Project no.:	**610917**
Project full title:	**Sustainable and reliable robotics for part handling in manufacturing**
Project Acronym:	**STAMINA**
Deliverable no.:	**D2.2**
Title of the deliverable:	**Robot Navigation System**

Contractual Date of Delivery to the CEC:	**31.10.2016**
Actual Date of Delivery to the CEC:	**31.10.2016**
Organisation name of lead contractor for this deliverable:	**Albert Ludwigs Universitaet Freiburg**
Author(s):	**Alexander Schiotka, Jörg Röwekämper, Denis Tananaev**
Participants(s):	**P04**
Work package contributing to the deliverable:	**WP2**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**14**
Start date of project:	**01.10.2013**
Duration:	**42 months – 31.03.2017**

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610917

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This deliverable is an outcome of Task 2.2 "Robot Navigation" and provides a description of all software components for the navigation part.

Keyword list: Navigation, Integration

Document History

Version	Date	Author (Unit)	Description
0.9	Sep 26, 2016	Alexander Schiotka, Denis Tananaev (ALU-FR)	Started document
1.0	Oct 19, 2016	Alexander Schiotka, Jörg Röwekämper (ALU-FR)	Finalized the document

Table of Contents

1 INTRODUCTION **4**
2 NAVIGATION **5**
2.1 TEACH AND REPEAT 5
2.2 AUTONOMOUS NAVIGATION 5
2.3 COMPARISON 6
TEACH AND REPEAT 6
AUTONOMOUS NAVIGATION 6
2.4 NAVIGATION ON RAILS 6
3 EXPERIMENTS **9**
4 LESSONS LEARNED **11**
4.1 KINEMATIC MODEL 11
4.2 DUST ON THE FLOOR 11
5 SUMMARY **12**
BIBLIOGRAPHY **13**

1 Introduction

The navigation of a mobile robot is a crucial skill, which is essential for the STAMINA project. Besides mapping and localization the robot needs to accurately navigate along a planned trajectory. This ability has a significant impact on the reliability and safety in warehouse environments. In the STAMINA project we build up on previously developed software and focused on the robustness and usability of the software by an external expert user. We developed a system that allows the user to create user defined paths on which the robot navigates accurately.
Figure 1 shows an abstract model of the mapping and navigation system. The left hand side of the figure shows all components of the mapping setup. The right hand side shows all components of the navigation module. This deliverable reports on the colored modules of the navigation setup without going into depth about the AVG and LiDAR driver which are discussed in Deliverable 2.1. The main focus of this deliverable is the path planning and controller module. The path planner plans a path from the current location to a goal location, which is then executed by the controller.

In summary, this deliverable reports on the work for task 2.2. The document describes the deployment of the software and robustness tests concerning the navigation behavior.

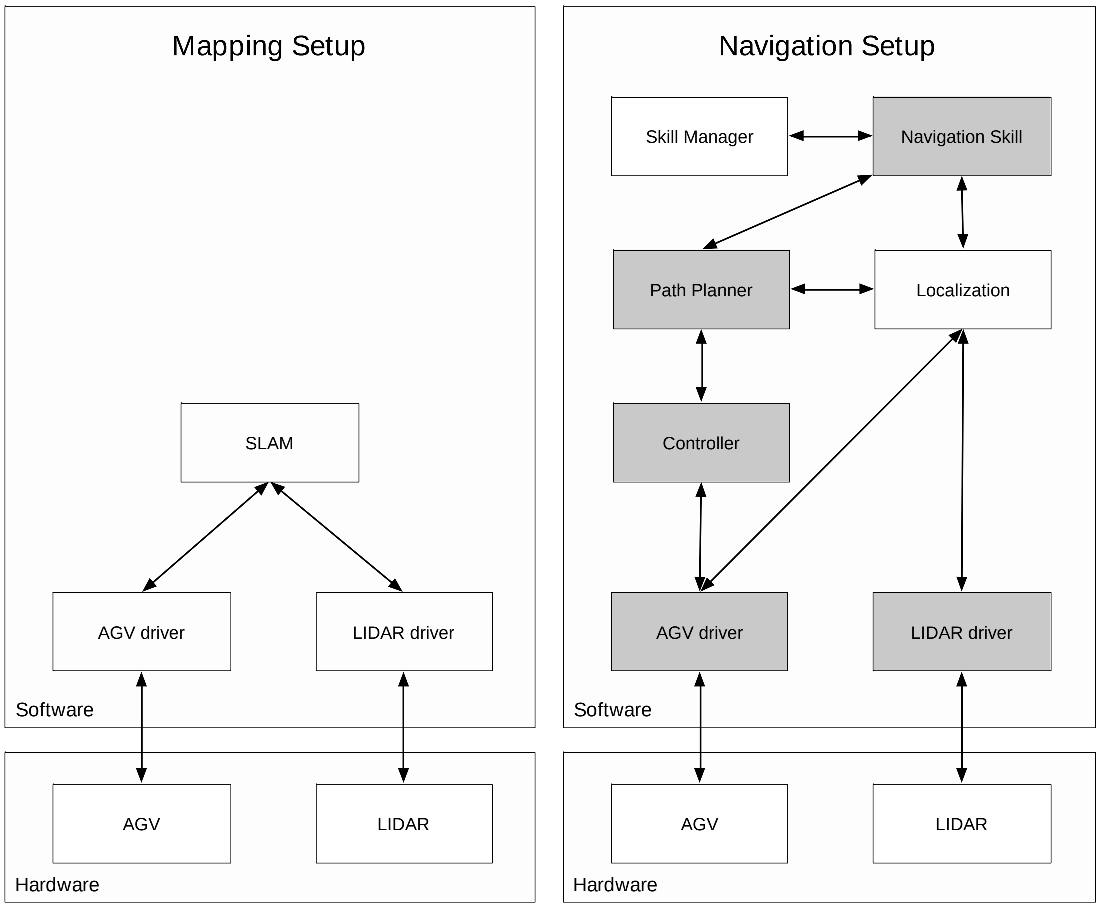


Figure 1: This deliverable embraces all colored modules of the illustration. They are part of the navigation process (right) where we navigate the robot autonomously. The Path Planner is responsible for finding the shortest trajectory on a previously defined graph whereas the Controller moves the robot. The drivers in both setups act as an interface for all other software components in the mapping and navigation modules.

2 Navigation

The project partners jointly defined multiple requirements regarding the navigation in warehouses. One characteristic of the system should be that the robot only navigates on previously defined paths, which increases the predictability of the robot's path for workers and forklift drivers. Also the area in which forklifts are allowed to drive is further restricted.
Since the user should have full control over the traveled trajectories of the robot, the navigation software needs a tool for building graphs. In terms of usability the user should be able to define these graphs without having physical access to the robot. Also because of changes in the environment (e.g. reconfiguration of shelves in a warehouse) the previously defined graph structure of the navigation should always be editable.
When navigating to a certain goal location in the environment, the robot receives a map coordinate. This goal location can have an arbitrary position on the graph since the picking stations can be freely defined. In order to determine a path to the goal location the robot invokes the planning algorithm.
The robot needs to operate in a long-term navigation scenario where the system has to work for hours robustly and accurately. The navigation system should avoid potential collisions and place the robot accurately in front of picking stations.
Two previously developed navigation systems have been developed and should be considered for the navigation task in the STAMINA project. The first system is a teach and repeat approach [1,2] where the robot follows a previously manually driven trajectory and the second system is an autonomous navigation approach which allows the robot to navigate freely in the whole environment and avoids dynamic obstacles [3,4].
Further on the two systems are shortly described in more detail and evaluated if they fulfill the requirements of the STAMINA project.

2.1 Teach and Repeat

Teach and repeat is an approach where a robot accurately follows a pre-taught trajectory. During a teaching phase, a non-expert user manually demonstrates a desired trajectory, which the robot can follow in the repeat phase. Instead of localizing against a globally consistent map, the robot uses the gathered distance measurements, e.g., laser scans as local maps, in order to localize and accurately follow the taught trajectory [1,2]. The approach utilizes scan matching to estimate the deviation of the robot’s position from points of the taught trajectory.
Teach and repeat fulfills the requirement of navigating on a predefined path. However, the main requirement of creating and adjusting the path without physical access to the robot is not possible since the approach only uses local instead of global maps for localization. The requirement of freely defined stops is fulfilled by this approach.

2.2 Autonomous Navigation

The autonomous navigation plans a path between the current location of the robot and the goal location. It generates a path, which avoids collisions with static and dynamic obstacles. It updates an obstacle map and replans the path to adapt to changes in the environment. It uses a localized robot in a global map and a trajectory controller to follow the planned trajectory.
In detail, the planner plans an initial path to the goal along the Voronoi graph created from an obstacle map, which contains all static obstacles as well as all dynamic obstacles with their current locations. This initial path is then optimized with the approach from Sprunk et al. [3,4]. The result of

the optimization is a smooth and continuous trajectory, which gets executed by a trajectory controller.
The approach allows the robot to navigate freely in the environment and does not restrict the robot's trajectory. Therefore, regarding the requirements defined above the autonomous navigation does not fulfill the constraint to predefined paths.

2.3 Comparison

Both approaches do not fulfill all requirements (see Table 1). Therefore, we decided to extend the approach of autonomous navigation since it is more flexible for adaptation and it is easier to further restrict the freedom of the navigation. The teach and repeat approach localizes the robot against single scans which represent local maps in the environment. It has the limitations that a robot is needed to create a new path or edit an existing one.
The newly developed approach is based on the previous presented autonomous navigation, which can navigate to arbitrary points in the environment. The approach uses a global map for which we developed an easy-to-use system for creating paths in this map. This enables a non-expert user to freely adjust the paths by defining waypoints with corresponding edges in the environment. All poses on the waypoints and edges are potential states of the robot.

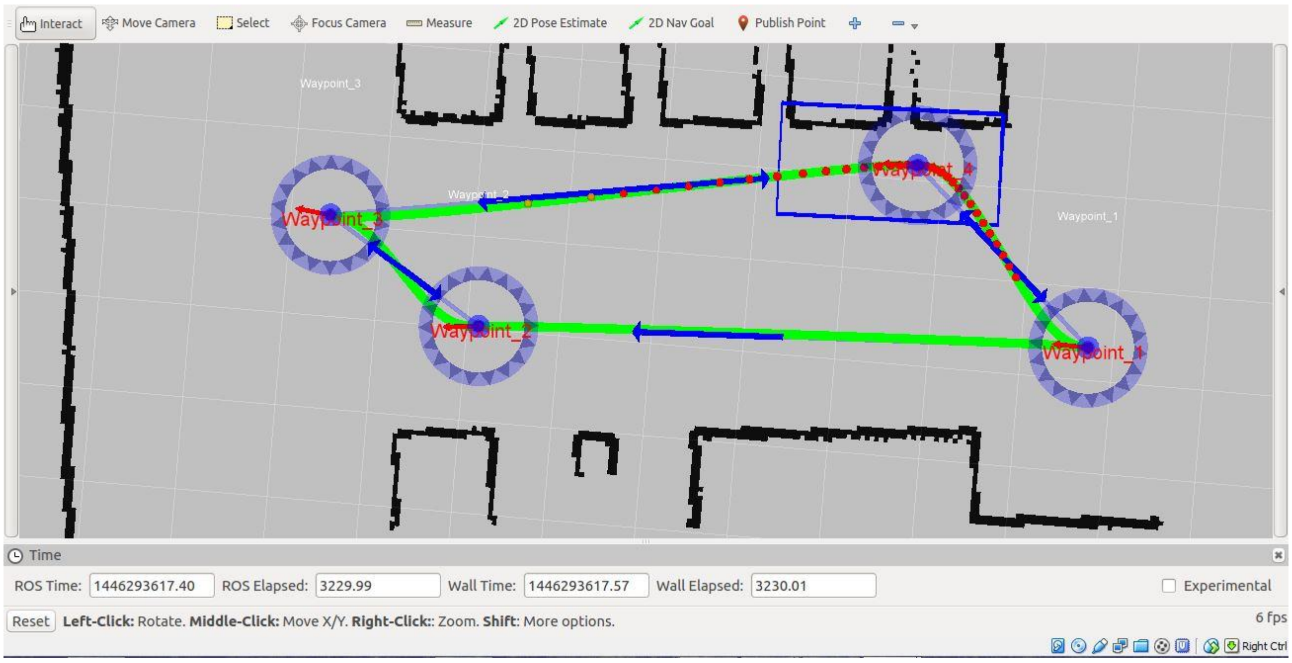
	Teach and repeat	**Autonomous Navigation**
Navigation on predefined path	Fulfilled	Not Possible
Design path on a remote PC	Not Possible	Not Possible
Edit path on PC	Not Possible	Not Possible
Freely defined stops	Fulfilled	Fulfilled

Table 1: Defined requirements for the two compared approaches of “teach and repeat” and “autonomous navigation”. The table shows whether the requirements are fulfilled for each specific approach.

2.4 Navigation On Rails

In order to satisfy the requirements of the project for a safe and robust navigation of the mobile platform in the warehouse we developed a technology to navigate the robot on predefined paths (virtual rails). The developed navigation on virtual rails is an extension of an autonomous navigation system adapted to the requirements of the STAMINA project.
The navigation rail is a directed graph structure in map coordinates. The vertices represent waypoints which are locations in the environment and the edges are paths between two vertices. The robot can travel along these paths from one vertex to another. We also constrained the orientation of the robot at each position on the graph. Furthermore it is possible to constrain the direction of traversability of each edge. We allow either only forward, only backward or forward and backward motions on an edge.

We provided a user friendly GUI which allows a non-expert user to define the navigation rail. It visualizes the smooth spline trajectories of the robot through the path and highlights dangerous parts of the trajectory or possible collisions with the environment depending on the clearance distance between robot and obstacles. Figure 2 shows an example where the red dots indicate possible collision with the static environment.



ROS Time: 1446293617.40 ROS Elapsed: 3229.99 Wall Time: 1446293617.57 Wall Elapsed: 3230.01 Experimental

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift:** More options. 6 fps

Figure 2 – The example of the navigation path. Waypoints represent specific robot poses: The location of the robot corresponds to the small blue circle at the center of the large blue ring. The orientation of the robot is indicated by the direction of the red arrow. Furthermore, the name of the waypoint is written on top of the waypoint. The light blue line indicates that there is an edge between two waypoints. The blue arrows on the line indicate the direction of the edge. In this case, the edges allow the robot to travel in both directions as there is an arrow in each edge pointing to either of the two waypoints. The green line corresponds to the actual trajectory that the robot would follow to move from the pose specified by the first waypoint to the pose specified by the other waypoint. The possible collision points on the trajectory path are visualized as red and yellow dots.

The input of the navigation system is a goal location in map coordinates. The goal and the current robot location get projected onto the graph. In a second step we search the graph for the shortest path from start to goal location. This path is then used as an initialization for the local path planner which optimizes the path and produces a collision free and continuous trajectory. The optimization criteria is the optimal travel time under consideration of the kinodynamic constraints of the robot, such as maximum velocities and accelerations. More information on the optimization can be found in Sprunk et al. [3,4]. An error-feedback controller executes the final trajectory and uses the current robot pose as the feedback signal. The path planning and optimization is continuously repeated during the navigation to a goal location to deal with deviations from the path and dynamics in the environment. Figure 3 shows an overview of the planning and execution module.

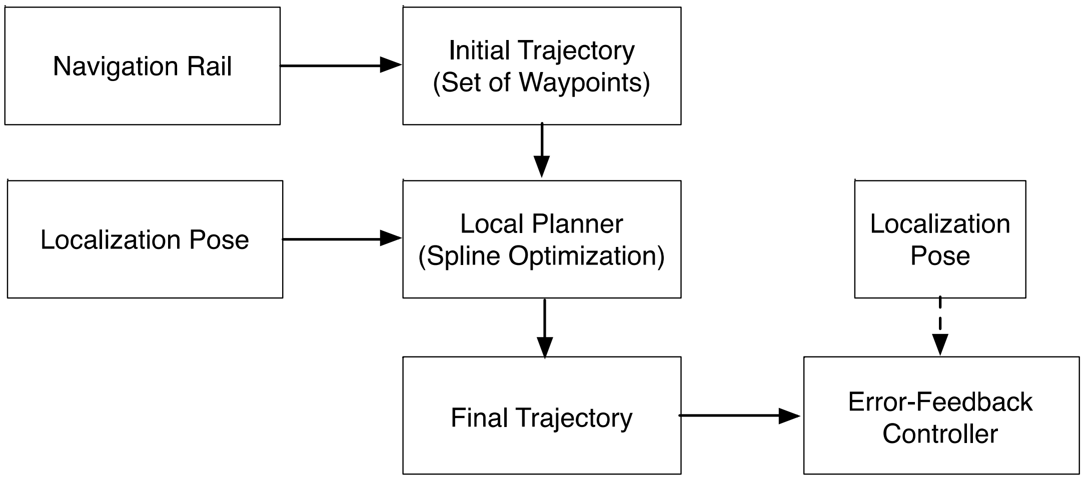

Navigation Rail
Initial Trajectory (Set of Waypoints)
Localization Pose
Local Planner (Spline Optimization)
Localization Pose
Final Trajectory
Error-Feedback Controller

Figure 3 – System overview. The navigation rail is used to generate an initial trajectory from the current robot position to the goal location. The local planner optimizes the trajectory in order to reduce the travel time to the goal. An error-feedback controller executes the final trajectory while the localization pose is used as a feedback.

Additionally we check and correct the orientation of the robot at a goal location. This is important in particular at picking locations where a small orientation error can lead to objects being out of view of the cameras. This would then result in failures regarding the object detection.
The full autonomous navigation skill which moves the robot without being constrained to the graph is still available and could be used to navigate around obstacles which are on the graph. This feature is at the moment not used due to the requirement of moving only on defined paths.

3 Experiments

We conducted multiple experiments in order to ensure robustness and reliability of the developed navigation software. The experiments took place at an experiment area of University Freiburg. We used the AGV produced by BAS and recorded the following properties: the duration of the experiments, how often a robot travelled to a target location, and how often it failed to reach a target location.

The total time of the experiments was approximately 14 hours. Within this time period we navigated the robot to randomly chosen waypoints on the graph. The sampling of these random goals only took place along the two edges in front of the simulated picking stations that are visualized in Figure 4. In order to perform curves, we used two different navigation paths with 90 degree rotations and both 90 and 45 degree rotations (see Figure 4).

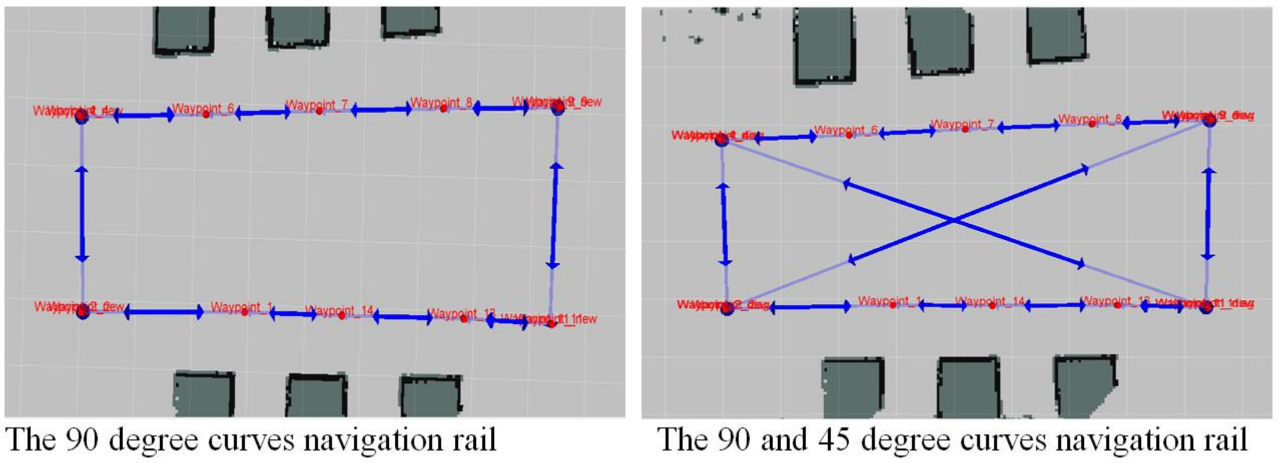

Waypoint_6
Waypoint_7
Waypoint_8
Waypoint_1
Waypoint_14
Waypoint_13
The 90 degree curves navigation rail
The 90 and 45 degree curves navigation rail

Figure 4 – The navigation paths with 90 degree curves (left) and 90 and 45 degree curves (right) used for long-term navigation experiments in Freiburg. The red dots represent the waypoints of the navigation path and the blue lines represent the edges. The dark blue arrows at the lines show the allowed motion direction (in this cases it is always forward and backward). We set up an aisle of a warehouse with three pallets from both sides.

The localization robustness experiments were performed with a navigation system which was still under development. We carried out all navigation robustness tests with a maximal linear velocity of 0.8 m/s and 1.0 m/s. In all cases, the linear acceleration and deceleration was 0.1 m/s. Furthermore we added an additional weight of 216 kg to the robot to mimic the conditions of the experimental setup in Rennes. All results of the previous experiments for the localization robustness and the current navigation robustness are presented in Table 2.

In the first navigation experiment we observed three failures in three hours of navigation. After adjusting control parameters we were able to achieve a robust navigation behavior. The robot was able to navigate on the virtual rail for approximately 11.65 h and reached 930 randomly chosen goal locations without any navigation failures or localization failures.

The comparison of the navigation experiments from October 2016 with the localization experiments in February 2016 show that the continuous development and improvement of the navigation system lead to a more robust and stable navigation module.

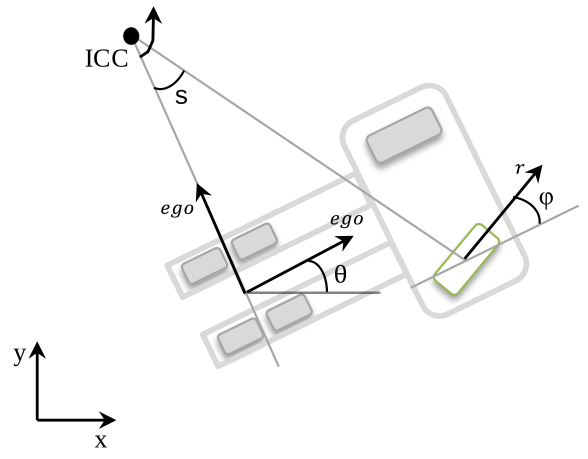
Experiment	Velocity	Payload	Duration	Targets	Failures in total	Navigation Failures
Localization Robustness Test in February 2016						
1	0.5 m/s	0 kg	5.25 h	565	2	2
2			3.5 h	260	5	5
3			3 h	200	3	3
4			3 h	200	2	2
Navigation Robustness Test in October 2016						
1	0.8 m/s	216 kg	3 h	260	3	3
2			2 h	200	0	0
3			4 h	300	0	0
4			1 h	70	0	0
5			3.5 h	250	0	0
6	1.0 m/s		1.15 h	100	0	0

Table 2 – Results of the navigation experiments at Freiburg performed with the AGV from BAS in February and October 2016. After adjusting the control parameters of the system we observed no navigation failures in the October experiment runs which had a duration of approximately 11.65 h with 920 randomly chosen goals.

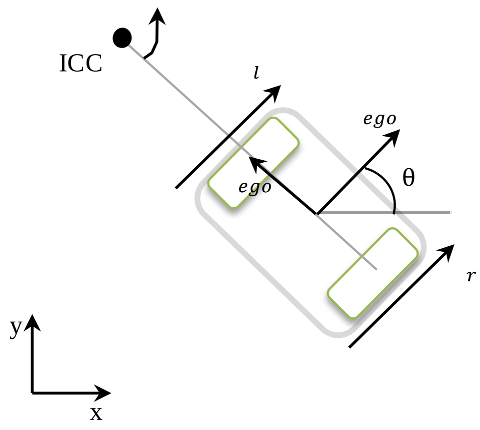
4 Lessons learned

4.1 Kinematic Model

During the tests in February 2016 we had some issues with the controller related to a deviation from the predefined trajectory. Because of the approximation of the kinematic model the adjustment of the planning- and control-software was more difficult. We solved this problem and achieved a robust behavior by setting up proper planning and control parameters. Furthermore we added an angular correction of the robot orientation with respect to the navigation path at the beginning and at the end of the motion trajectory. The robustness and reliability of the navigation module allowed us to double the maximum speed of the robot from 0.5 m/s to 1.0 m/s.
The provided control interface for the hardware is the differential drive model, which is a simplification of the existing kinematics of the robot (see Figure 5). For future development we believe considering the real kinematic model (e.g. Ackermann drive model) in the path planning and trajectory following could allow a higher navigation speed.



Schematic representation of the real kinematics of the robot



Schematic representation of the differential drive kinematical model

Figure 5 – Schematic representation of the real kinematics of the robot (left) and the differential drive model (right). The green color highlights the driving wheels while gray color highlights passive wheels. The following notations are used: ICC – instantaneous center of curvature; w – angular velocity; v – linear velocity; vr and vl – velocity of the right and left wheel; θ – orientation of the robot with respect to the global coordinate system; φ – driving wheel angle.

4.2 Dust on the floor

The test sprint in Rennes revealed an undesired navigation behavior. Randomly during operation the robot suddenly stopped or started jerking. We found out that this is caused by dust on the floor, which gets swirled up by the robot's motion. It gets detected by the safety laser scanners, which activate the hardware emergency stop. In order to solve this issue we cleaned the floor of the experimental area. An alternative option would be the adaptation or partially deactivation of safety fields, which is not allowed due to safety standards.

5 Summary

This document gave an overview on the results of Task 2.2. The goal was to develop a navigation system for the AGV. We adapted already existing software to the use-case of navigating on virtual Rails (previously defined path) that can be created by a non-expert user. The developed system navigates accurately and reliably over hours of operation as shown in the experiments.

Bibliography

[1] C. Sprunk, G. D. Tipaldi, A. Cherubini and W. Burgard, Lidar-based Teach-and-Repeat of Mobile Robot Trajectories, Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013

[2] M. Mazuran, C. Sprunk and W. Burgard and G. D. Tipaldi, LexTOR: Lexicographic Teach Optimize and Repeat Based on User Preferences, Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2015

[3] C. Sprunk, B. Lau, and W. Burgard, Improved Non-linear Spline Fitting for Teaching Trajectories to Mobile Robots. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2012

[4] C. Sprunk and B. Lau and P. Pfaff and W. Burgard, An Accurate and Efficient Navigation System for Omnidirectional Robots in Industrial Environments, Autonomous Robots, 2016

Abbreviations

AGV	Automated Guided Vehicle
BAS	BA Systèmes
GUI	Graphical User Interface
ICC	Instantaneous Center of Curvature