA` implementation with Android OS support.

In order to use and work with ROS, it is important to understand how the communication layer is designed. Therefore, the ROS concepts of *nodes*, *master*, *messages*, *topics* and *services* are explained in more detail.

Nodes Within ROS, nodes are processes that perform some kind of computation. The main idea is to increase the software modularity and to distribute the computational burden on different threads, each performing a specific task. The tasks range from collecting and processing sensor data to the control of actuators and the execution of planning algorithms. A computational graph allows the nodes to communicate with each other.

Master The computational graph is set up by the ROS master. It allows nodes to connect to each other by registering new and looking up existing nodes. Furthermore, the master manages the so called *parameter server*, which is used to update and configure node parameters during run time.

Messages The communication itself is based on messages between the nodes. Messages are a typed data structures. Primitive data types such as integer, floating point, and Boolean as well as arrays of primitive types are supported. A previously defined message can also be used as a type within another message.

Topics In order to identify messages they are sent under a certain name, the so called topic. Topics are implemented based on the publish and subscribe paradigm. In order to send a message, it has to be *published* by a node to a certain topic. If another node wants to receive this specific message, it has to *subscribe* itself to the according topic. Multiple nodes can publish and subscribe to the same topic, while a single node can also publish and subscribe to multiple topics.

Services While topics provide a one-way interaction only, services expand the same concept to a request and reply interaction. Thus, a node can send a *request* message to another node which is offering a service. Once the request is received and processed by the service offering node, a *reply* message is sent back in return. Therefore, services can be thought of as a remote procedure call.

5.4.2 Point Cloud Library

The Point Cloud Library (PCL) is an open source software framework implementing a set of `C++` libraries for processing multi-dimensional point clouds [75]. In this context, a point cloud is a group of points, each encoding a certain set of information such as its three dimensional position, its RGB color or laser intensity. They are usually obtained using depth cameras, stereo vision or three dimensional LIDARs.

The PCL framework provides a numerous collection of state-of-the art algorithms and can be, for example, used to solve segmentation, noise filtering, model fitting and registration problems.

5.4.3 Open Computer Vision

The Open Source Computer Vision Library (OpenCV) is also an open source software framework. It is a cross-platform framework supporting `C++`, `C`, `Python`, `JAVA` and `MATLAB` interfaces [76]. OpenCV provides speed optimized algorithms for computer vision and machine learning tasks. The algorithms include simple image manipulations as well as more complex operations such as face recognition, object identification and automatic feature extraction. Furthermore, it provides algorithms for camera calibration, image rectification and the computation of depth images from a stereo camera setup.

6 Implementation

Figure 6.1 gives an overview of the implemented software system.

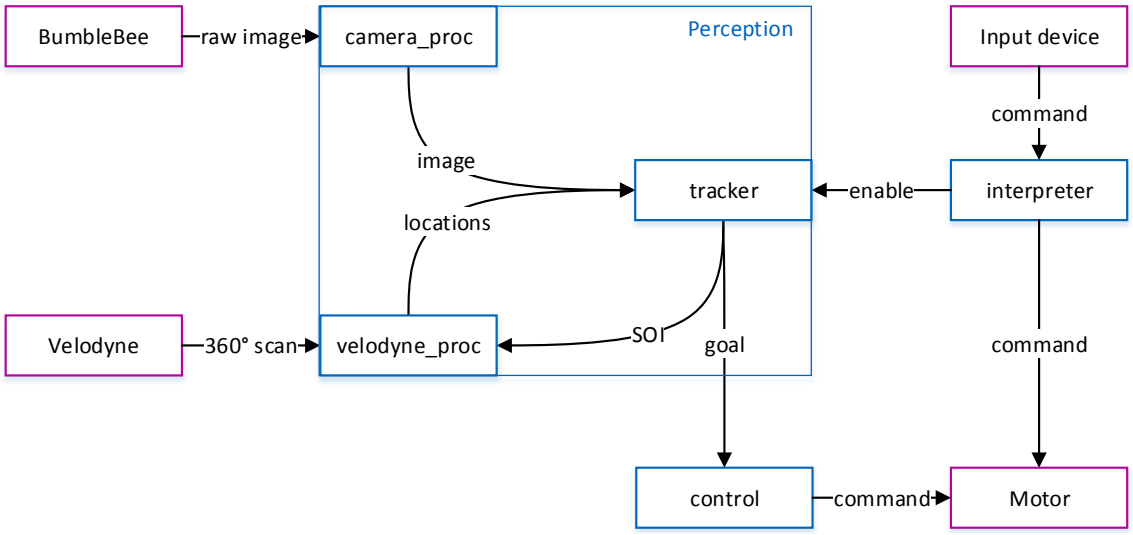


Figure 6.1: Software implementation overview: The purple and blue boxes indicate interfaces to the hardware and processing nodes, respectively.

The tracking behavior can be commanded using the *interpreter* node described in Section 6.1. This node implements a common interface for all input devices. As input device, a PS3 controller, a Bluetooth headset or a smart phone can be utilized.

The 3D data collected by the Velodyne, as well as the images captured by the Bumblebee are processed in parallel. Full 360° scans are collected from the Velodyne using the Velodyne ROS driver. Subsequently, each scan is processed in the *velodyne_proc* node. First, a Sector of Interest (SOI) is selected during the preprocessing step (see Subsection 6.2.1). Once the SOI is extracted, the ground is estimated and removed in the segmentation step, explained in Subsection 6.2.2. Finally, the remaining point cloud is clustered in order to find the groups of points that match the geometric characteristics of a human. This last step is described in Subsection 6.2.3.

Simultaneously, the Bayer encoded images, which are received from the Bumblebee, are decoded and rectified using the *camera_proc* node. A more detailed description of this node is given in Subsection 6.3.1.

The goal of the *tracker* node is to determine the current SOI based on the true user location. Therefore, the tracker combines the information obtained from the Bumblebee with the information from the Velodyne. The rectified images are used to track the user based on a single frame and the initial user position. Subsequently, the obtained tracking results are combined with the cluster information that was extracted from the Velodyne data. A detailed explanation of the tracking algorithm and sensor fusion process is given in Section 6.4.

The *control* node computes suitable speed and steering commands to follow the human user. The applied control algorithm is described in Section 6.5.

6.1 User Interface

In order to manage the tracking behavior of the AATTV, a communication link has to be established between the user and the vehicle. For each of the three different devices listed in Subsection 5.3.4, an individual interface was implemented. Thereby, each interface communicates with a common message interpreter, allowing to use multiple input devices simultaneously. Thus, for example, it is possible to use the PS3 controller in combination with the Bluetooth headset. Furthermore, any ROS supporting system can be used as input device. Figure 6.2 shows the command flow through the interpreter. Depending on the peripheral device, feedback from other nodes can be directly used or has to be processed by the interpreter first.

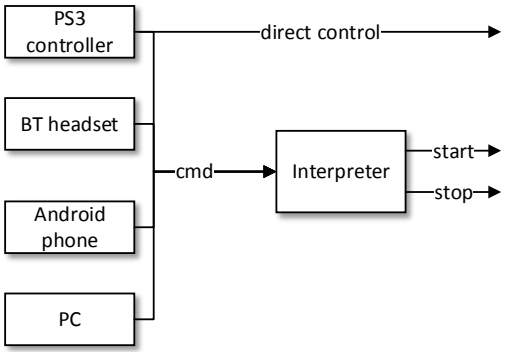


Figure 6.2: Communication flow from periphery to interpreter and system.

6.1.1 Controller Interface

Utilizing the ROS package *ps3joy* and a commercial Bluetooth 2.0 USB dongle, the PS3 controller could be easily integrated into the overall system. All necessary driver functionality is provided by *ps3joy*. In addition to the tracking management, the controller allows the user also to control the robot's speed and steering angle directly. A ROS node was implemented to publish according steering commands as well as ROS messages for the interpreter. Figure 6.3 shows the different functions assigned to the controller. The PS3 controller does not provide feedback.

Figure 6.3: The PS3 controller user interface: The X-button activates the tracking interface, the O-button deactivates the tracking interface, the right analog-stick controls the speed and the left analog-stick controls the steering angle.

6.1.2 Headset Interface

The headset interface allows the user to control the tracking interface based on voice commands. For speech recognition the ROS package *pocketsphinx* was used, a `Python` wrapper of the eponymous library developed at Carnegie Mellon University (CMU). The speech recognizer has to be configured to identify specific words and phrases using a so called knowledge base. The knowledge base can be easily created using CMU's Sphinx knowledge base tool [77]. Table 2 lists all supported commands.

Voice command	Action
Car, follow me.	activate interface
Car, stop.	deactivate interface
Car, halt.	deactivate interface

Table 2: Available voice commands for the headset user interface.

As an additional safety feature, the headset's button for call acceptance implements an emergency stop button. A `Python` script listens to the operating system's D-Bus and publishes a stop message to the interpreter.

In addition, the headset interface provides direct audio feedback that indicates whether a command has been correctly interpreted and executed. Therefore, the *interpreter* implements a text-to-speech synthesizing sound interface based on the ROS package *sound_play*.

6.1.3 Smart Phone Interface

As a third option, an application for Android smart phones and tablets was implemented. The app allows the user to start and stop the tracking interface of the ATV. Furthermore, it provides audiovisual feedback information from the robot and displays an image of the current tracking result. For audiovisual feedback, feedback messages are displayed and synthesized into speech. Figure 6.4 shows the graphical user interface. The application uses the *rosjava* client for Android.

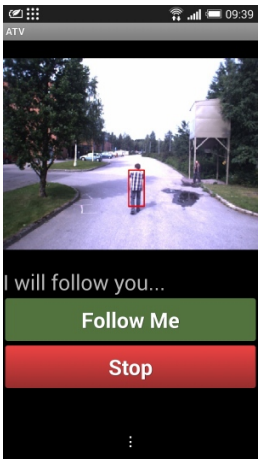


Figure 6.4: Graphical user interface of the Android application.

6.2 LIDAR Processing

The LIDAR data is retrieved using the ROS package *velodyne*. A full scan is shown in Figure 6.5. The collected data is processed in three steps. First, the Sector of Interest (SOI) is extracted based on the previous estimation of the user location. In the second step, the remaining data is segmented and the ground removed. Two different approaches for the ground segmentation were implemented in order to handle different terrain more efficiently. Finally, the remaining points are clustered.

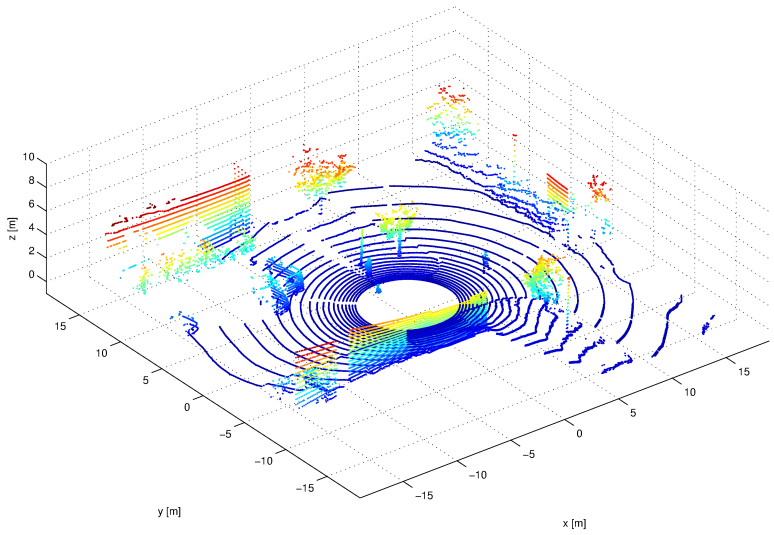


Figure 6.5: A full 360 ° scan from the Velodyne HDL-32E.

6.2.1 Preprocessing

In the preprocessing step, the SOI is extracted from the incoming scans. The SOI is based on the angle of the last known user position or straight in front of the AATTV if the user interface is initialized. The sector covers approximately one eighth of a full scan. This preprocessing step is necessary in order to execute the following processing steps at a rate of 10 Hz. Figure 6.6 shows the result of the preprocessing step.

A difficulty within this step is that the point clouds provided from the ROS Velodyne driver cover more than 360°. Consequently, the angle of the first element within each point cloud changes over time with respect to the car's reference frame. However, the index of the point that is corresponding to the user's direction can be quickly found using a simple binary search. Once that index is found, the sector can be constructed around that center point.

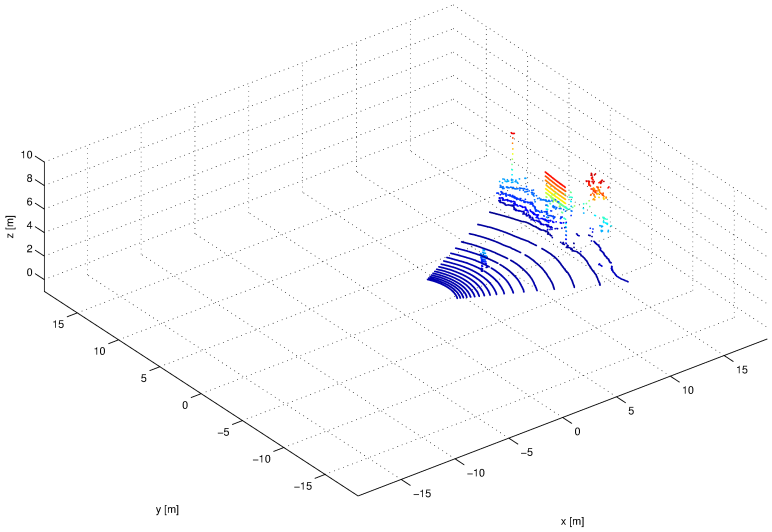


Figure 6.6: The selected sector of interest.

6.2.2 Segmentation

The goal of the segmentation step is to extract the ground floor within the SOI as shown in Figure 6.7. This step is essential in order to be able to distinguish between different objects based on their connectivity within the point cloud. Two different algorithms were implemented. The first algorithm assumes a planar ground model and is therefore limited to a certain kind of environment. The second approach constructs a more general ground model at the cost of a higher computation time.

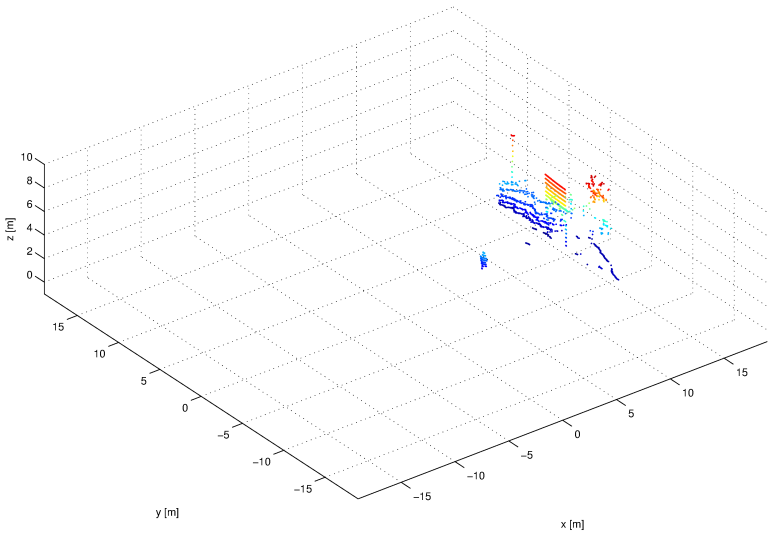


Figure 6.7: the ground segmentation result.

Random Sample Consensus The Random Sample Consensus (RANSAC) based ground segmentation tries to fit a plane model into the received LIDAR data. Once an appropriate model is found, all points matching the plane model within a given threshold will be removed. This is repeated until the original point cloud size is halved or the estimation of the plane model fails.

In order to fit a model to the received data, points are randomly picked to construct a plane. Based on this model a so called consensus set is identified. The consensus set is defined by the number of points that match the constructed model within a given threshold. If the set is sufficiently large enough or the maximum of allowed iterations is exceeded the algorithm terminates. If, however, the found consensus set is too small, the algorithm starts over again by selecting new points in order to construct the model. Figure 6.8 illustrates the result for a single slice within a 3D scan.

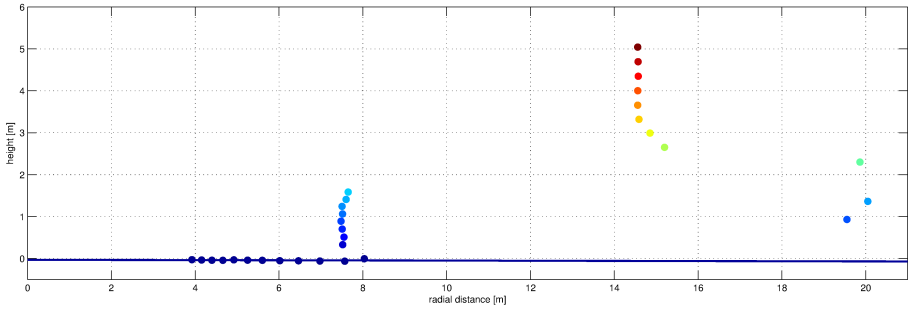


Figure 6.8: RANSAC ground estimation result for a slice of the 3D scan. The dots represent the LIDAR measurements, while the estimated plane is indicated by the blue line.

The RANSAC algorithm was realized using PCL's *SACSegmentation* class. The class allows to configure various parameters shown in Table 3.

Parameter	**Value**	**Description**
Method type	RANSAC	random sample consensus
Axis	$[0, 0, 1]$	z-axis
Model type	perpendicular plane	extract a plane perpendicular to axis
EpsAngle	$10°$	max. plane inclination
Max. iterations	100	max. number of iterations
Distance threshold	0.10 m	model threshold

Table 3: PCL's SACSegmentation configuration parameters.

Gaussian-Process-Based The second approach implements a Gaussian-Process-Based (GPB) real-time ground segmentation introduced by Tongtong et al. [78]. The proposed algorithm reduces the complexity of the complicated 2D ground segmentation to many simple 1D Gaussian process regression problems and allows therefore to be used in real-time applications.

The algorithm itself consists out of six major steps, which will be explained in the following sub-paragraphs.

1. Polar Grid Map In the first step, a polar grid map representation of the incoming scan is created. Points that exceed a defined maximum distance $R_{max} = 20\,\text{m}$ are ignored. Figure 6.9 shows a polar grid map consisting of $M = 180$ segments which are each separated into $N = 100$ bins.

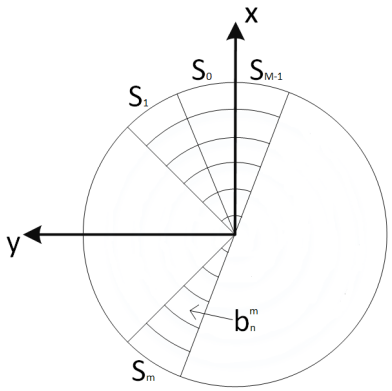


Figure 6.9: A polar grid map consisting of M segments with N bins in each segment (adapted from [78]).

Each point p_i of the incoming 360° scan is assigned to one bin in one of the polar grid map's segments. The segment number m for an incoming point is defined by:

$$m(p_i) = \left\lceil \frac{M \cdot \text{atan2}(y_i, x_i)}{2 \cdot \pi} \right\rceil \tag{1}$$

The according bin n for a point is given by:

$$n(p_i) = \left\lceil \frac{N \cdot \sqrt{x_i^2 + y_i^2}}{R_{max}} \right\rceil = \left\lceil \frac{N \cdot r_i}{R_{max}} \right\rceil \tag{2}$$

Additionally, a new subset of points PG_m is defined for each segment m, containing the 2D points p_i' which have the minimum height z_i within all points $P_{b_n^m}$ in bin n:

$$PG_m = \left\{ p_i' = (r_i, z_i)^T | z_i = \min(H_n^m), n = 1...N \right\} \tag{3}$$

where H_n^m is defined as:

$$H_n^m = \left\{ z_i | p_i \in P_{b_n^m} \right\} \tag{4}$$

2. Line fitting In the second step, the set PG_m is utilized to calculate the length-scale parameters l_i which are later used to construct a model of the ground for each segment. Therefore, for every segment m lines are extracted based on the points in PG_m using the incremental line fitting algorithm presented in [79]. Lines with a gradient bigger than 10° are omitted. Subsequently, each input r_i is associated with the gradient $g(r_i)$ of the closest line. The closest line is defined by that line with the minimum distance between the input point and the endpoints of the line. Finally, based on $g(r_i)$ the length-scale parameter l_i can be computed for each input using the following equation:

$$l_i = \begin{cases} a \cdot \lg\left(\frac{1}{|g(r_i)|}\right) & \text{if } |g(r_i)| > g_{def} \\ a \cdot \lg\left(\frac{1}{|g_{def}|}\right) & \text{otherwise,} \end{cases} \tag{5}$$

where a is a scaling parameter that can be trained, and g_{def} is a boundary to prevent flat ground causing infinite length-scales. Figure 6.10 shows the line fitting result as well as the calculated length-scale parameters within one segment m.

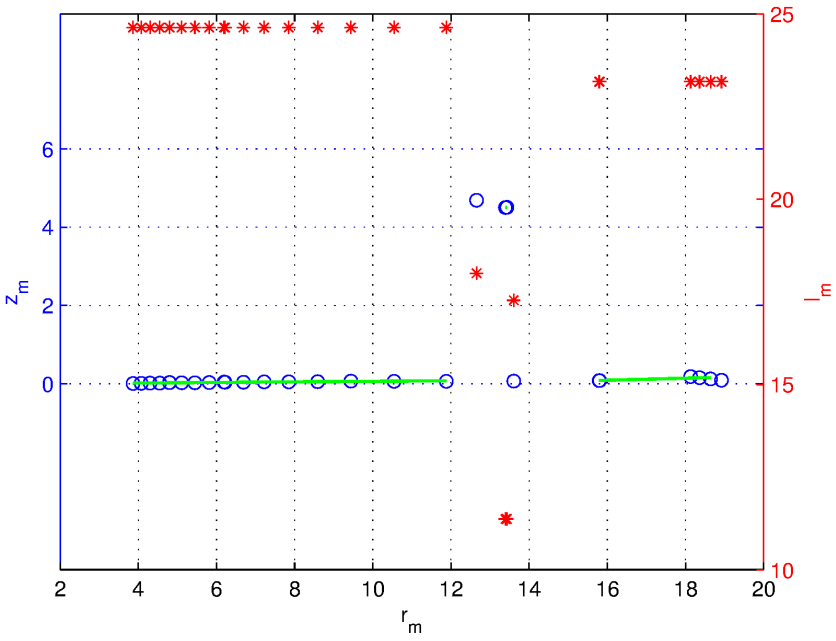


Figure 6.10: Line fitting results for one segment with scale parameter $a = 6.2978$. Red stars show the length-scales, blue circles represent the set PG_m and green lines indicate the incremental line fitting results.

3. Seed Estimation In the third step, initial seeds are estimated. All points within a certain distance $B = 10\,\text{m}$ that have an absolute height lower than the threshold $T_s = 0.30\,\text{cm}$ are selected as initial seeds.

4. Gaussian Process Regression The goal of the Gaussian process regression is to obtain a ground model. Therefore, the following equation is used to calculate the covariance matrix K between pairs of seeds (r_i, r_j):

$$K(r_i, r_j) = \sigma_f^2 \cdot \sqrt{\frac{2 \cdot l_i \cdot l_j}{l_i^2 + l_j^2}} \cdot \exp\left(-\frac{2(r_i - r_j)^2}{l_i^2 + l_j^2}\right) + \sigma_n^2 \cdot \delta_{ij}, \tag{6}$$

where σ_f^2 and σ_n^2 are hyper-parameters that have to be trained and δ_{ij} is a Kronecker delta.

5. New Seed Evaluation Based on the obtained model, all remaining points in the set PG_m that are no seeds yet are tested whether they fit into the previously obtained model or not. If the evaluated point is likely to be part of the ground, it becomes a new seed for the next model evaluation. Depending on the predicted mean height $\bar{z}_*$ and the covariance $V[z_*]$ a new point r_* can be classified as seed. The calculation of $\bar{z}_*$ and $V[z_*]$ is based on the predictive step of the Gaussian process regression:

$$\begin{aligned} \bar{z}_* &= K(r_*, R)K^{-1}Z \\ V[z_*] &= K(r_*, r_*) - K(r_*, R)K^{-1}K(R, r)', \end{aligned} \tag{7}$$

where $K \in \mathbb{R}^{s \times s}$ is the model matrix, $K(r_*, R) \in \mathbb{R}^{1 \times s}$ is a vector for the tested point and $R = [r_1, ..., r_s]^T$, $Z = [z_1, ..., z_s]^T$ are the old seeds s.

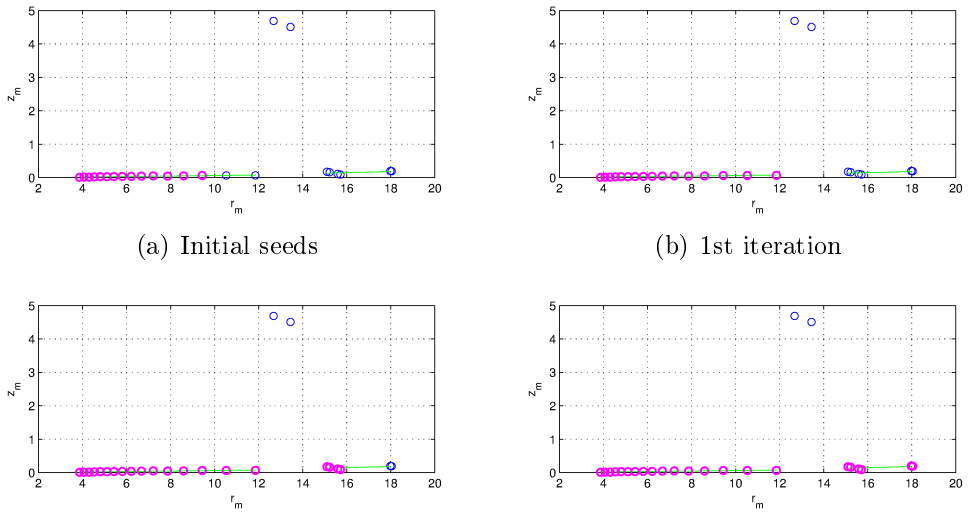
A new point is classified as seed if the following two evaluation criteria are fulfilled:

$$\begin{aligned} V[z_*] &\leq t_{model} \\ \frac{|z_* - \bar{z}_*|}{\sqrt{\sigma_n^2 + V[z_*]}} &\leq t_{data}, \end{aligned} \tag{8}$$

where t_{model} and t_{data} are threshold parameters. Figure 6.11 shows the seed selection and evaluation process for a single segment.

6. Point-wise Segmentation In the last step all points of the scan are classified depending on the estimated model ground height in each bin. The mean height for each bin can be estimated using Equation 7 and the mean radial distance of the bin. A point in the bin is classified as ground if its height is within a threshold T around the estimated height.

Training of hyper-parameters The performance of the algorithm depends mainly on the values of the hyper-parameters a, σ_n and σ_f. The authors of the original paper present a way to train those parameters based on labeled data sets. However, because this implementation sets up the polar grid map in the same fashion as in the original approach and because the ATV is supposed to work in a similar environment, the hyper-parameters suggested in the original paper show already good results. Therefore, a training of the hyper-parameters is not necessary. The output of the algorithm on a typical scan is shown in Figure 6.12.



(a) Initial seeds (b) 1st iteration

(c) 2nd iteration (d) 3rd iteration

Figure 6.11: Seed estimation based on Gaussian process regression. Per iteration the new seeds are selected until a good approximation of the ground is obtained.

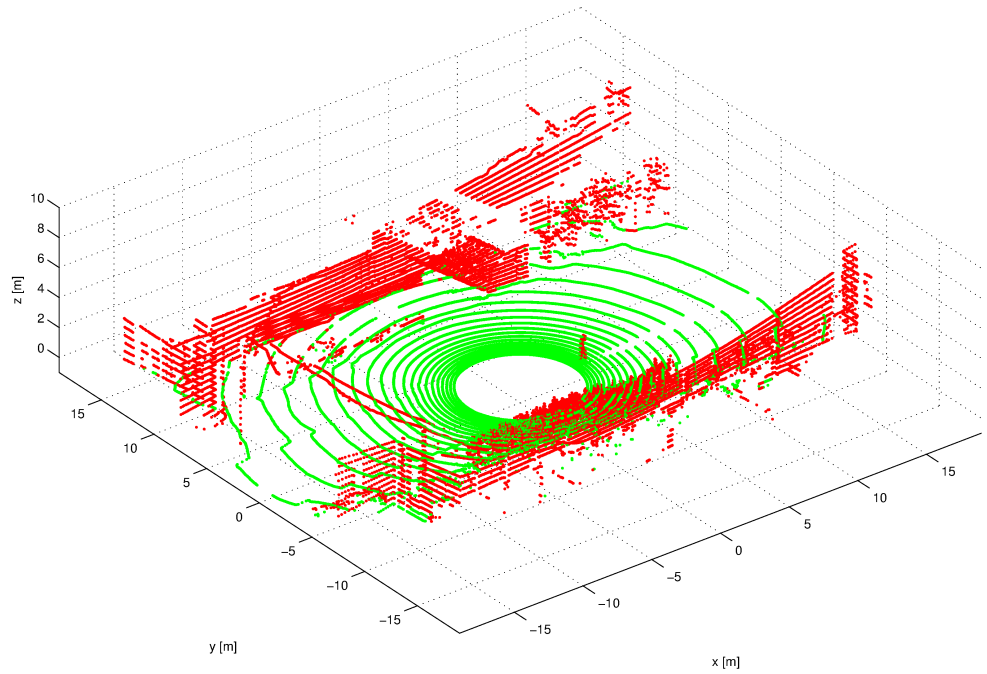


Figure 6.12: Gaussian-process-based ground segmentation result with $\sigma_f^2 = 0.0528$ and $\sigma_n^2 = 0.0012$.

6.2.3 Clustering

So far, a Sector of Interest (SOI) has been selected based on an initial guess of the user's position. Furthermore, the ground has been removed within the SOI. The goal of this third processing step is to divide the remaining points into groups and calculate their center of gravity as well as a 3D bounding box. The determined groups are than filtered according to different requirements. The requirements are based on the geometric characteristics of a human in the point cloud. In total, five requirements can be formulated:

1. The group consists of at least $N_{\min}$ points.
2. The group consists of at most $N_{\max}$ points.
3. The distance between adjacent points in the group is less than $d_{\max}$.
4. The group can be enclosed by a box with fixed dimensions h, w and d.
5. The group's center of gravity is smaller than a threshold T_h height h_c.

The first three requirements are evaluated using the *EuclideanClusterExtraction* class implemented in the Point Cloud Library. The algorithm is based on [80]. It assumes a point cloud represented in a Kd-tree as input and extracts clusters that fulfill the first three requirements. The parameters N_{min}, N_{max} and d_{max} depend on the mounting height and the vertical resolution of the Velodyne sensor. The parameters can be determined from sample scans as shown in Figure 6.13.

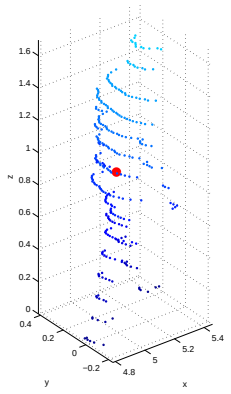


Figure 6.13: Scan of a human in 5 m distance consisting of 300 points. The average over all points is marked with a red dot.

Parameter	Value
N_{min}	50
N_{max}	1000
d_{max}	0.40 cm
h	2.00 m
w	1.20 m
d	1.20 m
h_c	1.50 m

Table 4: Point Cloud Library clustering parameters for a Velodyne HDL-32E mounted on the ATV in 2.30 m height.

Each cluster is then tested for the requirements four, and five. In order to check the bounding box requirement, PCL is used to find the minimum and maximum values of each cluster in x, y and z direction. If the difference in each direction between the maximum and the minimum value is smaller than the according box dimensions, the requirement is fulfilled. For the last requirement the center of gravity of all points is calculated and its height compared to the reference height h_c. Table 4 summarizes all clustering conditions for the Velodyne mounted on the ATV.

6.3 Camera Processing

The images from the Bumblebee XB3 are captured using the ROS driver node *camera1394*. The driver can be used to interface any camera supporting the IEEE-1394 interface. Although the camera is originally intended to provide depth information based on stereo images, only the center camera is used to capture images. This is simply because the computation time, which is needed to compute stereo images, is too high and therefore in conflict with the real-time requirement formulated in Section 5.1. Furthermore, the measurements from the Velodyne are collected at a higher frequency and are more accurate.

6.3.1 Decoding and Rectification

The camera calibration is based on the factory calibration which can be read from the camera itself. The factory calibration data provides the intrinsic camera parameters as well as two matrices to remap each pixel location in order to obtain an undistorted image. The camera matrix is given by:

$$K_{\text{XB3}} = \begin{pmatrix} 997.8547 & 0 & 644.2970 \\ 0 & 997.8547 & 489.7782 \\ 0 & 0 & 1 \end{pmatrix} \tag{9}$$

Both, the decoding as well as the rectification are realized using the OpenCV library. The decoding is implemented using the *cvtColor* function, converting raw Bayer GBRG encoded images into 3-channel RGB images. After the image is decoded, it is rectified using the *remap* function with the factory calibration matrices. The results of this processing step are shown in Figure 6.14.

Figure 6.14: Image decoding and rectification result.

6.4 Sensor Fusion

The sensor fusion step combines the processed Velodyne information with the information obtained from the image processing. However, in order to fuse the 3D locations with the 2D images, a precise calibration between the Velodyne and the Bumblebee is necessary. The external calibration is described in Subsection 6.4.1. Additionally, the efficiency of the camera tracker depends strongly on the quality and illumination characteristics of the image. Therefore, an automatic shutter adjustment as described in Subsection 6.4.2 is needed. The final fusion algorithm is described in Subsection 6.4.3.

6.4.1 External Calibration

The external calibration between the Velodyne and the Bumblebee XB3 was determined manually. Therefore, a graphical user interface was implemented to display and project the LIDAR data into the rectified image (Figure 6.15). The projected scan is color coded in order to illustrate the measured depth in the 2D image. By adjusting the Euler angles as well as the translational offset between the camera and the LIDAR frame a simple and fast calibration based on an initial guess is possible. The final projection matrix used to transform homogenous LIDAR coordinates into homogenous image coordinates was found to be:

$$F = \begin{pmatrix} 646.2073 & -996.6125 & -3.5120 & -70.8727 \\ 488.3785 & 3.5500 & -998.5342 & -203.5538 \\ 1.0000 & 0.0019 & -0.0014 & -0.1100 \end{pmatrix} \tag{10}$$

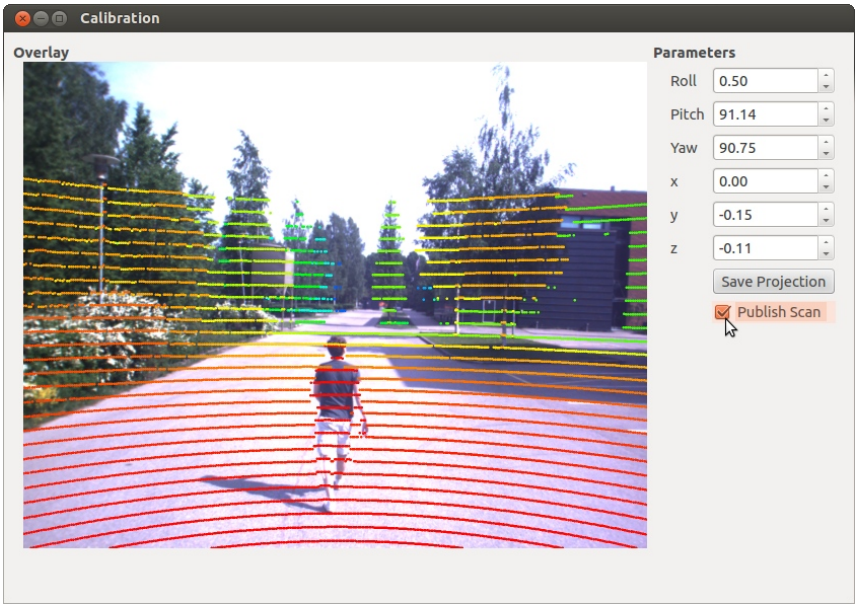


Figure 6.15: The Graphical user interface to determine parameters for external calibration. The final projection matrix can be saved as .yaml-file. Additionally the user interface allows to publish a colored point cloud.

6.4.2 Shutter Adjustment

Working with cameras, ubiquitous illumination changes are a major downside in outside environments. Usually, this problem is tackled by varying the camera's shutter time in order to increase or decrease the illumination time dynamically. However, if the camera is used to track any kind of object, a constant illumination is crucial to keep track of the object. In the case of feature tracking algorithms, the illumination time has to be selected in such a way that at least a minimum number of features can be extracted and tracked. Therefore, a simple proportional controller is implemented to adjust the shutter and therefore the illumination time automatically.

The goal of this controller is to change the illumination time by adjusting the shutter time in such a way that a minimum number of features can be extracted from the object of interest. It is therefore assumed that the number of features that can be extracted from the object of interest is proportional to the contrast that is provided by the object itself. This assumption is illustrated in Figure 6.16.

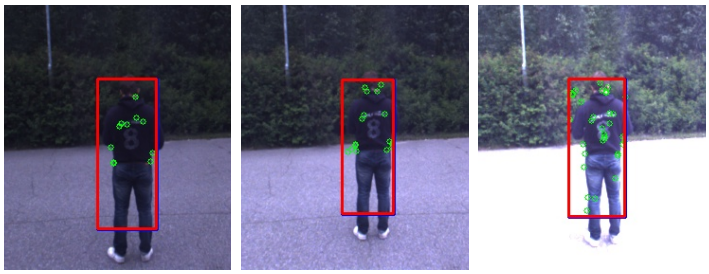


Figure 6.16: Relation between number of extracted features and the contrast of the object for a certain object of interest. The number of features is increasing proportionally to the image contrast.

The controller adjusts the shutter time until a certain minimum threshold of extracted features is reached. The input of the proportional controller is based on the difference between the average pixel intensity within the selected region of interest and a constant value. This value can be set to half the maximum intensity, under the assumption that the targeted object provides a high-contrast texture with a different intensity than its surrounding background. Figure 6.17 illustrates the applied control algorithm.

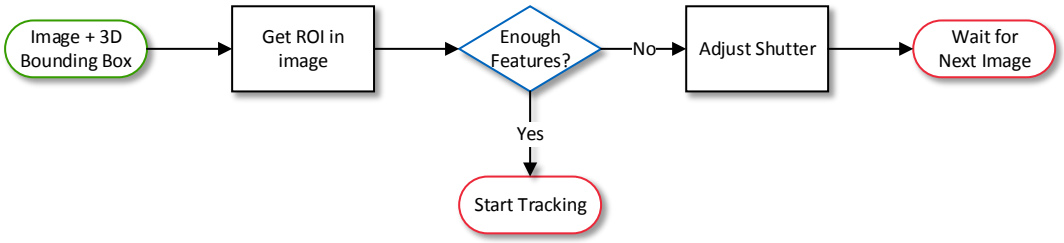


Figure 6.17: Automatic adjustment of the shutter time.

6.4.3 Hybrid Tracking

In order to get a reliable tracker, both information sources are combined to overcome the drawbacks of each source. While the detection of objects based on the LIDAR data is robust against variations in rotation and shape of the detected object, it fails if multiple targets are very close to each other and thus can not be identified as different objects. In contrast to that, the suggested camera tracking algorithms are capable of differentiating between objects that are close to each other. However, due to environmental illumination changes, a long time, reliable tracking performance is almost impossible to achieve, especially if non-rigid objects are considered.

Therefore, the proposed algorithm takes situation-dependent different information sources into account. Depending on the number of possible objects detected in the LIDAR scan, either a tracker based on the LIDAR data and a Kalman Filter is used solely or in combination with an additional image tracker. A state machine within the tracker decides which kind of information should be utilized. The state machine and tracking results for different states are shown in Figure 6.18 and 6.19, respectively.

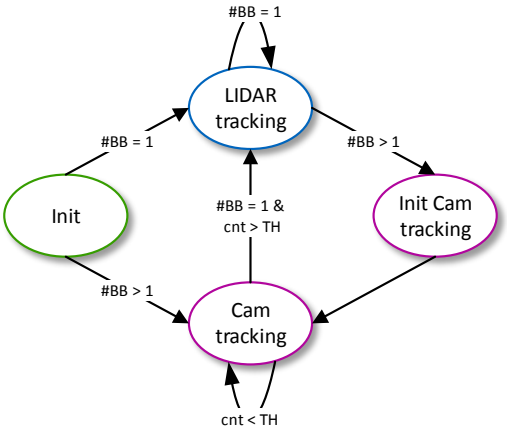


Figure 6.18: Finite state machine for the hybrid tracker: $\#BB$ describes the number of bounding boxes received from the LIDAR processing, cnt is a counter that counts how many single bounding boxes were received in a row and TH is a positive threshold.

(a) LIDAR tracking (b) Init Cam tracking (c) Cam tracking

Figure 6.19: The different results for each state of the hybrid tracker.

Initialization After the tracking behavior of the robot is activated using any of the input devices described in Section 6.1, the hybrid tracking algorithm is initialized. Initially, the shutter time is adjusted as described in Subsection 6.4.2. Subsequently, the x and y coordinate of the center of gravity of the detected object that closest to the center in front of the car, are used to initialize the Kalman Filter. Therefore, the user has to stand in the center in front of the vehicle when the interface is activated. The filter is defined by the following transition equation:

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot X_k + w_k, \tag{11}$$

where x and y are the 2D position, $\dot{x}$ and $\dot{y}$ are the derivatives of x and y, dt is the time interval between two state transitions and w is the process noise. The measurement equation is given by:

$$Z_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot X_k + v_k, \tag{12}$$

with v as the measurement noise. Depending on the number of detected bounding boxes, the person is either tracked using the LIDAR data and the Kalman Filter only or by relying on the camera data and image tracking result. If no valid user location can be found, the Kalman Filter propagates itself based on its last prediction. If, however, the user can not be re-detected within 2 s, the vehicle is forced to stop.

LIDAR Tracking This state is active as long as only one object is detected in the LIDAR data. The x and y coordinate of the center of gravity of the detected object is compared to and verified by the prediction of the Kalman Filter. If the prediction and the actual measurement coincide within a certain threshold, the received 2D position is used as input for the steering control, which is described in Section 6.5.

Init Camera Tracking If, however, more than one possible object is detected, the prediction of the Kalman Filter is used to determine the current target. Subsequently, a region of interest is selected in the image by projecting the 3D bounding box of the target into the image frame. This region of interest is then used to initialize one of the image trackers described in Subsection 4.1.3. The center of the corresponding bounding box is forwarded to the vehicle controller and used to update the Kalman Filter.

Camera Tracking As long as the camera tracking is active, all results from the LIDAR tracking are compared to the image tracking result. Therefore, all determined 3D bounding boxes are projected into the image and the overlapping area between each projected bounding box and the image tracking result are computed. The projected bounding box with the biggest, realtive overlap is assumed to be the correct target. Again, the determined location is then forwarded to the vehicle

controller and used to propagate the Kalman Filter. Additionally, if only one object is detected in the LIDAR data, a counter is incremented. This counter is set back to zero if two objects are detected again. Once the value of this counter exceeds a certain threshold, the tracker changes back into the LIDAR tracking state.

6.5 Speed and Steering Control

The goal of the speed and steering control is to calculate a suitable vehicle speed and a valid steering angle in order to follow the user. It is assumed that the user walks along a trajectory that can be followed by the vehicle. Therefore, complex maneuvers do not have to be performed and the turning radius of the vehicle does not have to be considered. Figure 6.20 defines the used coordinate system and illustrates the applied approach.

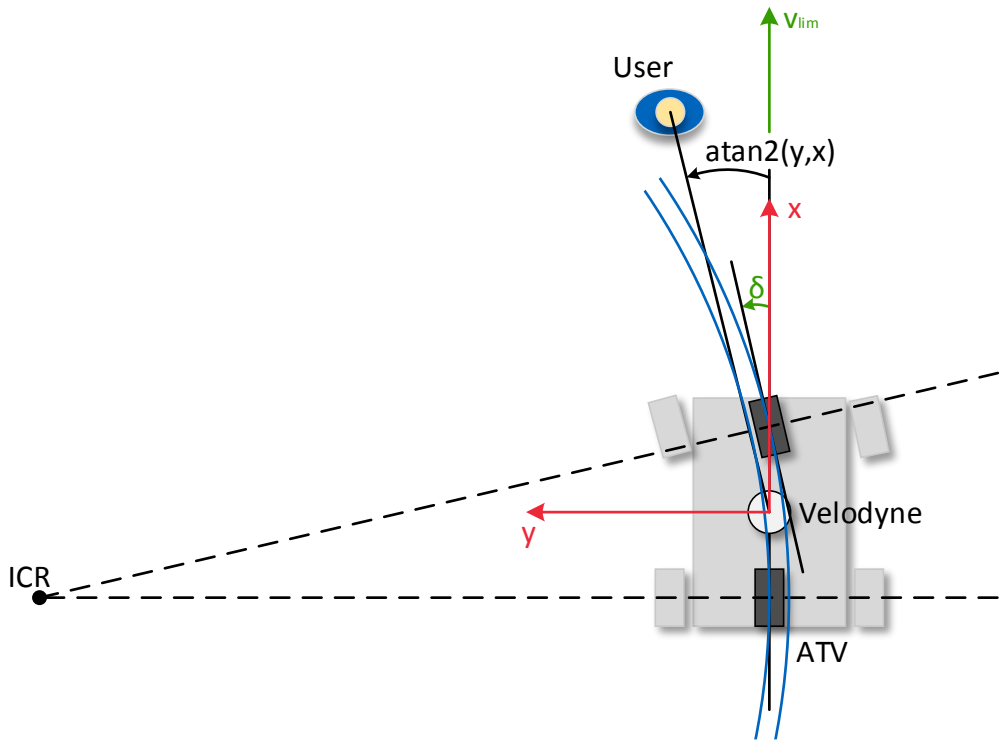


Figure 6.20: Vehicle control approach: The ATV is shown in light grey, and the applied bicycle model in dark grey. δ is the desired steering angle and defined as the angle between the virtual steering wheel and the positive x-axis. Together with the rear axis, δ defines the Instantaneous Center of Rotation (ICR). v_{lim} is the desired vehicle speed. The user position is obtained in the LIDAR frame that is aligned with the frame of the vehicle.

In this approach, the car-like vehicle is approximated using the bicycle model, consisting of two virtual wheels – the fixed and the steering wheel. The commands applied to the vehicle are calculated based on the 2D position of the user relative to the car. By changing his position, the user is able to control the vehicle. Using two simple controllers, the necessary steering angle δ as well as the required vehicle speed v_{lim} can easily be calculated.

The control loop for the ATV is shown in Figure 6.21. The Figure shows an open loop system since the user's location is measured relatively to the car. The algorithm is adapted from [81].

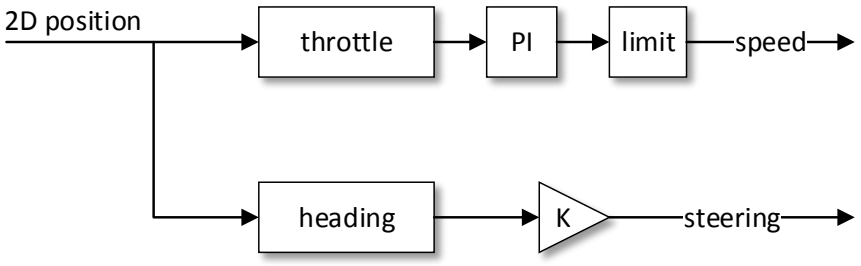


Figure 6.21: Speed and steering angle control for Ackermann steering.

The speed is controlled using a PI controller. As input $e(t)$ for speed control, the throttle is given as follows:

$$e(t) = \sqrt{x(t)^2 + y(t)^2} - \Delta, \tag{13}$$

where $[x(t), y(t)]^T$ is the 2D position at time t and Δ the minimum distance the car should keep. Thus, the speed v is calculated by:

$$v(t) = P \cdot e(t) + I \cdot \int e(t)dt \tag{14}$$

The output is limited between 0 m/s and 2 m/s in order to prevent driving backwards or faster than step speed, respectively. Additionally, the speed drops to 0 m/s if the vehicle gets closer than a certain threshold Δ_{min}. This is necessary since the brakes of the vehicle are mechanically not implemented yet. Consequently, the speed $v_{lim}(t)$ that is actually applied to the vehicle is given by.

$$v_{lim}(t) = \begin{cases} 0 & \text{if } v(t) < 0 \vee \sqrt{x(t)^2 + y(t)^2} < \Delta_{min} \\ 2 & \text{if } v(t) > 2 \\ v(t) & \text{otherwise} \end{cases} \tag{15}$$

The steering angle $\delta(t)$ is calculated based on the user's direction relative to the vehicle orientation:

$$\delta(t) = G \cdot \text{atan2}(y(t), x(t)) \tag{16}$$

7 Evaluation

The evaluation of the system can be divided into five parts. First, the three different user interfaces are compared. Second, both ground segmentation algorithms are evaluated by comparing the segmentation result in different scenarios. Next, the tracking behavior of the system is assessed. Therefore, first both image trackers are compared to each other and, subsequently, the one more suitable for the hybrid tracker is tested within different scenarios. In the fourth step of the evaluation, the control behavior of the system is analyzed. Finally, the overall system performance is verified during a real world scenario test.

7.1 User interfaces

A very important aspect when comparing the user interfaces is their ease of use. However, this is closely linked to the user's personal experience with each device and consequently difficult to evaluate. Therefore, the different user interfaces are compared to each other based on their versatility, their operating range, and reliability. Table 5 gives an overview of different characteristics for each interface.

	Android App	**Headset**	**PS3 controller**
Button input	✓	✓	✓
— *count*	2+	$1-4$	16
— *labeled*	✓	✗	✗
Voice input	✓	✓	✗
Interface	WiFi	Bluetooth	Bluetooth
— *Range*	> 20 m	< 10 m	< 10 m
— *Reliability*	high	low	low
Feedback	✓	✓	✓
— *Audio*	✓	✓	✗
— *Video*	✓	✗	✗
— *Vibration*	✓	✗	✓
Manual control	✓	✗	✓

Table 5: Comparison between different user interfaces: The green check symbol indicates that this feature is possible and implemented, while the yellow one indicates that it is possible but not implemented. The red cross marks impossible features for each device.

This overview indicates that the Android application is the most versatile interface, since it can provide all different kinds of input and feedback functionality. On the contrary, the headset is more or less limited to audio commands and feedback, while the PS3 controller offers simple input possibilities but only limited feedback.

The Android application is also the interface with the highest communication range, which directly influences the interface reliability. The short operating range of the Bluetooth devices is linked to reappearing communication losses if the distance increases between the user and the robot or if the Bluetooth receiver and the device are not in a direct line of sight.

7.2 Segmentation Algorithms

In order to compare the RANSAC based segmentation with the GPB approach, LIDAR data was collected in different environments that provide different surface structures in order to simulate the possible operational environment of an AATTV.

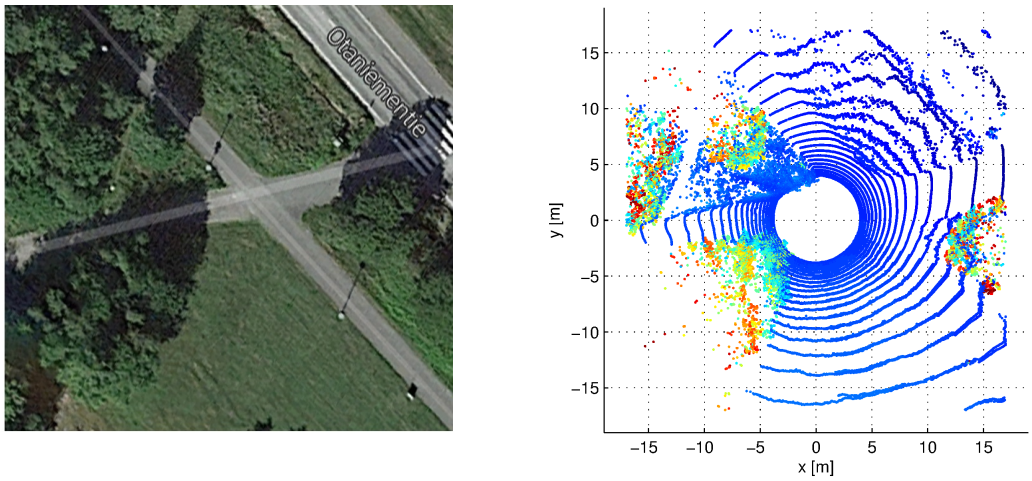
As a first performance indicator, the computation time for a complete LIDAR processing step, as described in Subsection 6.2.1, was determined by averaging the computation time needed for each recorded LIDAR scan. The average computation time using PCL's RANSAC ground segmentation was found to be 2.958 ms with a standard deviation of the 2.265 ms. Using the Gaussian-Process-Based (GPB) segmentation, the average computation time was 9.833 ms with a standard deviation of 3.310 ms.

Although the RANSAC based solution is computational less expensive, both algorithms are able to perform all necessary operations at a rate of at least 10 Hz and are therefore both suitable for this application.

Comparing the segmentation result of both algorithms in flat environments, no real performance difference was recognized. However, this changes in uneven, highly cluttered, and unstructured environments. The next subsections will determine the performance differences between both segmentation algorithms applied to those environments.

7.2.1 Uneven Environment

Experiment The first data set was recorded in front a small slope with an inclination of less than 10°. Figure 7.1 shows a bird's-eye view of the location and a single frame of the recorded LIDAR scan.

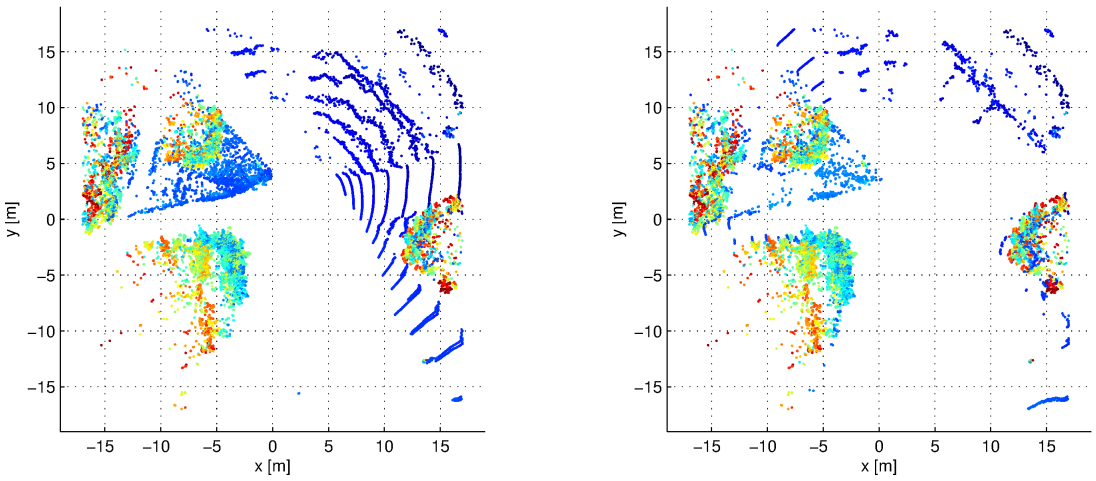


(a) Bird's-eye view

(b) Raw scan

Figure 7.1: Data set Slope: Bird's-eye view and LIDAR scan.

Result The segmentation results of both algorithms for a full 360° scan are shown in Figure 7.2. The RANSAC based ground segmentation fails to remove the entire slope in front of the car, but identifies the paths to its left, right, and back as well as the meadow correctly as drivable terrain. In a sequence of 200 scans, the slope was only removed in 31 cases by the RANSAC segmentation. On the contrary, the GPB segmentation always identified and removed the slope correctly. In the case shown in Figure 7.2 however, slight segmentation mistakes can be recognized with increasing distance.



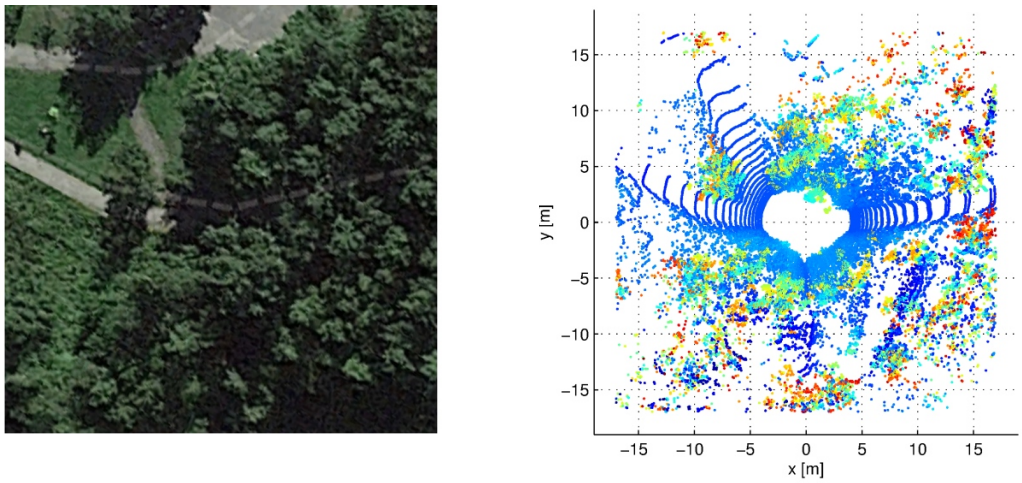
(a) RANSAC result (b) GPB result

Figure 7.2: Data set Slope result: While the RANSAC segmentation fails to recognize the slope, it is correctly removed by the GPB segmentation.

Discussion The different results can be easily explained. While the GPB segmentation builds a model of the ground, the RANSAC segmentation tries to fit a plane by randomly selecting points. If the random starting points are within the plane of the slope, this first plane can be segmented. Since this plane contains less than half of the original point count, another plane is fit into the scan and the segmentation algorithm continues. Conversely, if the starting points are not on the slope and more than half of all points in the point cloud fit into the first estimated plane, the algorithm determines and fails to segment the slope.

7.2.2 Highly-cluttered Environment

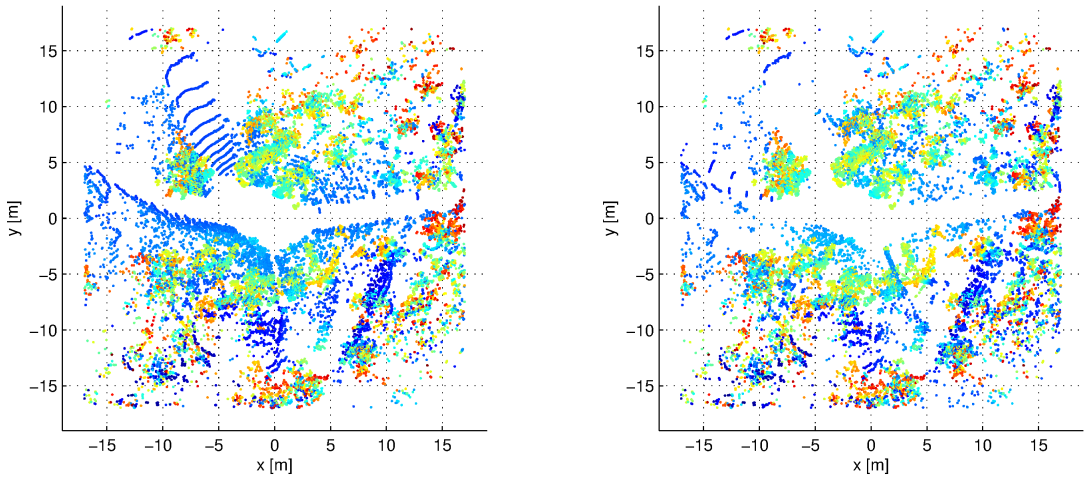
Experiment The second data set was recorded in a highly cluttered environment. In this set, the ATV is surrounded by bushes and trees. Three paths are leading away from the vehicle center as shown in Figure 7.3.



(a) Bird's-eye view (b) Raw scan

Figure 7.3: Data set Forest: Bird's-eye view and LIDAR scan.

Result Again 200 scans were visually analyzed. In 79%, all three paths were correctly segmented using RANSAC. Conversely, the GPB segmentation identified the three paths as drivable terrain in 96.5%. Figure 7.4 illustrates the result of the both segmentation algorithms for one scan. The path going in the positive y-direction is missed by the RANSAC segmentation, while the GPB segmentation shows minor mistakes on the path in negative x-direction.



(a) RANSAC result (b) GPB result

Figure 7.4: Data set Forest result: The RANSAC segmentation fails to remove the path going upwards. The GPB segmentation shows minor mistakes removing the path going into the negative x-direction.

Discussion The reason for the weaker performance of the RANSAC algorithm is again the random selection of the initial points. Although a single plane that includes all paths within a certain threshold exists, the random selection of the starting points for the plane estimation does not guarantee to find this particular plane.

7.3 Hybrid Tracking

The tracking performance of the overall system is evaluated at three different levels. First, the performance of TLD tracker is compared offline to the CMT tracking result based on common image sequences. Next, the influences of different lighting conditions as well as the user's appearance are analyzed using online data. Finally, the tracking performance is evaluated for different crossing situations and in a group of people.

7.3.1 Image Tracker Comparison

Experiment In order to compare both image tracking algorithms that are suitable for the hybrid tracker, the ATV was controlled manually and data was collected during a false crossing between two people. During a false crossing, the user walks straight ahead in front of the car until another person crosses his path. Instead of crossing, the user continues on the other person's path and vice versa as shown in Figure 7.5.

During this maneuver, the bounding boxes of the LIDAR data merge into a single box. Since a single Kalman Filter is used, it assumes that the person continues to walk straight forward. Consequently, the LIDAR tracker fails in this situation if no additional information is utilized from the image tracker. The maneuver also includes a change of the user's heading direction and, therefore, a change in the user's orientation towards the vehicle.

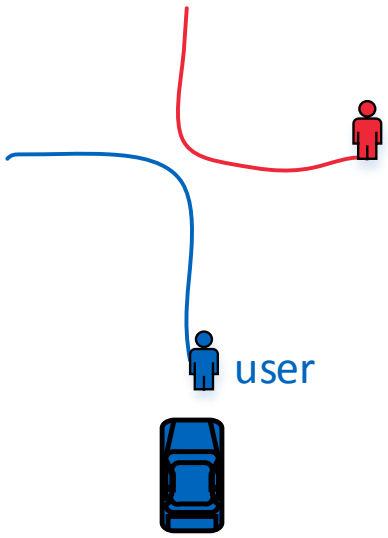


Figure 7.5: The False Crossing maneuver in order to intentionally confuse the tracker.

The CMT tracker was integrated in the overall system as described in Subsection 6.4.3. In contrast to the CMT tracker, the TLD tracker is able to learn and obtain a better model of the tracked object during run time. Therefore, in order to improve the tracking performance of the TLD tracker, it was started at the very beginning of the tracking and active throughout the whole experiment.

Figure 7.6 on the next page compares the hybrid tracker output.

Result The left image column shows the result of the hybrid tracker using CMT, while the one on the right illustrates the result of the tracker based on TLD. The top row shows the hybrid tracking result a few seconds after the initialization. At this point, both trackers rely on the LIDAR data only, since only one person is detected within the Sector of Interest (SOI) by the LIDAR tracker. The yellow box in the left image indicates that the TLD tracker is active and that its tracking confidence is already below 60%.

In the second row, two people are identified in each image based on the LIDAR data: In both cases, the correct user is identified with the help of the image tracker. The tracker confidence of the TLD tracker in this case is above 60%.

In the next row, the two clusters of the people merge into a single one, which is just small enough to fulfill the cluster requirements described in Subsection 6.2.3. The CMT tracker keeps successfully track of the user, while the TLD tracker moves in the middle of both people. The TLD tracker confidence drops again below 60%.

The clusters are separated again in the last row. The user is correctly tracked using the CMT tracker, while the TLD tracker follows the wrong person.

Discussion In order to exclude that this was a one-time failure, the TLD tracker was initialized with different bounding boxes and within different frames. Furthermore, it was tested on other image sequences with false crossings maneuvers. The TLD tracker was also integrated into the hybrid tracker in the same way as the CMT tracker. Nevertheless, the TLD tracker was always outperformed by the CMT based solution.

When the tracker was used in the same way as the CMT tracker, it stopped tracking completely once the user changed his direction. The same observation is reported in [82] for the tracking of non-rigid objects.

A reason for the weak performance of the TLD tracker is linked to the limited time and small frame number that is available to create a reliable model. In addition to that, the model of the user is created mainly based on the user's appearance from behind. However, when the user changes his direction, his orientation towards the camera changes, too. Consequently, the obtained model is no longer valid and the tracker fails.

In contrast to the TLD tracker, the CMT tracker is based on a model-free approach. The CMT tracker is, therefore, independent of the number of analyzed frames and more robust against quick changes in the user's appearance based on his orientation. Although the flaws of the TLD tracker might be compensated using pre-learned models, the CMT tracker was integrated into the final system due to its better performance in this experiment.

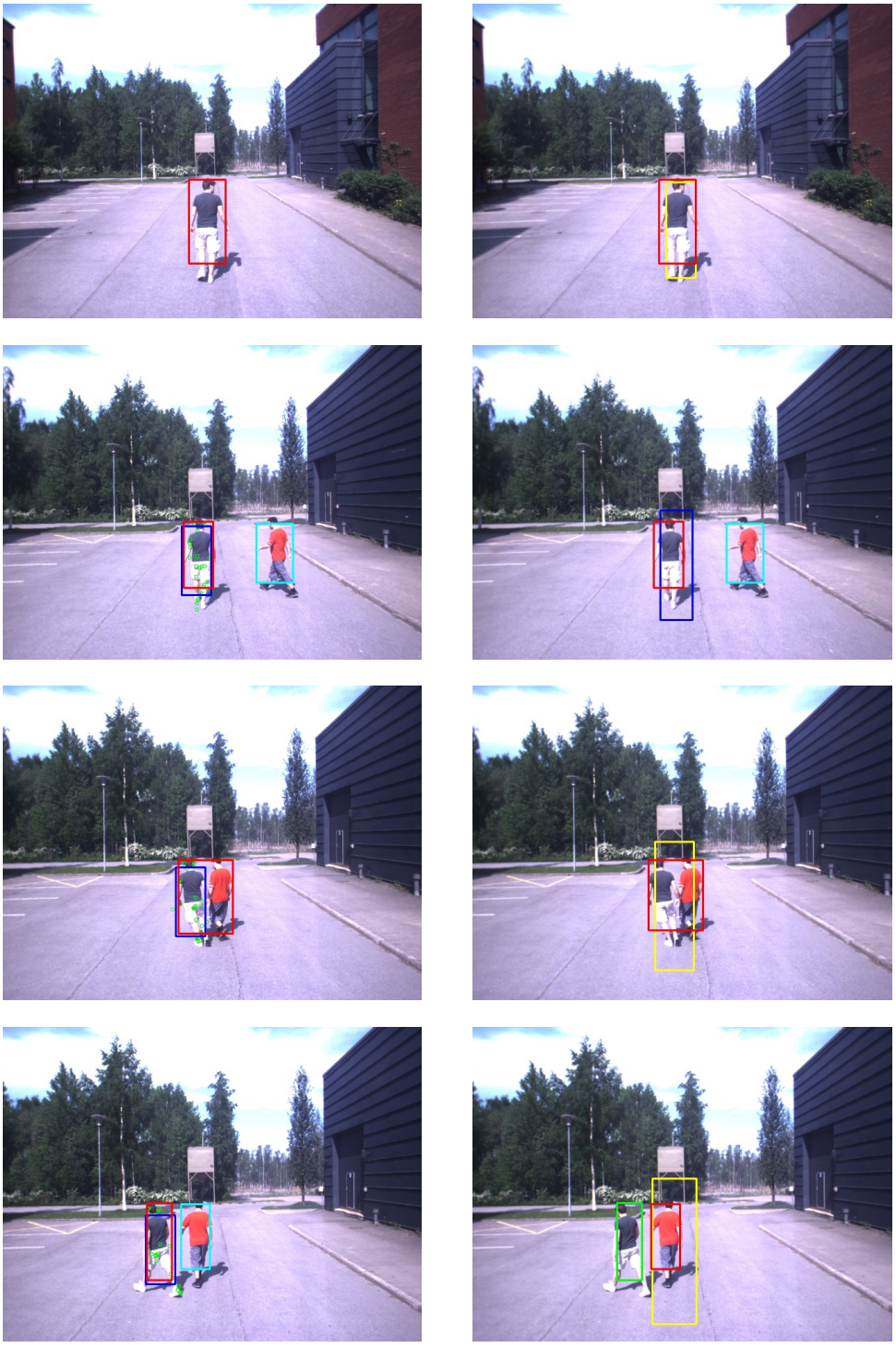


Figure 7.6: False Crossing: CMT on the left, TLD on the right. A red box indicates the position result based on fusion of the image tracking and the LIDAR tracking. A cyan box indicates another person detected by the Velodyne, while a green box indicates a miss-classified user. A blue box in the left pictures indicates the tracking estimation of the CMT tracker. The same applies for the right side if the TLD tracking confidence is above 60%, while a yellow box is displayed otherwise.

7.3.2 Tracking Influences

Experiment Next, the influence of different environment illumination and the user appearance are evaluated. The data used for this analysis was collected while driving autonomously under real-world conditions. In order to produce different lighting conditions, the shutter time was adjusted manually. Similarly, the user appearance was changed by providing clothes with different levels of contrast.

Result Figure 7.7 illustrates two extreme setups from this experiment. The image sequence represented in the left column shows a false crossing maneuver during good lighting conditions with a user wearing very high-contrast clothes. On the other hand, the images in the right column show the same maneuver using a very short illumination time, while the user wears very dark and low-contrast clothes.

The first image row shows the tracking result after initializing the hybrid tracker in each sequence. Under good circumstances, 20 features are selected and tracked by the CMT tracker. However, only 7 features are marked in the right image.

This low number of features is the reason for the bad tracking result in second image row on the right side. About 3 seconds after the initialization, the tracker is already off by more than 60%. In contrast to that, the tracker performs very well on the left side, showing only a slide rightward shift.

The actual false crossing is shown in the third row. The image tracker resolves the conflict correctly in the left column. In the right column, however, the user is completely lost out of scope by the CMT tracker. As a consequence, the image tracking result coincides now with the wrong person's bounding box.

In the last row, the result of the image tracker on the right side does not overlap with either of the two bounding boxes that were determined by processing the LIDAR data. In this situation the car can either continue to drive and rely on the information from the Kalman Filter or stop. Depending on the safety requirements, either choice can be preferred.

Discussion The present results are significant in two major respects. First of all, the proposed system is able to follow a person completely autonomously. Moreover, the system can resolve the conflict resulting from a false crossing. Nevertheless, the number of initially detected and tracked features is a criterion that solely decides whether the tracking can succeed or not. Therefore, a sufficiently illuminated environment as well as a user's appearance that provides a high contrast are essential. The easiest way to artificially create a high contrast in the user's appearance is to use clothes with a high contrast. The experiments indicate that reliable results are achieved if at least 15 features can be detected.

Figure 7.7: False Crossing in different situations. The scenes in the left column are well illuminated and the user wears clothes with a high contrast. Conversely, the scenes in the right column are badly illuminated and the user wears low-contrast clothes. The user is marked with a red box, while other people that are detected by the LIDAR are indicated with a cyan box. The blue box shows the CMT image tracking result. The green circles show the tracked features.

7.3.3 Tracking with Crossing People

Experiment In this step, the tracking performance is evaluated in true crossing situations. Two kinds of crossings were performed. First, another person crosses the user's path in front of the user. Second, the crossing occurs between the user and the vehicle in such a way that the user is almost fully occluded for a short time.

Result Figure 7.9 shows the result of this experiment. The left column shows the crossing in front of the user. The right column shows the crossing between the user and the car.

In the first row, both image trackers are initialized with a sufficiently high number of features and the number of tracked feature remains constantly high, even if the other person comes closer to the user, which is shown in the second row.

The system works as it is supposed to until the actual crossings starts in the third row. In the left column, some of the tracked features are wiped away during the crossing. However, the majority stays on the user, allowing to keep track of him. Conversely, the second person wipes all features from the user while he is partly occluded in the right image.

While the correct user is still tracked in the last image on the left side, the user is confused with another person on the right side. The CMT tracker fails in this situation.

Discussion Again, both situations were analyzed in several sequences in order to exclude a one-time failure. The CMT tracker is capable to resolve ambiguities resulting from multiple people in the scene, as long as the user is fully visible. Conversely, the CMT tracker managed only occasionally to resolve crossing situations with user occlusion. This observation contradicts the statement of the authors in the original publication [60]. However, the *person crossing* sequence from [83] used in the original publication indicates the same problem (see Figure 7.8).

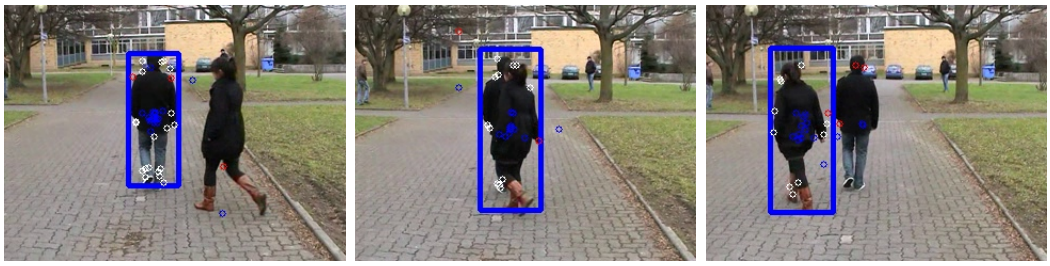


Figure 7.8: CMT tracking result on the *person crossing* sequence.

The CMT tracker fails in this situation since the features that describe the user, are wiped away by the crossing person. This feature displacement results in incorrect votes for the object's center. However, the features stick to the crossing person, since a similar consensus can be found. The initial feature displacement results from the Lucas-Kanade optical flow tracker that is used in the CMT algorithm to estimate the object motion.

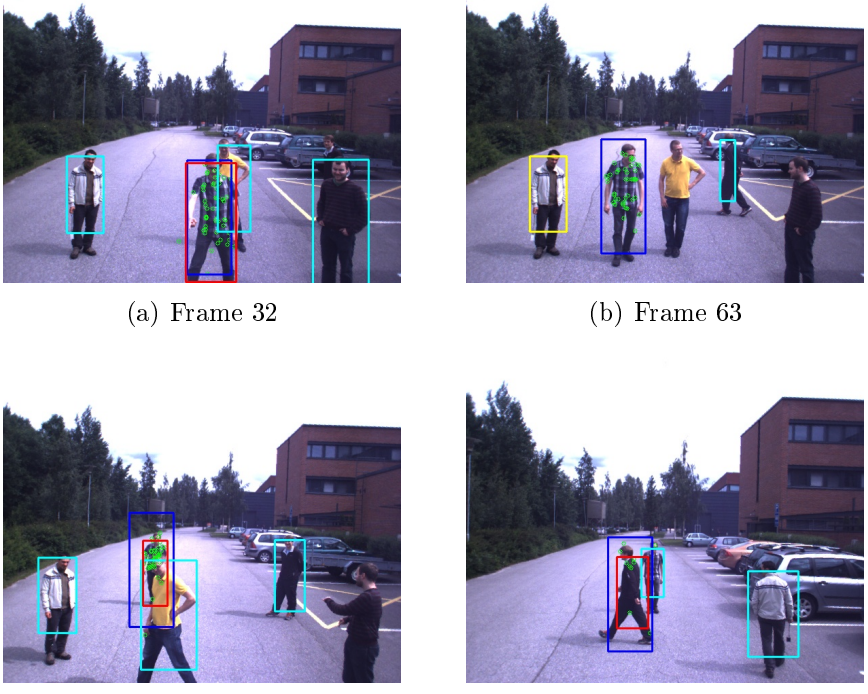
Figure 7.9: Two different crossing situations: On the left side, the user's path is crossed by another person in front of the user, while on the right side, the crossing happens between the user and the car. The user is marked with a red box, while other people that are detected by the LIDAR are indicated with a cyan box. The blue box shows the CMT image tracking result. The green circles show the tracked features.

7.3.4 Tracking in a Group of People

Experiment Finally, the tracking performance in a group of people is evaluated. The person who is the closest in front of the car is selected as user. First, the user moves randomly within the group while he is fully visible. Next, he moves away from the vehicle and three people cross between him and the vehicle after each other.

Result The user can be tracked in a group of people as long as he is fully visible like in Figure 7.10 a). However, if the user is too close to another person, the resulting point cloud can not be clustered correctly and the tracker fails as in Figure 7.10 b). The tracker can recover from this situation if the user increases the distance to his neighbors again. In contrast to that, the tracker can fail completely if the user is occluded. Depending upon how much the user is occluded, a different amount of features is wiped away by the crossing person. The more features remain on the user, the higher are the chances to keep track of him. While the crossing is successfully resolved in Figure 7.10 c), it fails for the third crossing in Figure 7.10 d).

Discussion The observations from this experiment confirm the previous results.



(a) Frame 32 (b) Frame 63

(c) Frame 111 (d) Frame 224

Figure 7.10: Tracking in a group of people: The user is marked with a red box, while other people that are detected by the LIDAR are indicated with a cyan box. The yellow box indicates a falsely classified person. The blue box shows the CMT image tracking result. The green circles show the tracked features. The person on the right in image b) and c) is not within the Sector of Interest of the LIDAR processing.

7.4 Control Behavior

Experiment In order to evaluate the control behavior of the system, the user walks along a fixed path, while the car follows according to the controller output. The desired distance between the user and the vehicle was set to 7.5 m. The user's position relative to the vehicle, the GPS position of the vehicle, as well as its speed and steering wheel position are recorded and analyzed offline.

In order to record all necessary information, a separate ROS driver had to be implemented for collecting the GPS and IMU data from the Velodyne.

Result Figure 7.11 shows the raw GPS trajectory of the ATV as well as the estimated path of the user. The path estimation is based on the relative user position and corrected GPS heading data.

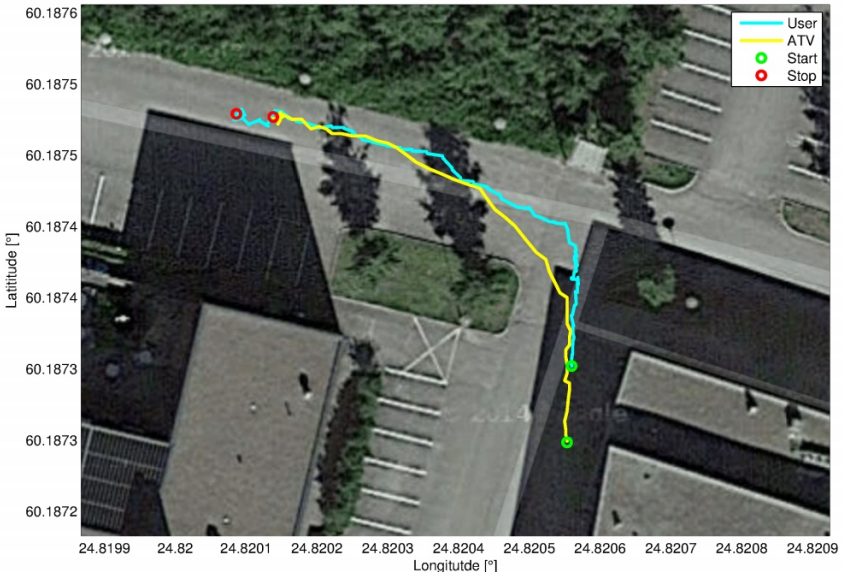


Figure 7.11: GPS based ATV trajectory and estimated user position.

This experiment shows that the ATV follows very close to the user's path as long as the user moves straight ahead. However, as expected, the vehicle does not always follow exactly the user's path. Once the user starts to walk a curve, the car starts to steer in the according direction. Therefore, the curve described by the car's trajectory appears to be long and flat compared to the user's immediate change in direction. The average distance between the car and the user was measured to be 7.35 m with a standard deviation of 0.15 m.

Figure 7.12 shows the relative distance between the vehicle and the user, as well as the measured speed of the vehicle. The speed measurements are based on the ATV's odometry as well as the GPS data.

Figure 7.13 plots the user direction relative to the ATV. Additionally, the current steering wheel position is shown.

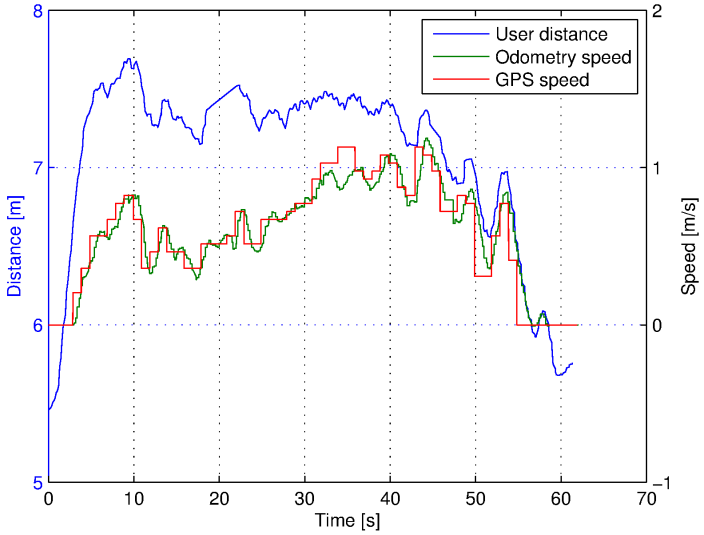


Figure 7.12: Input and measured result of the speed control.

Discussion Comparing the user distance and the correspondingly measured speed, two different speed levels can be identified, although the distance fluctuates more or less around the average distance of 7.35 m. In the curve (10 s – 35 s), the car moves noticeably slower than when driving straight ahead. The reason for this is that the applied control speed is only proportional to the user's distance. The actual speed, however, depends also on the orientation of the steering wheels, since the friction of the front wheels is higher if they are not aligned with the driving rear wheels.

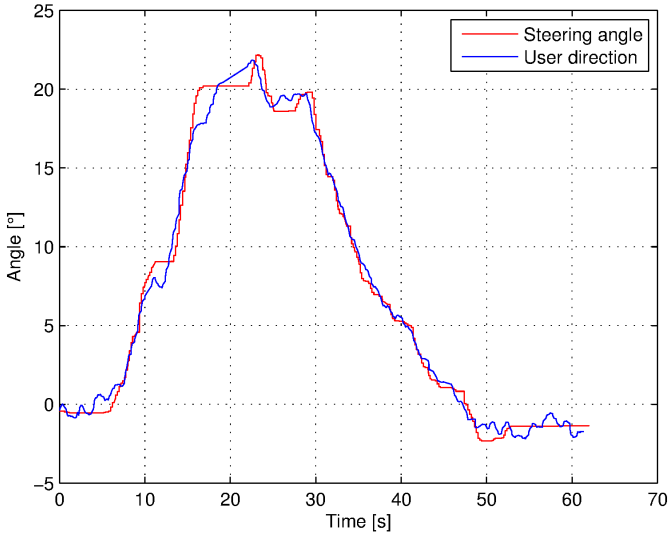


Figure 7.13: Input and measured result of the steering control.

The proportional controller for the steering angle works as expected. The angle in- and decreases depending on the user's direction relative to the vehicle.

7.5 System Performance

Experiment The overall system performance is evaluated based on a long test run. In this test run, a total distance of 450 m was covered in 400 s. Figure 7.14 shows the traveled trajectory. The trajectory was obtained by combining the GPS data, the odometry and IMU data in an Extended Kalman Filter. The selected track provides curves with different radii, narrow corridors between bushes and parked cars as well as randomly walking by pedestrians. The complete trajectory is shown in Figure 7.15. In this experiment, the controller was set to follow the person in a distance of 6.5 m.

Figure 7.14: Trajectory of the AATTV during a 400 s test run.

Result The average distance between the user and the ATV was found to be 6.45 m, having a standard error of 0.66 m. The average speed was measured to be 1.11 m/s with a standard deviation of 0.33 m/s. During the complete run, the user was not lost a single time. Consequently, the hybrid tracker did not have to be restarted once.

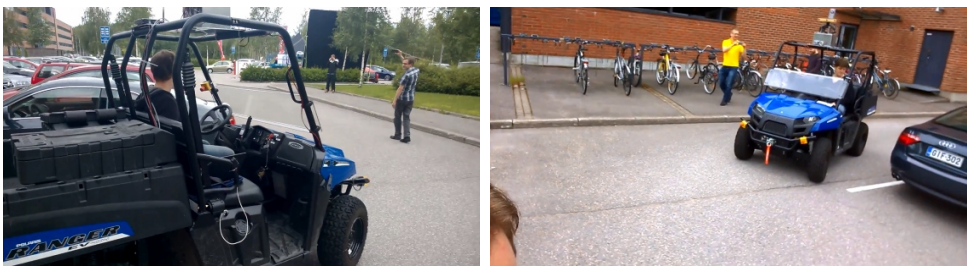


Figure 7.15: The test environment: Tight turning during the test run with multiple pedestrians and parked cars. A person sits in the car for safety and insurance reasons.

8 Conclusions

A human-robot interface that allows a human user to interact with an Autonomous All-Terrain Transport Vehicle (AATTV) was designed from scratch, implemented and extensively evaluated. The developed system combines 3D range data from a Velodyne HDL-32E with intensity images from a Bumblebee XB3 camera. Thereby, the sensors are combined in such a way that situation-dependent either only the laser scan is used or both information sources are fused in order to resolve ambiguities. This adaptive combination allows a reliable, long-term and real-time tracking of a human user within a real world environment, particularly in the presence of frequent illumination changes and multiple humans.

Subsequently, the major steps during the development of the system are summarized, known system limitations discussed and solutions within a possible future work outlined.

8.1 System Development

The system was developed in three phases: The design, the implementation and the evaluation phase.

In the design phase, existing solutions were compared and evaluated. While full solutions are completely restricted to military applications, potential applications exist in civil and space scenarios, too. Important requirements for an easy-to-use human-robot interface were identified. Above all, the system must operate in real time under real conditions. Furthermore, it must be able to identify, track, and follow a human user.

In order to implement a system that fulfills the given requirements, various subtasks had to be solved. Those tasks included, among others, the implementation of different user interfaces, the readout of sensor data, the processing of the obtained information, the external sensor calibration, the sensor fusion, and the actual vehicle control. The software development was based on the Robot Operating System (ROS), a meta-operating system.

The evaluation assessed different, critical sub-systems as well as the total system performance. It shows that a smart phone user interface impresses by its strong reliability, high range capacity and its overall versatility. More over, the evaluation proves that the implemented LIDAR segmentation algorithm can be used in uneven and highly-cluttered environments. It also shows that the proposed hybrid tracker is able to resolve ambiguities that result from the presence of multiple people in the observed scene as long as the user is fully visible. However, a well illuminated environment as well as a user appearance that provides a sufficient number of features are required in order to guarantee a reliable and robust tracking. In addition to that, the implemented control algorithm to drive the vehicle is suitable to perform tight turnings and follow a human user precisely on straight tracks. Finally, the evaluation shows that the developed system is able to identify, track and follow a human user under real world conditions over a total distance of at least 450 m.

8.2 System Limitations

Although the system performed overwhelmingly well during the system evaluation, different limitations could be identified.

First of all, the performance of the hybrid tracker is limited to the performance of the image tracker that is used to resolve ambiguities. Meaning that if the utilized image tracker is unable to handle the complete occlusion of the user, the hybrid tracker will fail as a direct consequence. Contrary to the assertion of the authors [60], the CMT tracker failed to resolve full user occlusion correctly in the majority of the conducted experiments. The system is therefore not able to handle people crossing between the user and the vehicle reliably.

Another problem that could be observed is the limited horizontal Field of View (FOV). Although a 360° LIDAR is used, the overall FOV is limited to the FOV of the statically mounted camera. Nevertheless, as long as no ambiguities result from the LIDAR tracking, the system is able to follow the user even if he is not visible in the camera image. In this situation, the system fails immediately if more than one person is detected.

The most immediate problem, however, is the vehicle's low level of autonomy. Currently, there is neither an active path planning nor any kind of obstacle avoidance. It is therefore within the user's responsibility to select suitable paths for the vehicle in order to avoid potential collisions.

8.3 Future Work

Possible future work should aim to solve the mentioned problems and limitations.

Thus, a different approach could be used to resolve LIDAR ambiguities. For example, a multi-hypothesis Kalman Filter could be implemented to estimate the possible paths of each user that is detected from the LIDAR data. Other solutions could try to improve the occlusion handling of the utilized image tracker by evaluating other features such as a user based color histogram.

The limited horizontal FOV could be improved by using a 360° camera like the Ladybug 5 camera from Point Grey. Another solution could be a pivotable camera, that can be turned in the user's direction in similar way as the Sector of Interest (SOI) is selected from the LIDAR data.

The vehicle's level of autonomy could be raised by implementing standard approaches for obstacle avoidance and path planning. In this regard, ROS offers a diverse selection of ready made algorithms and solutions such as the potential field method or sliding window approaches, respectively. In ROS, however, those algorithms are mainly designed for differential drive and holonomic robots. They therefore have to be adapted in order to suit a car-like robot.

References

[1] NASA – National Aeronautics and Space Administration. Lunar Exploration Timeline. Website, 2013. Accessed on 3.3.2014. URL: http://nssdc.gsfc.nasa.gov/planetary/lunar/lunartimeline.html.

[2] NASA – National Aeronautics and Space Administration. Apollo 17 press kit. Website, 1972. Archived from the original on 21.7.2011, accessed on 3.3.2014. URL: https://mira.hq.nasa.gov/history/ws/hdmshrc/all/main/DDD/17980.PDF.

[3] NASA – National Aeronautics and Space Administration. Lunar roving vehicle. Website, 2014. Accessed on 10.3.2014. URL: http://www.nasa.gov/images/content/199818main_rs_image_feature_774_946x710.jpg.

[4] CNN – Cable News Network. Robots fail to complete Grand Challenge. Website, 2004. Accessed on 10.2.2014. URL: http://edition.cnn.com/2004/TECH/ptech/03/14/darpa.race/index.html.

[5] DARPA – Defense Advanced Research Projects Agency. Urban Challenge. Website, 2007. Accessed on 10.2.2014. URL: http://archive.darpa.mil/grandchallenge/.

[6] E. Guizzo. How Google self-driving car works. Website, 2011. Accessed on 10.2.2014. URL: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works.

[7] S. Thrun. What we're driving at. Website, 2010. Accessed on 12.2.2014. URL: http://googleblog.blogspot.fi/2010/10/what-were-driving-at.html.

[8] C. Urmson. Just press go: designing a self-driving vehicle. Website, 2014. Accessed on 17.7.2014. URL: http://googleblog.blogspot.fi/2014/05/just-press-go-designing-self-driving.html.

[9] S.J. Moorehead, C.K. Wellington, H. Paulino, and J.F. Reid. R-Gator: An Unmanned Utility Vehicle. In *Proceedings of the SPIE*, page 769, 2010. doi:10.1117/12.852483.

[10] Lockheed Martin. Squad Mission Support System specifications, 2013. Accessed on 10.2.2014. URL: http://www.lockheedmartin.com/content/dam/lockheed/data/mfc/pc/smss/mfc-smss-pc.pdf.

[11] Lockheed Martin. Squad Mission Support System product page, 2013. Accessed on 10.2.2014. URL: http://www.lockheedmartin.com/us/products/smss.html.

[12] Lockheed Martin. Squad Mission Support System product page, 2013. Accessed on 10.2.2014. URL: http://www.lockheedmartin.com/content/dam/lockheed/data/mfc/photo/smss/mfc-smss-photo-01-main-h.jpg.

[13] DARPA – Defense Advanced Research Projects Agency. Legged Squad Support System. Website, 2013. Accessed on 10.2.2014. URL: http://www.darpa.mil/Our_Work/TTO/Programs/Legged_Squad_Support_System_%28LS3%29.aspx.

[14] Boston Dynamics. LS3 - Legged Squad Support Systems. Website, 2014. Accessed on 10.2.2014. URL: http://www.bostondynamics.com/robot_ls3.html.

[15] BBC – British Broadcasting Corporation. US marines test robotic mule at RIMPAC, Hawaii, 2014. Accessed on 20.7.2014. URL: http://www.bbc.com/news/technology-28290945.

[16] E. Ackerman. Video Friday: RHex Pronking, LS3 Goes to Hawaii, and RoboBoat 2014. Website, 2014. Accessed on 20.7.2014. URL: http://spectrum.ieee.org/automaton/robotics/robotics-hardware/video-friday-rhex-pronking-ls3-goes-to-hawaii-and-roboboat-2014.

[17] M. Seelinger and J.-D. Yoder. Automatic Visual Guidance of a Forklift Engaging a Pallet. *Robot. Auton. Syst.*, 54(12):1026–1038, December 2006. doi:10.1016/j.robot.2005.10.009.

[18] T.A. Tamba, B. Hong, and K.-S. Hong. A path following control of an unmanned autonomous forklift. *International Journal of Control, Automation and Systems*, 7(1):113–122, 2009. doi:10.1007/s12555-009-0114-y.

[19] S. Teller, M.R. Walter, M. Antone, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, J. Glass, J.P. How, A.S. Huang, J.H. Jeon, S. Karaman, B. Luders, N. Roy, and T. Sainath. A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 526–533, May 2010. doi:10.1109/ROBOT.2010.5509238.

[20] ROS – Robot Operating System. A new robot joins the ros community. Website, 2014. Accessed on 10.2.2014. URL: http://www.ros.org/news/2013/04/a-new-robot-joins-the-ros-community.html.

[21] Adept MobileRobots. Seekur Jr. Website, 2012. Accessed on 3.3.2014. URL: http://www.mobilerobots.com/ResearchRobots/SeekurJr.aspx.

[22] Clearpath Robotics. Grizzly. Website, 2014. Accessed on 10.2.2014. URL: http://www.clearpathrobotics.com/grizzly/features/.

[23] E. Ackerman. Clearpath Robotics Announces Grizzly Robotic Utility Vehicle. Website, 2013. Accessed on 3.3.2014. URL: http://spectrum.ieee.org/automaton/robotics/industrial-robots/clearpath-robotics-announces-grizzly-robotic-utility-vehicle.

[24] MobileRobots. Seekur Jr. Website, 2013. Accessed 3 3.2.2014. URL: http://www.mobilerobots.com/ResearchRobots/SeekurJr.aspx.

[25] NASA – National Aeronautics and Space Administration. Chronology of Mars Exploration. Website, 2013. Accessed on 3.3.2014. URL: http://nssdc.gsfc.nasa.gov/planetary/chronology_mars.html.

[26] NASA – National Aeronautics and Space Administration. NASA's Space Exploration Vehicle (SEV). Website, 2012. Accessed on 28.2.2014. URL: http://www.nasa.gov/exploration/technology/space_exploration_vehicle/index.html.

[27] A.F.J. Abercromby, M.L. Gernhardt, and H. Litaker. Desert research and technology studies (drats) 2009: a 14-day evaluation of the space exploration vehicle prototype in a lunar analog environment. *NASA Technical Report*, 2012. URL: http://ston.jsc.nasa.gov/collections/TRS/_techrep/TP-2012-217360.pdf.

[28] NASA – National Aeronautics and Space Administration. Building the Future Spacesuit. Website, 2012. Accessed on 3.3.2014. URL: http://www.nasa.gov/offices/oce/appel/ask/issues/45/45s_building_future_spacesuit.html.

[29] R.R. Murphy, T. Nomura, A. Billard, and J.L. Burke. Human-Robot Interaction. *Robotics Automation Magazine, IEEE*, 17(2):85–89, June 2010. doi:10.1109/MRA.2010.936953.

[30] M. A. Goodrich and A. C. Schultz. Human-robot Interaction: A Survey. *Found. Trends Hum.-Comput. Interact.*, 1(3):203–275, January 2007. doi:10.1561/1100000005.

[31] R.R. Murphy. Human-robot interaction in rescue robotics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):138–153, May 2004. doi:10.1109/TSMCC.2004.826267.

[32] J.L. Drury, J. Scholtz, and H.A. Yanco. Awareness in human-robot interactions. *IEEE International Conference on Systems, Man and Cybernetics, 2003.*, 1:912–918 vol.1, Oct 2003. doi:10.1109/ICSMC.2003.1243931.

[33] A. Halme, I. Leppänen, J. Suomela, Sami Ylönen, and I. Kettunen. WorkPartner: interactive human-like service robot for outdoor applications. *The international journal of robotics Research*, 22(7-8):627–640, 2003. doi:10.1177/02783649030227011.

[34] X. Ma, C. Hu, X. Dai, and K. Qian. Sensor integration for person tracking and following with mobile robot. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3254–3259, Sept 2008. doi:10.1109/IROS.2008.4650644.

[35] Wu, B.-F. and Jen, C.-L. and Li, W.-F. and Tsou, T.-Y. and Tseng, P.-Y. and Hsiao, K.-T. RGB-D sensor based SLAM and human tracking with Bayesian

framework for wheelchair robots. In *Advanced Robotics and Intelligent Systems (ARIS), 2013 International Conference on*, pages 110–115, May 2013. doi:10.1109/ARIS.2013.6573544.

[36] Y. Nagumo and A. Ohya. Human following behavior of an autonomous mobile robot using light-emitting device. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 225–230, 2001. doi:10.1109/ROMAN.2001.981906.

[37] R. Gockley, A. Bruce, J. Forlizzi, M. Michalowski, A. Mundell, S. Rosenthal, B. Sellner, R. Simmons, K. Snipes, A.C. Schultz, and Jue Wang. Designing robots for long-term social interaction. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1338–1343, Aug 2005. doi:10.1109/IROS.2005.1545303.

[38] K. Nohara, T. Tajika, M. Shiomi, T. Kanda, H. Ishiguro, and N. Hagita. Integrating passive RFID tag and person tracking for social interaction in daily life. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 545–552, Aug 2008. doi:10.1109/ROMAN.2008.4600723.

[39] R. C. Luo, C. H. Huang, and T. T. Lin. Human tracking and following using sound source localization for multisensor based mobile assistive companion robot. In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, pages 1552–1557, November 2010. doi:10.1109/IECON.2010.5675451.

[40] M.M. Loper, N.P. Koenig, S.H. Chernova, C.V. Jones, and O.C. Jenkins. Mobile human-robot teaming with environmental tolerance. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 157–164. ACM, 2009. URL: http://cs.brown.edu/~cjenkins/papers/matt_hri2009.pdf.

[41] N. Bellotto and H. Hu. Multisensor-based Human Detection and Tracking for Mobile Service Robots. *Trans. Sys. Man Cyber. Part B*, 39(1):167–181, February 2009. doi:10.1109/TSMCB.2008.2004050.

[42] R. Gockley, J. Forlizzi, and R. Simmons. Natural Person-following Behavior for Social Robots. In *Proceedings of the ACM/IEEE International Conference on Human-robot Interaction*, HRI '07, pages 17–24, New York, NY, USA, 2007. ACM. doi:10.1145/1228716.1228720.

[43] D. Geronimo, A.M. Lopez, A.D. Sappa, and T. Graf. Survey of Pedestrian Detection for Advanced Driver Assistance Systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1239–1258, July 2010. doi:10.1109/TPAMI.2009.122.

[44] T. Gandhi and M.M. Trivedi. Pedestrian Protection Systems: Issues, Survey, and Challenges. *Intelligent Transportation Systems, IEEE Transactions on*, 8(3):413–430, Sept 2007. doi:10.1109/TITS.2007.903444.

[45] A. Yilmaz, O. Javed, and M. Shah. Object Tracking: A Survey. *ACM Comput. Surv.*, 38(4), December 2006. doi:10.1145/1177352.1177355.

[46] M. Enzweiler and D.M. Gavrila. Monocular Pedestrian Detection: Survey and Experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, Dec 2009. doi:10.1109/TPAMI.2008.260.

[47] B. Schiele, M. Andriluka, N. Majer, S. Roth, and C. Wojek. Visual People Detection: Different Models, Comparison and Discussion. In *Proceedings of the IEEE ICRA Workshop on People Detection and Tracking*, May 2009. URL: http://www.d2.mpi-inf.mpg.de/sites/default/files/schiele09icraws.pdf.

[48] J. Canny. A Computational Approach to Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov 1986. doi:10.1109/TPAMI.1986.4767851.

[49] B.K.P. Horn and B.G. Schunck. Determining Optical Flow. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1980. URL: http://people.csail.mit.edu/bkph/papers/Optical_Flow_OPT.pdf.

[50] B.D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. URL: http://www.ces.clemson.edu/~stb/klt/lucas_bruce_d_1981_1.pdf.

[51] B.S. Manjunath, J.-R. Ohm, V.V. Vasudevan, and A. Yamada. Color and texture descriptors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):703–715, Jun 2001. doi:10.1109/76.927424.

[52] C. Papageorgiou and T. Poggio. A Trainable System for Object Detection. *Int. J. Comput. Vision*, 38(1):15–33, June 2000. doi:10.1023/A:1008162616689.

[53] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005. doi:10.1109/CVPR.2005.177.

[54] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi:10.1007/BF00994018.

[55] R.E. Schapire. The Strength of Weak Learnability. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 28–33, Washington, DC, USA, 1989. IEEE Computer Society. doi:10.1109/SFCS.1989.63451.

[56] Y. Freund and R.E. Schapire. A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997. doi:10.1006/jcss.1997.1504.

[57] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-Backward Error: Automatic Detection of Tracking Failures. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2756–2759, Aug 2010. doi:10.1109/ICPR.2010.675.

[58] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, July 2012. doi:10.1109/TPAMI.2011.239.

[59] G. Nebehay. Robust Object Tracking Based on Tracking-Learning-Detection. Master's thesis, Vienna University of Technology, Faculty of Informatics, 2012. URL: http://www.gnebehay.com/publications/master_thesis/master_thesis.pdf.

[60] G. Nebehay and R. Pflugfelder. Consensus-based Matching and Tracking of Keypoints for Object Tracking. In *Winter Conference on Applications of Computer Vision*. IEEE, March 2014. URL: http://www.epics-project.eu/publications/2014_nebehay_wacv.pdf.

[61] J. Satake and J. Miura. Robust Stereo-Based Person Detection and Tracking for a Person Following Robot. *Workshop on People Detection and Tracking IEEE ICRA*, 2009. URL: http://srl.informatik.uni-freiburg.de/conferences/icra09ws/papers/19P-Satake.pdf.

[62] L. Spinello and K.O. Arras. People Detection in RGB-D Data. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2011. URL: http://www.informatik.uni-freiburg.de/~spinello/spinelloIROS11.pdf.

[63] M. Munaro, F. Basso, and E. Menegatti. Tracking people within groups with RGB-D data. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2101–2107, Oct 2012. doi:10.1109/IROS.2012.6385772.

[64] T. Taipalus and J. Ahtiainen. Human detection and tracking with knee-high mobile 2D LIDAR. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 1672–1677, Dec 2011. doi:10.1109/ROBIO.2011.6181529.

[65] K.O. Arras, S. Grzonka, M. Luber, and W. Burgard. Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1710–1715, May 2008. doi:10.1109/ROBOT.2008.4543447.

[66] A. Carballo, A. Ohya, and S. Yuta. Fusion of double layered multiple laser range finders for people detection from a mobile robot. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 677–682, Aug 2008. doi:10.1109/MFI.2008.4648023.

[67] A. Carballo, A. Ohya, and S. Yuta. People detection using range and intensity data from multi-layered Laser Range Finders. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5849–5854, Oct 2010. doi:10.1109/IROS.2010.5649769.

[68] L. Spinello, K.O. Arras, R. Triebel, and R. Siegwart. A Layered Approach to People Detection in 3D Range Data. In *AAAI*, 2010. URL: http://www2.informatik.uni-freiburg.de/~spinello/spinello10layered.pdf.

[69] A. Teichman, J. Levinson, and S. Thrun. Towards 3D object recognition via classification of arbitrary object tracks. In *ICRA*, pages 4034–4041. IEEE, 2011. doi:10.1109/ICRA.2011.5979636.

[70] K. Bohlmann, A. Beck-Greinwald, S. Buck, H. Marks, and A. Zell. Autonomous Person Following with 3D LIDAR in Outdoor Environments. In *1st International Workshop on Perception for Mobile Robots Autonomy (PEMRA 2012)*, September 2012. URL: http://www.jamris.org/images/ISSUES/ISSUE-2013-02/JAMRIS_No02_2013_P_24-29.pdf.

[71] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 1044–1049, 2007. doi:10.1109/ITSC.2007.4357637.

[72] Velodyne. Velodyne HDL-32E data sheet. Website, 2014. Accessed on 22.4.2014. URL: http://velodynelidar.com/lidar/hdlproducts/97-0038d%20HDL-32E_datasheet.pdf.

[73] Point Grey. Bumblebee XB3 data sheet. Website, 2014. Accessed on 22.4.2014. URL: http://www.ptgrey.com/products/bbxb3/bumblebee2_xb3_datasheet.pdf.

[74] ROS – Robot Operating System. Robot Operating System – Introduction, 2014. Accessed on 13.7.2014. URL: http://wiki.ros.org/ROS/Introduction.

[75] PCL – Point Cloud Library. About Point Cloud Library, 2014. Accessed on 13.7.2014. URL: http://pointclouds.org/about/.

[76] OpenCV – Open Source Computer Vision Library. About Open Source Computer Vision Library, 2014. Accessed on 13.7.2014. URL: http://opencv.org/about.html.

[77] A. Rudnicky. Sphinx Knowledge Base Tool – VERSION 3, 2014. Accessed on 17.5.2014. URL: http://www.speech.cs.cmu.edu/tools/lmtool-new.html.

[78] T. Chen, B. Dai, R. Wang, and D. Liu. Gaussian-Process-Based Real-Time Ground Segmentation for Autonomous Land Vehicles. *Journal of Intelligent and Robotic Systems*, pages 1–20, 2013. doi:10.1007/s10846-013-9889-4.

[79] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart. A Comparison of Line Extraction Algorithms Using 2D Range Data for Indoor Mobile Robotics. *Auton. Robots*, 23(2):97–111, August 2007. doi:10.1007/s10514-007-9034-y.

[80] R.B. Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.* PhD thesis, Computer Science department, Technische Universität München, Germany, October 2009. doi:10.1007/s13218-010-0059-6.

[81] P.I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab.* Springer, 2011. doi:10.1007/978-3-642-20144-8.

[82] S. Duffner and C. Garcia. Pixeltrack: a fast adaptive algorithm for tracking non-rigid objects. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2480–2487. IEEE, 2013. URL: http://liris.cnrs.fr/Documents/Liris-6293.pdf.

[83] D.A. Klein, D. Schulz, S. Frintrop, and A.B. Cremers. Adaptive Real-Time Video-Tracking for Arbitrary Objects. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 772–777, Oct 2010. URL: http://www.iai.uni-bonn.de/~kleind/iros10_daklein.pdf.

A Hardware Setup

Figure A1 gives a detailed overview of the complete high-level hardware setup.

The Acrosser AIV-HM76V0FL is a fanless embedded PC running an Intel Core i7 3720QM processor with 16 GB RAM. The Acrosser provides inter alia two Ethernet ports as well as one integrated CAN interface.

The ASUS E45M1-M PRO is a motherboard with an integrated AMD Dual-Core Processor E-450. The board is equipped with 8 GB RAM and provides, among others, one Ethernet port and one IEEE 1394 Firewire interface.

The EPEC 5050 control unit is a commercial CAN controller. It hosts a 32-bit micro-controller running at 128 MHz.

The ZyXEL AC750 is a standard dual-band 12 V wireless router.

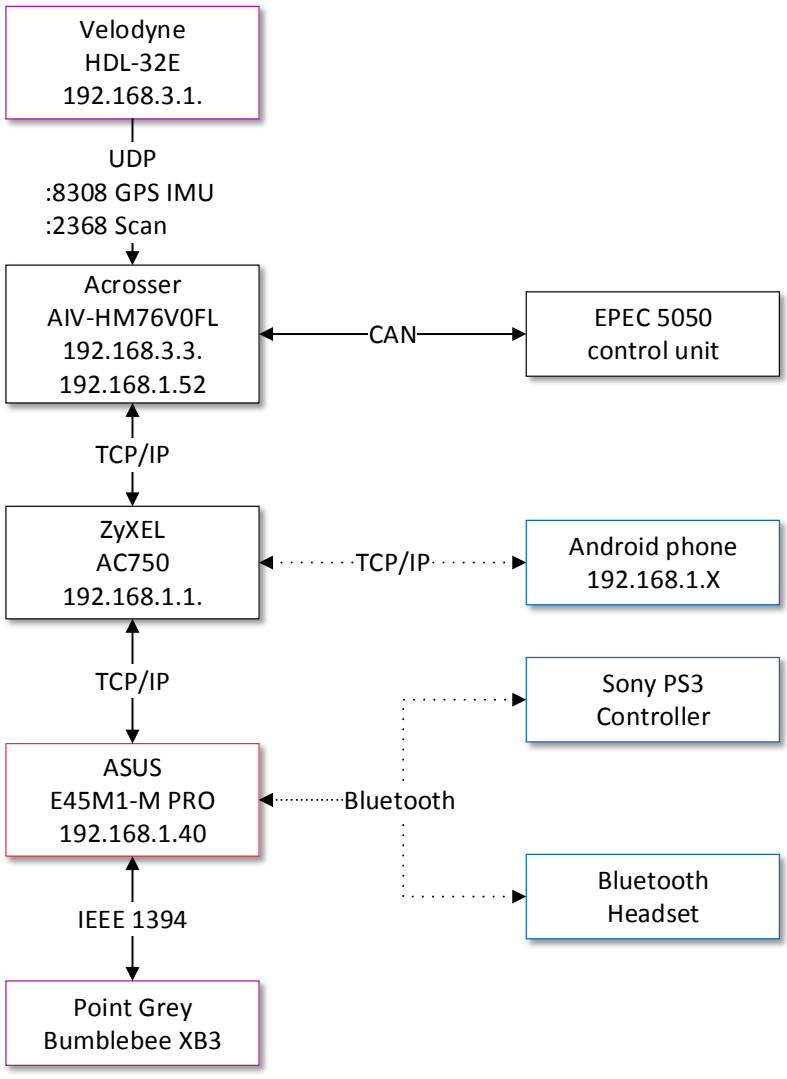


Figure A1: Detailed high-level hardware and network setup: Sensors in purple, user interface devices in blue, ROS Master in red.

B ROS Communication Graph

Figure B1 shows the complete ROS communication graph.

B.1 Custom Messages

Two new message types were defined in order to allow a customized information exchange between different nodes.

`tracking_msgs::location` The message type `tracking_msgs::location` is used to describe detected objects. It contains the following fields:

```
std_msgs/Header header
geometry_msgs/Point[] points
geometry_msgs/Point[] min
geometry_msgs/Point[] max
```

`points` is an array of 3D points describing the center of gravity for each object. The arrays `min` and `max` define a 3D bounding box for each object.

`imu_msgs::velodyneIMU` This message is used to publish the information collected from the Velodyne IMU and GPS. It contains the following fields:

```
std_msgs/Header header
float64 accel_x
float64 accel_y
float64 accel_z
float64 roll
float64 pitch
float64 yaw
string gps
```

The information of the accelerometer is stored in `accel_x`, `accel_y` and `accel_z`. `roll`, `pitch` and `yaw` contain the gyroscope data. `gps` is a NMEA sentence.

B.2 Topics

The most important topics shown in Figure B1, as well as other important topics for data logging and the vehicle control are explained in Table B1.

B.3 Nodes

The functionality of each node shown in Figure B1 is explained in Table B2.

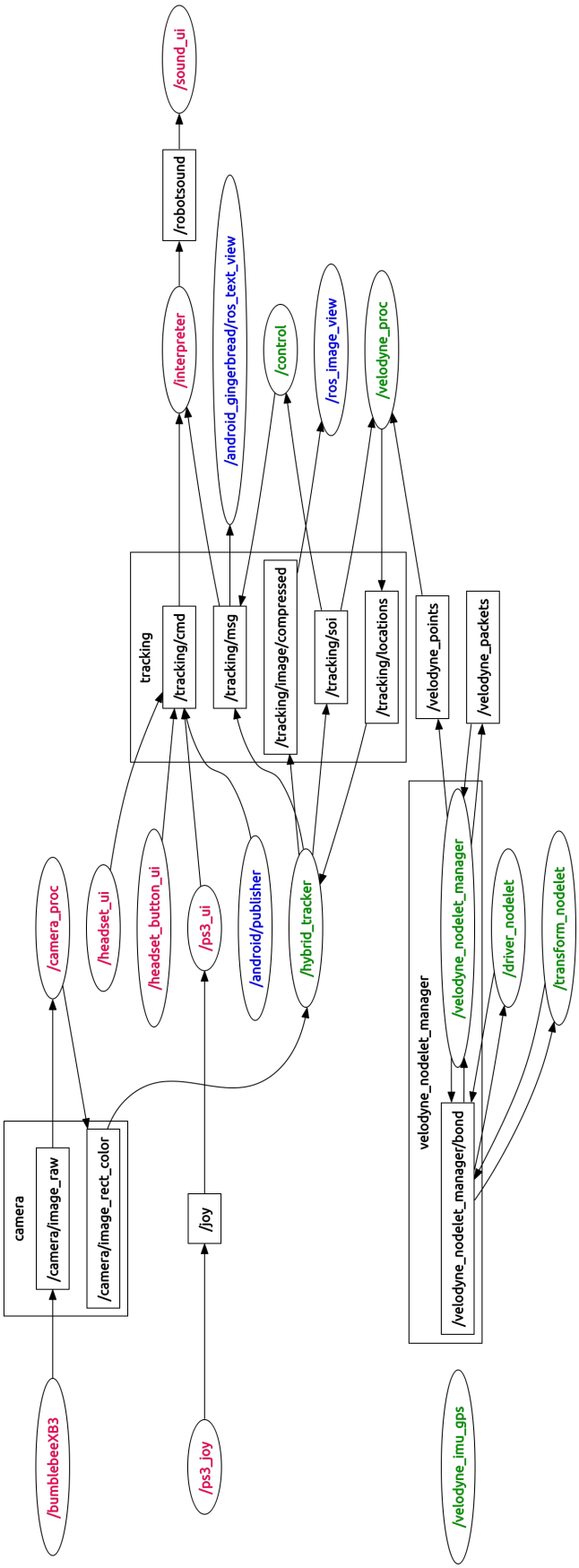


Figure B1: The complete ROS communication graph: Ovals indicate nodes, boxes indicate topics. Nodes with red text are running on the ASUS motherboard. The ones with green text run on the embedded PC Acrosser, while nodes with blue text run on an Android smart phone.

Topic	Message type	Description
/camera/image_raw	sensor_msgs::image	raw image grabbed from the Bumblebee XB3
/fromCan2Ros	ackermann_msgs::AckermannDrive	Measured steering angle and speed sent from CAN controller to ROS
/fromRos2Can	ackermann_msgs::AckermannDrive	Steering and speed control sent from ROS to CAN controller
/joy	sensor_msgs::Joy	input from the PS3 controller
/tracking/cmd	std_msgs::String	Commands published by user interface
/tracking/image/compressed	sensor_msgs::image	Image tracking result
/tracking/locations	tracking_msgs::locations	Detected objects
/tracking/msg	std_msgs::String	Debugging messages and user feedback
/tracking/soi	tracking_msgs::locations	Selected Target
/velodyne_IMU_GPS	imu_msgs::velodyneIMU	IMU and GPS information from Velodyne
/velodyne_packets	velodyne_msgs::VelodyneScan	raw packets from Velodyne LIDAR
/velodyne_points	sensor_msgs::PointCloud2	360° point cloud from Velodyne

Table B1: Important high-level topics.

Node	Description
/android_gingerbrea/ros_text_view	Feedback display on Android (Section 6.1.3).
/android/publisher	Publisher for Android user commands (Section 6.1.3).
/bumblebeeXB3	Camera1394 ROS driver for Bumblebee XB3.
/camera_proc	Image decoding and rectification.
/control	Speed and steering control (Section 6.5).
/driver_nodelet	ROS driver nodelet for Velodyne HDL-32E.
/headset_button_ui	User interface for headset button (Section 6.1.2).
/headset_ui	User interface for Bluetooth headset (Section 6.1.2).
/hybrid_tracker	Hybrid tracking (Section 6.4) and image processing (Section 6.3).
/interpreter	Common interpreter for user interfaces (Section 6.1).
/ps3_joy	ROS driver for PS3 controller.
/ps3_ui	PS3 user interface (Section 6.1.1).
/ros_image_view	Image display on Android (Section 6.1.3).
/sound_ui	Audio feedback user interface (Section 6.1.2).
/transform_nodelet	Velodyne point cloud frame transformation nodelet.
/velodyne_imu_gps	Driver for Velodyne IMU and GPS.
/velodyne_nodelet_manager	/driver_nodelet and /transform_nodelet manager.
/velodyne_proc	Velodyne data processing (Section 6.2).

Table B2: High-level ROS node functionality.

C Launch Instructions

C.1 Core Functionality

In order to run the tracking interface, first, all necessary nodes have to be started, since there is no automated start implemented yet. The most convenient way to start all required programs is to connect via SSH to both computers using another laptop in the same network. Therefore, the laptop has to establish a wireless connection with the open *ZyXEL* network.

C.1.1 ASUS PC

On Linux, a SSH connection can be established the following way:

```
user@client:~$ ssh pioneer@192.168.1.40
pioneer@192.168.1.40's password:$ pioneer
```

Once a connection is established, a ROS master has to be started with:

```
pioneer@192.168.1.40:~$ roscore
```

Next, another SSH connection has to be established. After the connection is established, a ROS launch-file can be used to bring up all necessary nodes on the ASUS PC. The launch-file can be started with:

```
pioneer@192.168.1.40:~$ roslaunch atv_ws/atv.launch
```

The content of the `atv.launch` file is shown below. The launch file starts the camera driver for the Bumblebee XB3, the camera processing node, and the common user interface interpreter.

```
1  <launch>
2    <!-- Bumblebee XB3 launch-->
3    <node pkg="camera1394" type="camera1394_node" name="bumblebeeXB3">
4      <param name="guid" value="00b09d0100b85670"/>
5      <param name="video_mode" value="1280x960_mono8"/>
6      <param name="iso_speed" value="400"/>
7      <param name="frame_rate" value="7.5"/>
8      <param name="bayer_pattern" value="gbrg"/>
9      <param name="bayer_method" value=""/>
10     <param name="pan" value="1"/>
11     <param name="camera_info_url"
12        value="file://$(find calib)/cal_center.yaml"/>
13   </node>
14
15   <node pkg="camera_proc" type="camera_proc" name="camera_proc"/>
16
17   <!-- user interface common interpreter-->
18   <include file="$(find user_interface)/launch/interpreter.launch"/>
19 </launch>
```

C.1.2 Acrosser Embedded PC

The procedure to launch all required nodes on the Acrosser Embedded PC is similar to the one used on the ASUS PC. First, a connection has to be established using SSH:

```
user@client:~$ ssh atv@192.168.1.52
atv@192.168.1.52's password:$ atv
```

Subsequently, a ROS launch-file can be used to start all nodes:

```
atv@192.168.1.52:~$ roslaunch catkin_ws/tracking.launch
```

The content of the `tracking.launch` file is shown below. The launch file starts the Velodyne ROS driver nodelets, the Velodyne processing node, the steering angle and speed control node, and the hybrid tracker.

```
<launch>
  <!-- Velodyne IMU and GPS data-->
  <node pkg="velodyne_imu" type="velodyne_imu" name="velodyne_imu"/>

  <!-- Velodyne Pointcloud-->
  <include file="$(find velodyne_driver)/launch/nodelet_manager.launch">
    <arg name="model" value="32E"/>
   </include>

  <include file="$(find velodyne_pointcloud)/launch/transform_nodelet.launch">
    <arg name="calibration" value="$(find calib)/32db.yaml"/>
    <arg name="frame_id" value="velodyne"/>
  </include>

  <!-- Velodyne Processing-->
  <node pkg="velodyne_proc" type="velodyne_proc" name="velodyne_proc"/>

  <!-- Control node and PID parameters-->
  <node pkg="control" type="control" name="control" output="screen">
    <remap from="/tracking/target" to="/tracking/soi"/>
  </node>
  <param name="loco_type" value="ackermann"/>
  <param name="P_v" value="1.0"/>
  <param name="I_v" value="0.02"/>
  <param name="D_v" value="0.0"/>
  <param name="P_a" value="1.0"/>
  <param name="nomDistAck" value="6.5"/>
  <param name="minNomDistAck" value="4.5"/>

  <!-- Hybrid Tracker -->
  <node pkg="tracker" type="tracker" name="hybrid_tracker" output="screen"/>
</launch>
```

C.2 User Interface Devices

Each of the implemented user interface devices needs to be setup correctly before it can be used. This section explains the easiest approach for each device.

C.2.1 PS3 Controller

In order to use the PS3 controller, it has to be connected to one of the PCs. Since the ASUS PC is mounted on top of the car, the Bluetooth receiver and the controller can be in a direct line of sight. It is therefore the preferred choice. First, the Bluetooth receiver has to be paired with the PS3 controller. The controller is connected via USB to the same PC as the receiver and the following command is used:

```
atv@192.168.1.52:~$ sudo bash
root@192.168.1.52:~$ rosrun ps3joy sixpair
```

If the setup is correct, the output will be something like this:

```
Current Bluetooth master: 00:21:17:ab:cd:09
Setting master bd_addr to 00:21:17:ab:cd:09
```

Once the receiver and the controller are successfully paired, the wired connection can be removed. The controller can now be wirelessly connected using the following command:

```
atv@192.168.1.52:~$ sudo bash
root@192.168.1.52:~$ rosrun ps3joy ps3joy.py
```

The PS3 controller can now communicate with the PC. However, in order to use the controller as input device for the system, the following lines have to be added to the corresponding launch file described in Section C.1:

```
1   <!-- PS3 Controller-->
2   <include file="$(find user_interface)/launch/ps3.launch"/>
```

C.2.2 Bluetooth Headset

The Bluetooth Headset interface does not depend on a specific headset. However, the device in use has to support the Headset Profile (HSP). First, the desired headset has to be paired with the Bluetooth receiver and connected. This can easily be achieved by using the user interface of the operating system. Once both devices are paired and connected, some configuration is necessary.

Next, the Headset Profile has to be activated in the `/etc/bluetooth/audio.conf` configuration file. This can be done by deactivating the Hands-Free Profile in the following lines:

```
user@client:~$ sudo nano /etc/bluetooth/audio.conf
...
# Set to true to support HFP, false means only HSP is supported
# Defaults to true
HFP=false
...
```

Subsequently, both, the MAC address of the used device and the device name have to be configured in the `headset.launch` file. The MAC address and the device name can be determined using the following command:

```
user@client:~$ pacmd list | grep -E 'name.*input|name.*source'
```

The output should look similar to this:

```
Default source name: alsa_input.pci-0000_00_1b.0.analog-stereo
        name: <alsa_input.pci-0000_00_1b.0.analog-stereo>
        name: <bluez_source.9C_3A_AF_A5_82_07>
```

The changes applied to the `headset.launch` file should look like this:

```
user@client:~$ nano catkin_ws/src/user_interface/launch/headset.launch
...
<param name="mic_name" value="bluez_source.9C_3A_AF_A5_82_07"/>
...
<param name="MAC" value="9C_3A_AF_A5_82_07"/>
...
```

Finally, the `headset.launch` file has to be included in one of the launch files described in Section C.1. In order to get audio feedback, the `soundplay.launch` file should be included, too:

```
1 <!-- Bluetooth headset with sound feedback-->
2 <include file="$(find user_interface)/launch/headset.launch"/>
3 <include file="$(find user_interface)/launch/soundplay.launch"/>
```

C.2.3 Android Smart Phone

In order to use the Android Smart Phone, the application has to be installed first. Therefore, the .apk-file has to be copied on the smart phone. Once the file is located on the phone, it can be installed by simply opening the file. It is important that installations from unknown sources are enabled on the phone. After installing the app, the user can connect to the ROS master by entering its address and pressing connect as shown in Figure C1. Additionally, the user can connect using a QR-Code that encodes the address. Once the connection is established, the smart phone can be used to interact with the robot.

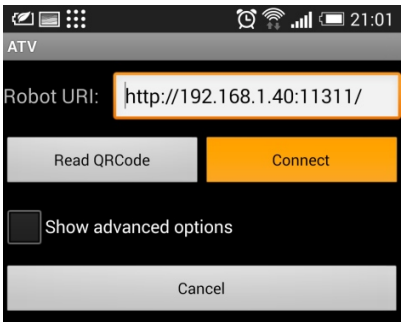


Figure C1: Connecting to the ROS master with the Android application.