Route programming on a mobile barn cleaner

By

Jan Pieter Mars

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

At the

University of technology Delft

2014

Committee

Daily suppervisor	**Responsible professor**
Martijn Buijs (Lely)	Dr. K.V. Hindriks

Committee members

Prof. Dr. C.M. Jonker - Full Professor, Interactive Intelligence

Dr. K.V. Hindriks - Assistant Professor, Interactive Intelligence

Dr. C.J.M. Verhoeven - Associate Professor, Microelectronics

Martijn Buijs - Manager Product Development at Lely Industries NV

Abstract

The ‘Lely Discovery’ is an automated mobile barn cleaner, which routes are programmed by a ‘teaching method’. The routes are specified by a linear action sequence to be executed by the robot. Teaching a route to the Lely Discovery requires users to execute every action first by walking the robot through the barn.

The route teaching method is time consuming and the current interface of the teaching method is not user-friendly. When a route requires changes for optimizing the route, the user has to re-execute all the route actions on the robot to and from the position where he wants to change the route.

This thesis explores the implementation of a route drawing system which enables users to draw routes on a map. This system can generate an action sequence from the drawn routes on the map. Users can program routes 2 – 3 times faster than with the teaching method and route changes can be made in short notice.

In this thesis the results of a user study are presented and the results of the tested routes on the robot via a first prototype of the drawing system. Although the results are not convincing, they provide useful leads for the design of a second prototype.

Table of Contents

Committee ... 2

Abstract ... 3

Table of Contents ... 4

List of figures ... 9

1 Introduction ... 12

1.1 Situation Sketch ... 12

1.2 Technical specifications Lely Discovery ... 12

1.3 Route programming ... 13

1.4 Requirements for a Good Route ... 15

1.5 Interface ... 16

1.6 Research question ... 17

2 Related work ... 19

2.1 Map making ... 19

2.2 Route teaching ... 21

2.3 Route planning ... 22

2.4 Route preferences ... 23

2.5 User interface ... 24

2.5.1 Gestures ... 24

2.5.2 Sketch ... 24

2.6 Route feedback ... 28

2.7 Conclusion ... 29

3 Field Research ... 30

3.1 Point learned during the field research ... 30

3.2 Conclusion ... 31

4 Improving the current system or making a new system ... 32

5 Prototype route drawer ... 33

5.1 Interface route drawer ... 33

5.2 Functionality route drawer ... 33

5.2.1 Map Maker ... 33

5.2.2 Specify Walls ... 34

5.2.3 Route Maker ... 34

5.2.4 Feedback system 37
5.3 System data flow 38
6 Requirements route drawer 41
6.1 Assumptions and requirements for the Lely Discovery 41
6.2 Map drawer 42
6.3 Route drawer 42
6.4 Route feedback 43
6.5 Action output 43
7 Scope route drawer 44
7.1 Barn layout 44
7.2 Lely Discovery design 46
7.2.1 Control points 47
7.3 Input for algorithm 48
7.3.1 Barn map 48
7.3.2 Route waypoints 48
8 Algorithm design for generating an route action sequence 50
8.1 Output 50
8.2 Search algorithms 50
8.3 Case based algorithms 51
8.4 Pseudocode 53
8.5 Route cases 54
8.6 Action sequence evaluation 58
8.7 Conclusion 62
9 Test plan 63
9.1 Pilot 63
9.2 User Study 63
9.2.1 Participants 64
9.2.2 General objective 64
9.2.3 Task 64
9.2.4 Preparation 64
9.2.5 Procedure 64
9.2.6 Instructions and questions 65

9.3 Routes from users on robot 65
9.3.1 General objective 65
9.3.2 Task 66
9.3.3 Preparation 66
9.3.4 Procedure 66
10 Evaluation 67
10.1 Background information participants 67
10.2 Route results 67
10.2.1 Route changes 68
10.2.2 Results 68
10.2.3 Route problems 69
10.2.4 Conclusion 71
10.3 Route draw and drive time 71
10.4 Route drawing on paper compared 73
10.5 Area cleaned 73
10.6 System usability 74
10.6.1 Results 74
10.6.2 Conclusion 76
10.7 Open questions 77
10.8 Component-Based Usability Questionnaire 77
10.8.1 Reliability Usability scales 77
10.8.2 Usability rating - Hypothesis testing Component rating 78
10.8.3 One-Sample t-Test 78
10.9 Interaction between background and results 79
10.9.1 Interaction between education and current system 80
10.9.2 Interaction between education and drawing system 82
10.9.3 Interaction between education, route success and draw time 83
10.9.4 Conclusion 85
11 Conclusion and recommendations 86
12 Future work 87
12.1 Adding cases 87
12.1.1 Learning from new cases 87

12.1.2 Opaque relations 88
12.2 Generate automatically routes 88
13 Bibliography 89
14 Appendix A 92
14.1 Requirements Lely Discovery 92
15 Appendix B 94
15.1 Usability test of route programming on the Lely Discovery 94
15.1.1 Introduction 94
15.1.2 Background 94
15.1.3 Part 1 Current route programming with the E-Link or Smartphone 94
15.1.4 Part 2 Instructions and questions about the new route drawing tool 96
15.1.5 Map 1 101
15.1.6 Map 2 102
15.1.7 Map 3 103
15.1.8 Map 4 104
15.1.9 Map 5 105
16 Appendix C 106
17 Appendix D 109
17.1 Correlation between factors questions and results 109
17.2 Answers open questions current programming method 116
17.3 Answers open questions route drawing method 118
18 Appendix E 122
18.1 Solutions 122
18.2 Building global map 122
18.3 Building local map 122
18.4 Route programming 123
18.5 Interface 124
18.6 Feedback 125
18.7 Solution comparison 125
18.8 Conclusion and decision 129
18.9 Research computer aided route design method 131
18.10 Route programming with CARD tool 131

18.11 Algorithm to build the action sequence 131
18.12 User study 131
18.13 Map making and route programming by drawing 132

List of figures

Figure 1 Lely Discovery Mobile Barn Cleaner (a) on the charger (b) cleaning 12
Figure 2 LELY Discovery dimensions with on the left the charger 13
Figure 3 The E-Link manual controller for programming the routes of the Lely Discovery 13
Figure 4 Route 1, clean all areas behind the cubicles each hour 14
Figure 5 Example actions 14
Figure 6 The E-Link Manual Controller menu structure 16
Figure 7 Turn right action options from the menu 17
Figure 8 An integrated framework of multiple, disparate representations of spatial knowledge. Each level of abstraction uses its own ontology with concepts motivated by human communication about spatial navigation (Beeson, et al., 2007, March) 20
Figure 9 The structure of SOM 21
Figure 10 Voronoi diagram 22
Figure 11 A high-level block diagram of the Crusher Autonomy system (Silver, Bagnell, & Stentz, 2010) 22
Figure 12 Safety grid built around every human in the environment. This depends highly on the human's posture. As the person feels less "threatened" when standing, the value and the range of the costs are less important. (c) Decreasing costs attributed to the zones hidden by obstacles. The supplementary costs discourage the robot getting too close to the obstacles and thus, prevents the robot from appearing suddenly behind hidden places (Sisbot, Marin-Urias, Alami, & Simeon, 2007) 23
Figure 13 Learning process (Park, Bell, Kaparias, & Bogenberger, 2007) 24
Figure 14 Waypoints (*) are extracted at points along the path that intersect with lines in the Voronoi–Delaunay graph (Shah & Campbell, A qualitative path planner for robot navigation using human-provided maps, 2013) 25
Figure 15 Example of a two-gesture stroke modeled using a variable duration hidden Markov model, where white nodes represent hidden (unobserved) variables, shaded nodes are observed variables, and arrows represent conditional dependencies. In this example, the first gesture g_1 has a duration of d_1 = 3 strokes, and the second gesture g_2 has a duration of d2 = 2 strokes. Interstroke nodes $i^1$1 , $i^2$1 , and $i^1$2 represent self-transitions on the gestures, while node $i_{1:2}$ represents a transition from gesture g1 to gesture g2. (Shah, Schneider, & Campbell, A robust sketch interface for natural robot control, 2010, October) 26
Figure 16 Sketch 1 (a) A route map sketched on a PDA. (b) The corresponding digital representation (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004) 27
Figure 17 Force histograms. (a) the scalar resultant of forces (black arrows). Each one tends to move B in direction. (b) The histogram of constant forces associated with (A,B), i.e., the position of A relative to B. (c) The histogram of gravitational forces associated with (A,B) (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004) 27
Figure 18 Synoptic diagram showing how spatial information is extracted from the sketch (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004) 28
Figure 19 The collaborative control system model (Fong, Thorpe, & Baur, 2003) 28
Figure 20 Interface design 33
Figure 21 the robot should drive with the center point of the robot to every waypoint. 35

Figure 22 Path segment aligned with half the width of the robot from the wall 35
Figure 23 path segment aligned with the width of the robot minus a margin for overlap with another route path segment 36
Figure 24 waypoints are horizontal and vertical aligned 36
Figure 25 waypoint drawn in the charger 36
Figure 26 example route where the route start and ends in the charger 37
Figure 27 feedback after calculating the route. The last parts or colored red, because the route line is drawn to close to the wall and the corner 38
Figure 28 Data flow diagram 39
Figure 29 action types for the Lely Discovery 41
Figure 30 minimal distances between objects for the ideal barn 44
Figure 31 2 + 2 with 2 rows of cubicles on one side of the feeding path and 2 rows of cubicles on the other side 45
Figure 32 4 + 4 barn with 4 rows of cubicles on one side of the feeding path and 4 rows of cubicles on the other side (one side displayed) 45
Figure 33 Lely Discovery turns from the wall without getting stuck on the wall 46
Figure 34 control points used for executing the actions 47
Figure 35 basic structure of a rule-based reasoning system (Negnevitsky, 2005) 52
Figure 36 pseudocode for generating the route actions 54
Figure 37 (a) generated action sequence which needs to be evaluated, (b + c) route goes wrong in practice (d) desired action sequence 58
Figure 38 complex corner in the workshop 68
Figure 39 Route 13 in map 1 where the robot crashed, right the simulation where the robot should end after the move forward action for 8 cm 70
Figure 40 Route 11 in map 2 where the robot did not get inside the cubicle 70
Figure 41 Route 2 in map 4 where the robot did not get inside the cubicle 71
Figure 42 Interaction between computer experience and the draw time for map 4 73
Figure 43 means usability questions of both systems 76
Figure 44 79
Figure 45 Interaction between experience with programming routes on the Lely Discovery and current system, a = Overall, I am satisfied with how easy it is to use this system, b = It was simple to use this system, c = I am able to complete my work quickly using this system 81
Figure 46 Interaction between diploma and current system, a = I feel comfortable using this system, b = The interface of this system is pleasant 82
Figure 47, a = It was easy to learn to use this system, b and d = I believe I became productive quickly using this system, c = This system has all the functions and capabilities I expect it to have 83
Figure 48 a-c = Interaction between computer experience (education 1) and draw time and route success. d = Interaction between draw time and experience with drawing tools. 84
Figure 49 Interaction between draw time and computer experience (education 1) and diploma (education 5). 84
Figure 50 The CBR cycle 87
Figure 51 Test 1, normal route 106

Figure 52 Test 2, turn at end of alley 106
Figure 53 Test 3, bouncing with different angels on the walls 107
Figure 54 Test 4, bouncing with different angels on the walls 107
Figure 55 Test 5, route drawn by Mauro 108
Figure 56 Decision tree with possible solutions 128

1 Introduction

In this chapter Lely and one of their products, the Lely Discovery are introduced. After introducing the Lely Discovery, the specifications of the Lely Discovery are shown and how routes can be programmed on the Lely Discovery. Then some requirements for a good route are listed and is explained how the interface works on the device which is used to program the routes. Finally the problematics of the current route programming method are shown and the research question is presented.

1.1 Situation Sketch

Lely is a Dutch manufacturer of agricultural machinery based in Maassluis. This agro-technical company mainly produces machines for the cattle industry such as lawn mowers for pastures, feeding- and milking machines, including automatic milking- and feeding robots. One specific machine is the Lely Discovery. The Lely Discovery is a battery-driven barn cleaner, which is able to ride and clean pre-programmed routes automatically. The robot cleans the slatted floors of a barn to ensure that the cows can walk on a clean and dry floor. Quick disposal of manure increases the hygiene and the well-being of milking cows, their health and milk quality.

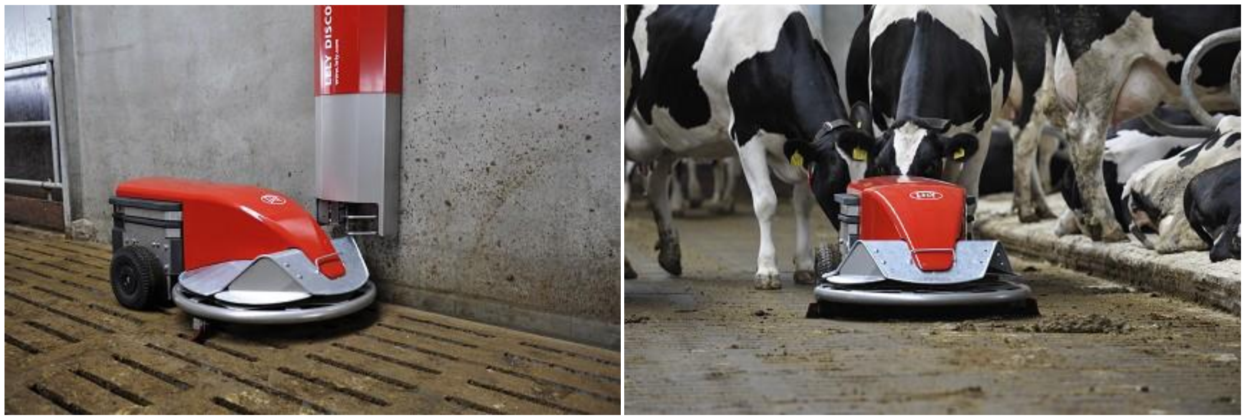


Figure 1 Lely Discovery Mobile Barn Cleaner (a) on the charger (b) cleaning

1.2 Technical specifications Lely Discovery

Technical specifications of the Lely Discovery from the website of Lely:

Weight (kg)	270
Empowered by	2 electric motors
Adjustable travel speed	9-18 m/min
Voltage	Gel battery 12 Volt
Determination of direction	Free spinning horizontal guiding wheel, gyroscope and ultrasound on the left
Floor	Slatted floor with max. 3 degrees of slope
Capacity	ca. 240 cows
Minimum height of cubicle floor	12.50 cm as a minimum

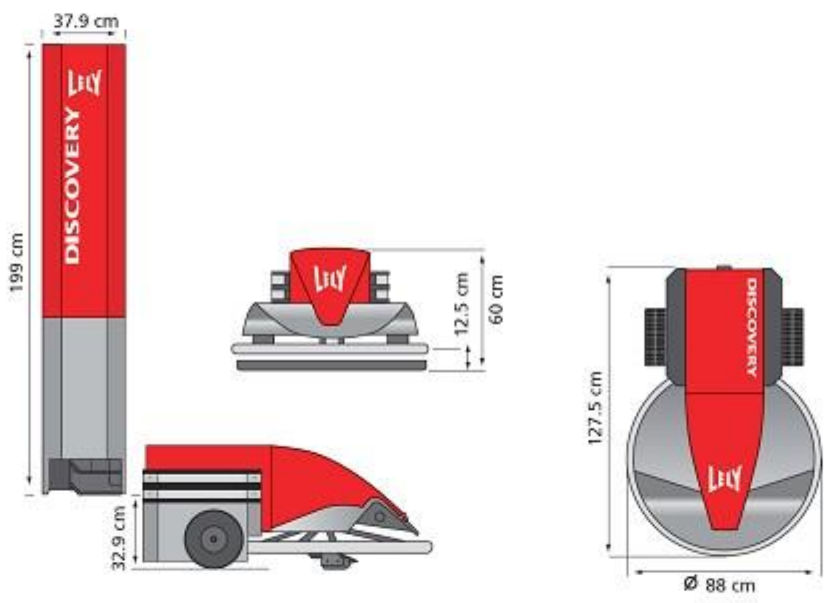
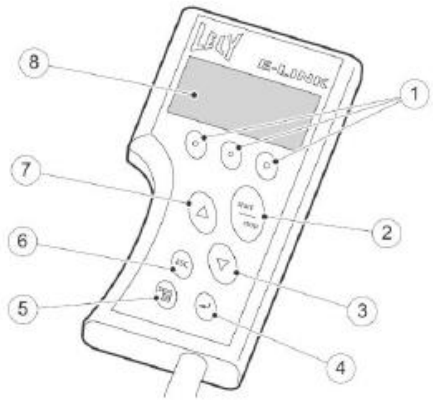


Figure 2 LELY Discovery dimensions with on the left the charger



The E-Link (page 4-11) has the following buttons:

- Soft buttons (1) execute the command on the LCD display above the applicable button
- Button (2) starts or stops an action
- Button (3) moves the selector down one item or decreases a value by one
- Button (4) opens the selected function or the next menu screen
- Button (5) reduces the speed of the vehicle during programming
- Button (6) returns to the previous screen
- Button (7) moves the selector up one item or increases a value by one

Display (8) displays menus and texts.

Figure 3 The E-Link manual controller for programming the routes of the Lely Discovery

1.3 Route programming

The routes of the Lely Discovery are specified by a linear action sequence to be executed by the robot. Every route starts and ends at a charger which loads the battery of the Lely Discovery. Every action is specified by a distance (total turns of the wheels), angle, stopping condition like bumping into a wall, and distance between the left side of the robot and the wall. The distance between the robot and the wall is measured by a sonar which is able to detect walls higher than 12.5 cm.

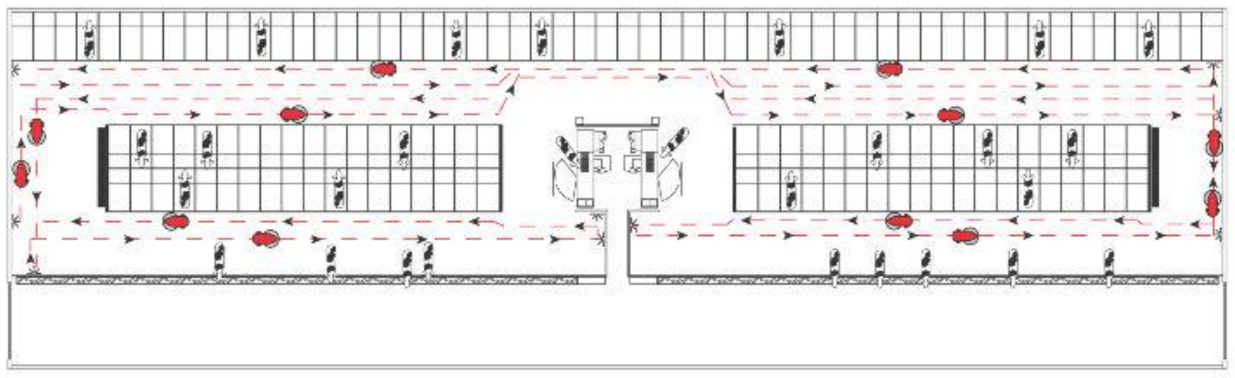


Figure 4 Route 1, clean all areas behind the cubicles each hour

Teaching a route to the Lely Discovery requires the user to execute every action. He can guide the robot with a smartphone or the E-link (see Figure 3). The technician or farmer can enter different actions which can be selected from a menu, for example: 'go straight forward', 'follow the wall' or 'make a curve'. Before a route is programmed, the route can be prepared on beforehand (on a piece of paper) or by just looking around and think about the route you want to program. When the route is conceived, it must be programmed by letting the robot ride the whole route, while instructions are given with the E-link or the Smartphone (see Figure 4 for an example route). Every route is built up by many actions (see Figure 5 and Table 1).

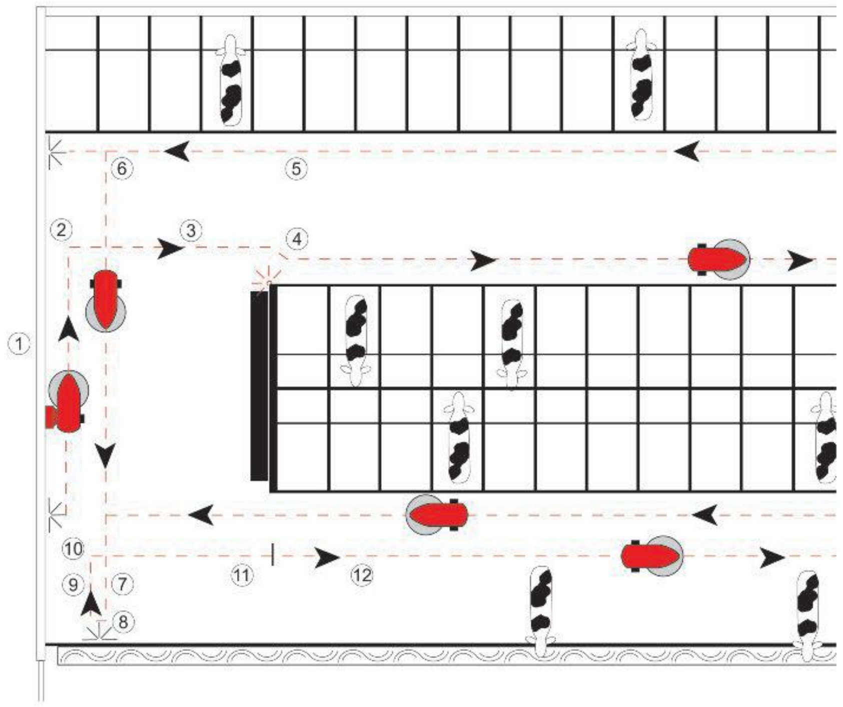
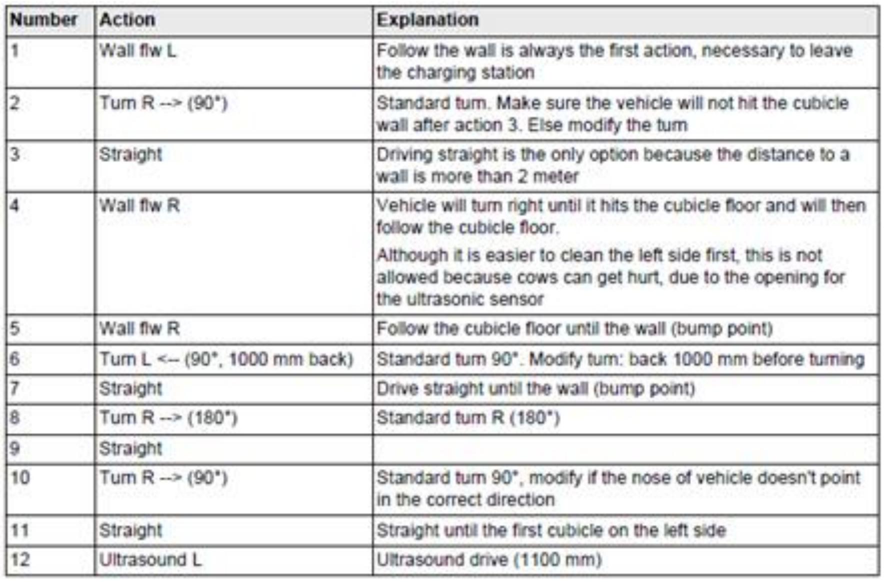


Figure 5 Example actions

Table 1 explanation of each action in Figure 5



Number	Action	Explanation
1	Wall flw L	Follow the wall is always the first action, necessary to leave the charging station
2	Turn R --> (90°)	Standard turn. Make sure the vehicle will not hit the cubicle wall after action 3. Else modify the turn
3	Straight	Driving straight is the only option because the distance to a wall is more than 2 meter
4	Wall flw R	Vehicle will turn right until it hits the cubicle floor and will then follow the cubicle floor. Although it is easier to clean the left side first, this is not allowed because cows can get hurt, due to the opening for the ultrasonic sensor
5	Wall flw R	Follow the cubicle floor until the wall (bump point)
6	Turn L <-- (90°, 1000 mm back)	Standard turn 90°. Modify turn: back 1000 mm before turning
7	Straight	Drive straight until the wall (bump point)
8	Turn R --> (180°)	Standard turn R (180°)
9	Straight	
10	Turn R --> (90°)	Standard turn 90°, modify if the nose of vehicle doesn't point in the correct direction
11	Straight	Straight until the first cubicle on the left side
12	Ultrasound L	Ultrasound drive (1100 mm)

1.4 Requirements for a Good Route

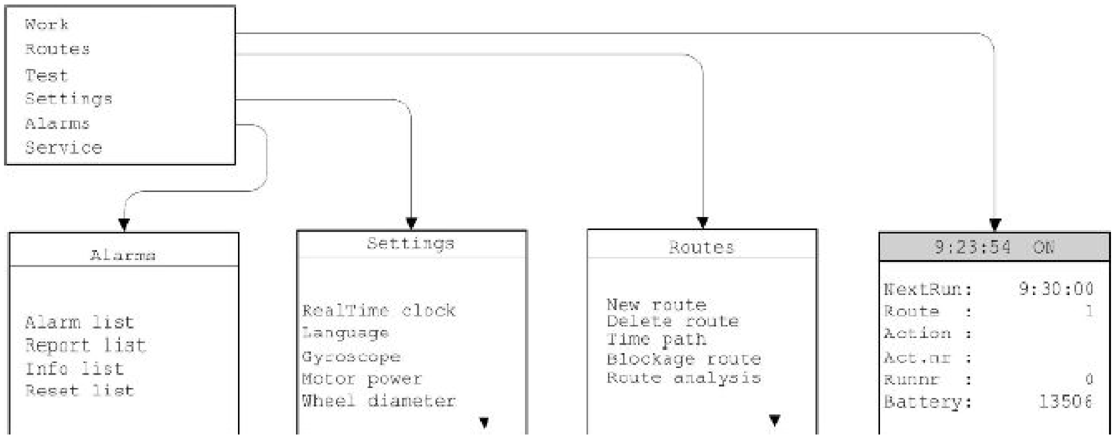
When the robot drives a route, many things can occur which can cause the robot to go in the wrong direction. Wheels might slip on the ground which can cause the direction of the robot to change and the wrong distance being measured. There is also the problem of noise in the data coming from the sensors. Sonar is not very precise and contains noise on the signal. The barn cleaner has to operate between cows. Cows might block the route by walking or lying in front of the barn cleaner. This could cause the stopping condition to be triggered too early or the sonar to obtain wrong information about the distance to the wall. If a specific action cannot being executed successfully, completing an entire route is problematic, because the robot has no map of his environment and hence no idea of his position. When a robot cannot finish his route, he stops and beeps until he is brought back manually to the charger. To make a good route and to increase the reliability of the route, Lely made the following route requirements for its customers.

Some requirements from the manual for a good route:

- Include as many bump points against the walls as possible in all directions. If, due to slipping, the bump point is not reached in time, the Lely Discovery continues until the bump point is reached. This improves the reliability of the route.
- Be aware that actions like 'turn' and 'straight' may cause inaccurate results. Make sure the vehicle cannot get stuck after making a turn or driving straight. Program a 'long action' (wall contact: 'wall follow' or 'ultrasound') after a 'short action' (no wall contact: 'turn' or 'straight').
- When moving straight forward, the deviation in direction can be more than 10%. Take this into account when setting up a route. If possible, include a bump point after going straight.
- If the angle between the Lely Discovery and the wall is more than 30 degrees, the angle between the direction of the robot and the direction of the wall, you must program a 'turn' action first before you program a 'follow wall' action.

- If the vehicle follows the wall or cubicle row ('wall follow' or 'ultrasound') and if there is a bend, stop the vehicle, program a turn and go on to follow the wall.
- Try to avoid to move past the cows on the left side, they may get hurt due to the opening for the ultrasonic sensor.
- Before starting, make sure the Lely Discovery points in the correct direction. Use the ''Straight' option as little as possible. If it really is necessary, for example to get the Lely Discovery through a doorway, make the distance travelled as short as possible. Once through the doorway, continue to follow a wall left or right, or ' Ultrasound Left'. This improves the accuracy of the route that is travelled.
- You can choose from 12 standard turns and modify the turn afterwards. Preferably do not make sharp turns in heavy polluted areas (less power).
- Do not program Ultrasound Left when the wall is more than 2 meters away.
- The Lely Discovery must not drive more than 40% of the total time in a time path. It must spend minimum 60% of the total time charging at the charging station.
- Make a backup each time you changed a route or time path or installed a new E-Link. Backups are made on the E-Link.
- Program routes from different positions in the barn back to the charging station. This will enable you to easily return the vehicle back to the charging station in case it gets lost.
- The beep can be modified separate for each route. If the beep length is set to 0, the beep is switched too OFF. If a warning signal is given, the cows have time to move out of the way. However, the sound may equally well disturb other cows.
- Set the speed to a lower level for parts of the route or the whole route if the cows are disturbed.
- The water tank capacity is 30 L (7.9 gal), which is enough for approximately 30 minutes of spraying. As a result the Lely Discovery must return to the water filling station (and the charging station) for a refill.

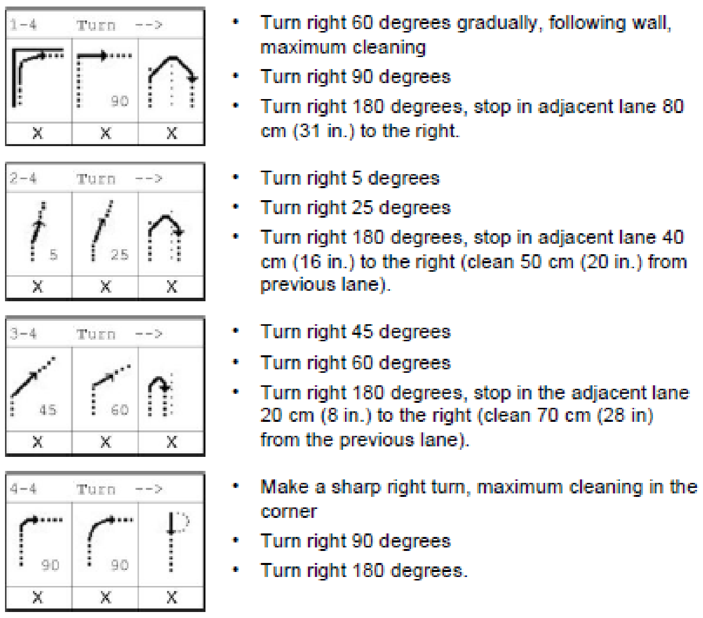
1.5 Interface



```
9:23:54  ON
NextRun:    9:30:00
Route   :         1
Action  :
Act.nr  :
Runnr   :         0
Battery:      13506
```

Figure 6 The E-Link Manual Controller menu structure

Route programming is done with the E-link or with a Smartphone. Route and actions can be selected from a menu (see Figure 6 for a part of the menu structure). To program a new route, you have to select ‘route’ from the menu. When pressed on the start button, the following actions can be selected: ‘wall follow left’, ‘wall follow right’, ‘straight’, ‘turn right’, ‘turn left’, ‘ultrasound left’, ‘follow wall to charger’. If an action is selected, a new menu opens with new options, (e.g. Figure 7). At the moment an action is selected, the robot executes the action. When the robot goes straightforward, users must press the start/stop button until the robot arrives at the right place for the next action. During the driving, parameters are set such as the distance. If the robot will bump on a wall within 20 cm, the user should press on a bump button (button 5 in Figure 3). This will cause the robot to drive slowly until it bumps on the wall, this way the stopping condition for that action is set as bump point.



- Turn right 60 degrees gradually, following wall, maximum cleaning
- Turn right 90 degrees
- Turn right 180 degrees, stop in adjacent lane 80 cm (31 in.) to the right.

- Turn right 5 degrees
- Turn right 25 degrees
- Turn right 180 degrees, stop in adjacent lane 40 cm (16 in.) to the right (clean 50 cm (20 in.) from previous lane).

- Turn right 45 degrees
- Turn right 60 degrees
- Turn right 180 degrees, stop in the adjacent lane 20 cm (8 in.) to the right (clean 70 cm (28 in) from the previous lane).

- Make a sharp right turn, maximum cleaning in the corner
- Turn right 90 degrees
- Turn right 180 degrees.

Figure 7 Turn right action options from the menu

When the routes are programmed, users can fill a timetable, in which the frequency of the routes is specified. Some routes which might need extra cleaning can be selected to be driven more frequent. Taking into account the feeding schedule of the cattle, on specific times the robot should not drive along the feeding fence. During this time the robot can drive another route that for example drives along the cubicles (resting boxes) of the cows.

1.6 Research question

This graduation project explores how technicians can create routes of the Lely Discovery in a different way and how the current technique can be improved. It is time consuming to create a route by teaching. To teach a route, users need to select all the route actions and let the robot directly drive the selected action. The Lely Discovery can drive the route independently by following the learned action sequence on a preferred time. The interface of the E-Link is not user-friendly, because the menu structure is complex and the user has to choose between different types of actions which are difficult to understand. So the action selection and programming is currently a time consuming and complex way for teaching the routes.

The same is true for changing time schemes and other preferences. From this the following research question can be defined, *how can the user friendliness and effectiveness of the route teaching for the Lely Discovery platform be improved?*

To answer the research question, research should be done on current route teaching techniques available and how routes can be defined with other techniques. Also field research is required to find the precise problems and difficulties in teaching the routes with the current technique.

This research will result in an overview of possible techniques and their pros and cons, a plan for a new method of route programming and how the current method could be made better. From the selected ideas a proof of concept will be made, which will be evaluated and tested on a test track (at Lely BV) and on different test farms.

2 Related work

Prior to the field research, research is done on current techniques for programming routes on a robot, to get insight in the field of route programming. This provides useful leads as to which solutions are possible to improve the current route programming technique. First some research is done on making a map, then how routes can be learned and how routes can be planned. Then some research is done on possible route preferences, and some more advanced user interfaces are shown for programming the routes or for sending instructions to the robot. Finally is shown how robots can ask questions about the routes if they get lost and how feedback can be given on programmed routes.

2.1 Map making

For path planning and navigation of a mobile robot a map of the environment is required. Most of the time a map is made by hand which contains inaccuracies in the map. An automatic construction of the map of the environment in which the robot is navigating is one of the fundamental problems in modern robotic systems. To accurately formulate the problem of simultaneous map building and localization of a mobile robot, dependencies between entities in the map must be considered (Castellanos, Tardos, & Schmidt, 1997, April).

In (Beeson, et al., 2007, March) is shown how spatial knowledge can be presented in different representations, like local metrical, local symbolic, global symbolic and global metrical representation (see Figure 8). The local topology of a place is a model of the topological relations between the path fragments at that place. For the Global metrical map-building a snapshot is stored of all the local places they detect. Then the best topological map which is consistent with the exploration is tried to find. This way a global metrical map is built by connecting and scaling all the snapshots. The symbolic representation is used for verbal instructions.

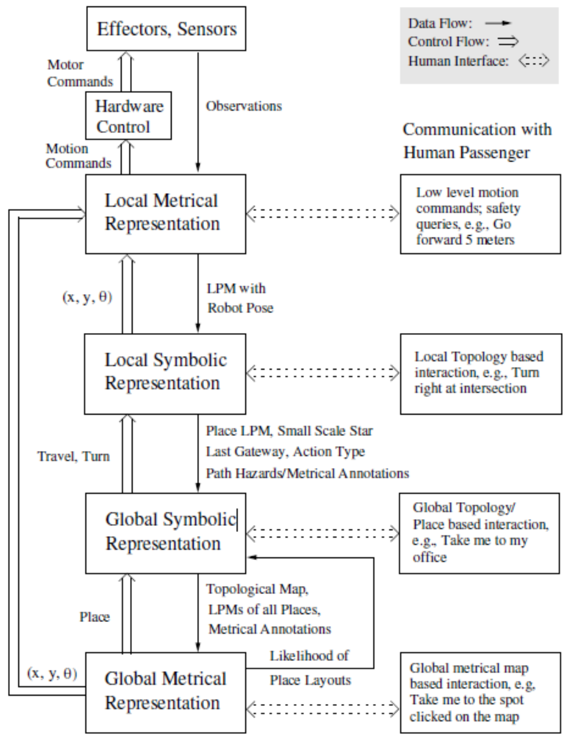

Effectors, Sensors
Data Flow:
Control Flow:
Human Interface:
Motor Commands
Hardware Control
Observations
Communication with Human Passenger
Motion Commands
Local Metrical Representation
Low level motion commands; safety queries, e.g., Go forward 5 meters
(x, y, θ)
LPM with Robot Pose
Local Symbolic Representation
Local Topology based interaction, e.g., Turn right at intersection
Travel, Turn
Place LPM, Small Scale Star
Last Gateway, Action Type
Path Hazards/Metrical Annotations
Global Symbolic Representation
Global Topology/ Place based interaction, e.g., Take me to my office
Place
Topological Map, LPMs of all Places, Metrical Annotations
Likelihood of Place Layouts
(x, y, θ)
Global Metrical Representation
Global metrical map based interaction, e.g. Take me to the spot clicked on the map

Figure 8 An integrated framework of multiple, disparate representations of spatial knowledge. Each level of abstraction uses its own ontology with concepts motivated by human communication about spatial navigation (Beeson, et al., 2007, March)

In (Hristu-Varsakelis & Andersson, 2002) is a directed graph used to build a global map where each landmark is stored in a node. The edge is what the robot has to do to travel from one node to the other. The proposed map representation is most useful when distinguishable terrain features are sparsely distributed or when most features are irrelevant to the robot's task.

In (Yamada, 2004) a mobile robot is build which use unsupervised-learning to recognize environments with low sensitive and local sensors. The robot does wall-following in enclosures. The action sequences are transformed into vectors and are used as input for a Kohonen's self-organizing maps (SOM). "SOM clusters large dimensional input vectors by mapping them to small discrete vectors. SOM is a two-layered network consisting of an input layer and a competitive layer (Figure 9). Any input node is linked to all competitive nodes, and all links have weights. As an input vector is given, input nodes have values corresponding to the input vector, and competitive nodes has values which stand for the distance between their weights and the input vector. A winner node having the minimum distance is determined, and weights of the winner node's neighbor nodes are updated" (Yamada, 2004). Learning is done without teaching. For the experiments a small robot is used in rooms with and without obstacles. Only simple rooms where recognized but a little more complex rooms had a very low recognition rate.

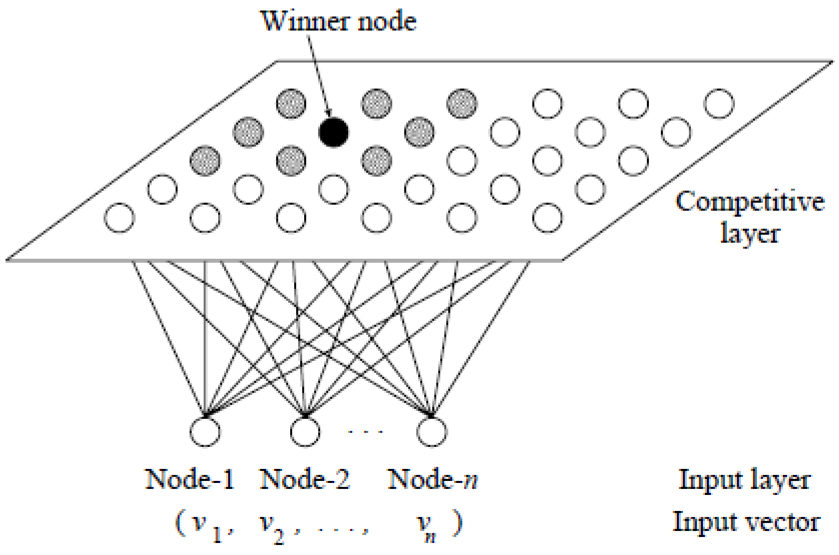


Figure 9 The structure of SOM

2.2 Route teaching

When a human navigates to a goal, he use as perception his vision rather than the estimated position. This because a human has a very powerful visual perception supported by much knowledge about the environment. A robot has little knowledge of the environment and a small capacity to manage this sparse knowledge. However a robot can measure the amount of movement precisely, for example the amount of wheel turns. This way the robot can navigate without much knowledge about the environment in comparison with a human (Maeyama, Ohya, & Yuta, 1996, November). To create the knowledge about the wheel turns, the route must be learned from a human.

In (Sprunk, Tipaldi, Cherubini, & Burgard, 2013, November) a robot is trained to drive a user-thought trajectory without using a global map of the environment. The robot uses two sensors, wheel encoders for the wheel turns and a laser scanner for a local 2D scan of the environment. The route is learned by driving the robot one time a specific route where the robot record the robot velocities from the wheel encoders. From the laser scans anchor points are constructed which are also recorded. From this information the robot can repeat the learned trajectory with millimeter accuracy.

In (Rawlinson & Jarvis, 2008) different attempts are described to direct an autonomous robot with topological instructions where the robot has no map of the environment. During the navigation a Generalized Voronoi Diagram (GVD) is build which can be used to transfer navigational concepts from human to robots. A Voronoi diagram is a way of dividing space into a number of regions (see Figure 10). For the navigation, two types of topological instructions are given to the robot. The first instruction specified the angle at which the robot should try to depart a point relative to the direction in which it approached. The second type of instruction gives more information about the edge which needed to be followed without information about the angle. The second type was the preferred one. Also different types of sensors are tested like sonar or a panoramic camera.

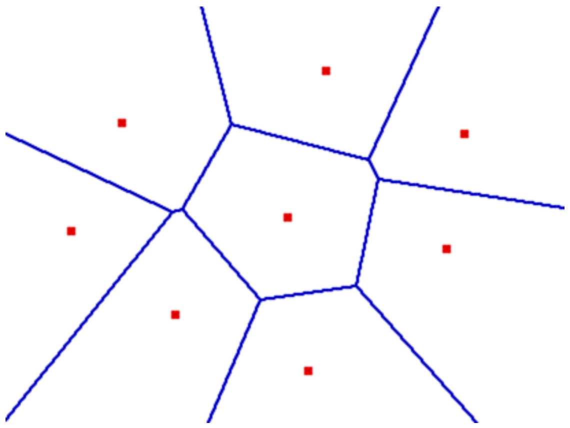


Figure 10 Voronoi diagram

In (Gat & Dorais, 1994, May) is investigated the application of conditional sequencing for robot navigation. Conditional sequencing is mostly applied to high level-task execution. The intuition behind this approach is that robots can navigate by following instructions similar to those that humans give to guide others to unknown destinations. The top-level sequence would are procedures for following halls, turning around corners, moving around obstacles, etc. The bottom procedures are simply sensorimotor control laws, laws that with a given sensor input controls the motor, are engineered by hand.

2.3 Route planning

Route planning for a navigation is a complex task. Route planning algorithms mostly use a minimum cost function, which calculates a route with minimum cost.

In (Silver, Bagnell, & Stentz, 2010) an offline and an online planning system is used. The offline planning system use satellite maps or other maps for global planning and the offline part use the local sensors for local route planning. The global map is divided into grid where every cell has a cost which is calculated by a cost function. An example of a cost function is shown. An demonstration is given to the robot by an expert to train the robot to interpret perceptual data to navigate autonomous. A robot was trained which was capable to traverse over rough, complex and unstructured terrain.

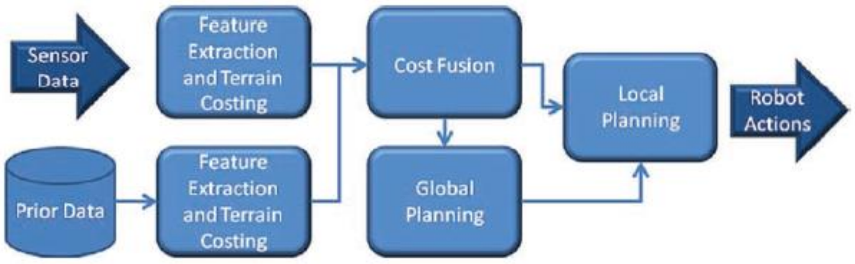

Sensor Data
Feature Extraction and Terrain Costing
Cost Fusion
Local Planning
Robot Actions
Prior Data
Feature Extraction and Terrain Costing
Global Planning

Figure 11 A high-level block diagram of the Crusher Autonomy system (Silver, Bagnell, & Stentz, 2010)

In (Sisbot, Marin-Urias, Alami, & Simeon, 2007) a motion planner is shown that takes human partners into account. The motion planner must provide safe, socially acceptable and legible paths. A safety criterion is used which focuses on ensuring the safety of the robot and the humans by controlling the distance between the two. This property aims to keep a safe distance between the robot and humans. The distance depends on the state of the human like if he is sitting or standing. Also a visibility criterion is used. This criterion aims to improve the human comfort during robot's motion. A visibility grid is build around the humans in the environment. The points before the human have a low cost but points at the back of the human have a high cost. Also hidden zones where the human can't see the robot have a high cost. The path planner use a cost function where the two criterions are merged with a weight.

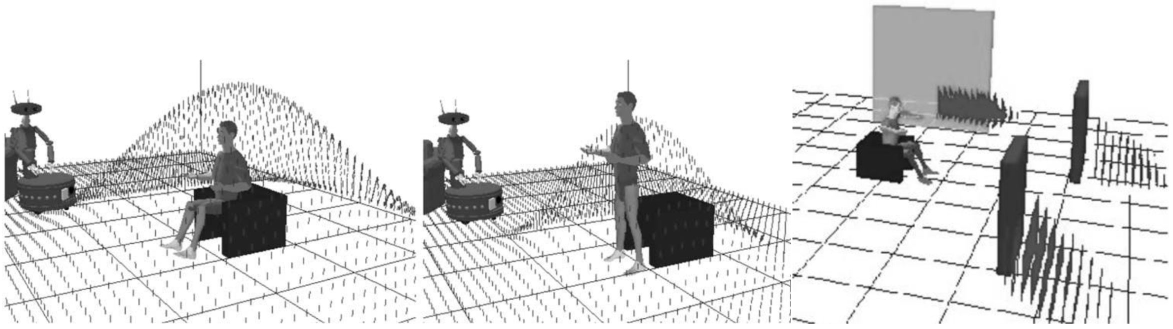


Figure 12 Safety grid built around every human in the environment. This depends highly on the human's posture. As the person feels less "threatened" when standing, the value and the range of the costs are less important. (c) Decreasing costs attributed to the zones hidden by obstacles. The supplementary costs discourage the robot getting too close to the obstacles and thus, prevents the robot from appearing suddenly behind hidden places (Sisbot, Marin-Urias, Alami, & Simeon, 2007)

2.4 Route preferences

In (Park, Bell, Kaparias, & Bogenberger, 2007) they incorporate user preferences into route guidance. They use a decision tree learning algorithm to choose the best route. From the user feedback they update the route choice model as in Figure 13. For generating the decision three they use a function which is based on travel time, travel time reliability, travel distance, the number of turns, directness, type of roads and familiarity.

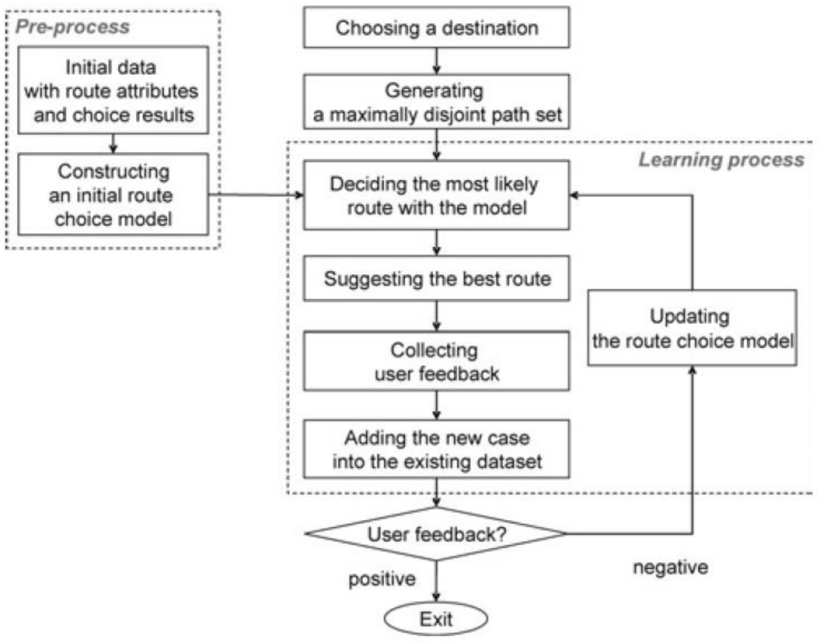

Pre-process
Initial data with route attributes and choice results
Constructing an initial route choice model
Choosing a destination
Generating a maximally disjoint path set
Learning process
Deciding the most likely route with the model
Suggesting the best route
Collecting user feedback
Adding the new case into the existing dataset
Updating the route choice model
User feedback?
positive
negative
Exit

Figure 13 Learning process (Park, Bell, Kaparias, & Bogenberger, 2007)

In (Wang, Huber, Papudesi, & Cook, 2003, October) they use a reinforcement learning algorithm and a semi-Markov decision process to learn the preference from the user from the user input. If a user gives input for example how to bring an object. The systems learns the preferred way how the object must be brought. Problems can be inconsistent commands.

2.5 User interface

To train a robot and to give preferences for a route, an interface is needed. An interface is used to communicate between two systems like a human and a machine. An interface can be a keyboard, smartphone which are used to give input to the system and to give feedback from the system.

2.5.1 Gestures

In (Guo & Sharlin, 2008, April) they explore the possibility of using gestures for HRI. They use the AIBO robot in their research which is able to response to high level commands such as walking or turning and low-level commands such as rotate a specific joint by a certain number of degrees. They designed two interaction techniques for manipulating an AIBO, gesture input through a Wii mote and nunchuk interfaces and another input technique uses a keypad. The Wii mote can be categorized as a generic 3D tangible user interface due to its ability to capture physical input and to interact with digital entities. They performed a user study with a navigation task and a posture task. The results point out that the Wii mote outperforming the keypad interfaces in terms of task completion time in both tasks.

2.5.2 Sketch

When a human tries to explain a route to a specific place, a sketch is a simple way to visualize it. The sketch contains landmarks like buildings and their relative positions. A route can be drawn between the landmarks.

In (Shah & Campbell, A qualitative path planner for robot navigation using human-provided maps, 2013) they do some research in path planning from a sketch. Waypoints are defined as points along the sketched path which intersect with the Voronoi-Delanuay graph (Figure 14). The reason that they use VD is because it mimics how humans navigate. This method selects locations along the path which can be described to two 'nearby' landmarks. The key difference of this approach with other work is that the waypoint extracted from the VD graph not need to correspond to a narrow opening or passage between two disjoint spaces. A sketched map is typically not an exact representation of the observed environment. In order to maintain appropriate spatial relationships within the environment, they use an optimization algorithm which connects each waypoint to each landmark by a virtual linear spring with a spring constant. This spring attempts to 'hold' the waypoint at the same relative position as indicated on the sketched map. Each spring affect the degree to which landmark influences the location of an estimated waypoint. It is calculated as a function of the Euclidean distance between the sketched waypoint and a sketched landmark and the uncertainty in the location of the landmark.

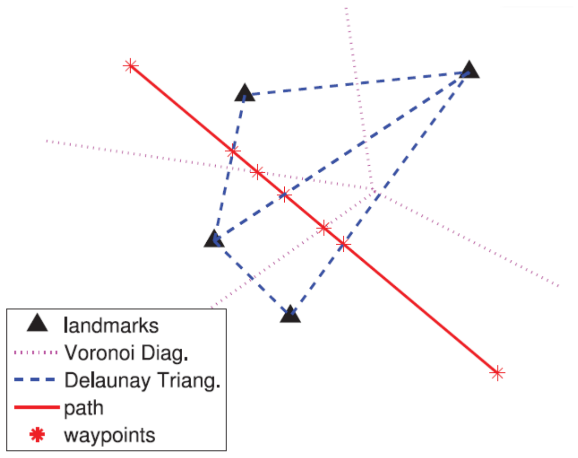


Figure 14 Waypoints (*) are extracted at points along the path that intersect with lines in the Voronoi–Delaunay graph (Shah & Campbell, A qualitative path planner for robot navigation using human-provided maps, 2013)

In (Shah, Schneider, & Campbell, A robust sketch interface for natural robot control, 2010, October) an architecture for controlling robots using a sketch interface is proposed. They use the variable duration hidden Markov model (VDHMM) for handwriting recognition (Figure 15). Every sketch is composed by a set of gestures where every gesture is made up by strokes like a circle ore a square. For classifying the gestures they use a classifier which needs to be trained with examples. A forward tree-search is used to find the most likely sketch by searching for the sequence of strokes. The sketched are made on a pre-drawn map.

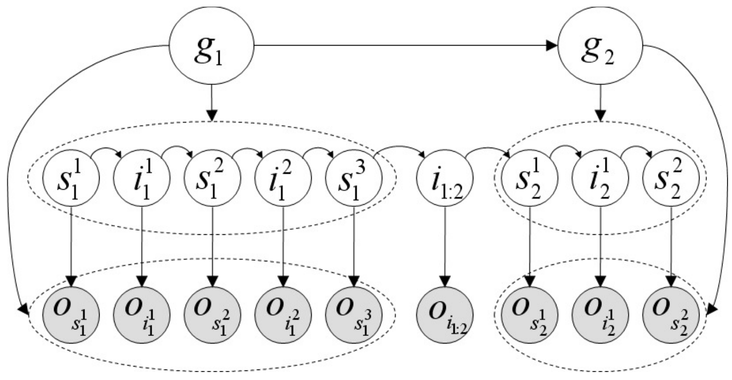


Figure 15 Example of a two-gesture stroke modeled using a variable duration hidden Markov model, where white nodes represent hidden (unobserved) variables, shaded nodes are observed variables, and arrows represent conditional dependencies. In this example, the first gesture g_1 has a duration of $d_1 = 3$ strokes, and the second gesture g_2 has a duration of d2 = 2 strokes. Interstroke nodes $i^1$1 , $i^2$1 , and $i^1$2 represent self-transitions on the gestures, while node $i_{1:2}$ represents a transition from gesture g1 to gesture g2. (Shah, Schneider, & Campbell, A robust sketch interface for natural robot control, 2010, October)

The sketch interface from (Skubic, Anderson, Blisard, Perzanowski, & Schultz, 2007) provides the possibility to manage multiple robots simultaneously. The user can sketch a map of the environment and navigation gestures to direct the robots. The user can add landmarks by sketching a closed polygon anywhere on the screen and provides an identifier for every landmark. The user needs also to sketch the current position of the robot with its direction. Feedback from the robot sensors can be used to detect the present environment, which allows a user to adjust the current placement of landmarks and robot by dragging them. The go-to commands are computed for each robot by looking at the relative position of the robot to the landmark closest to the goal point and the relative position of the goal point to the same landmark. These two quantities are extracted from the sketch as vectors and sent to the robot to be recomputed according to the relative positions of the robot and the landmark in the real environment. The location of the robot on the sketch is continue updated. This approach worked well for a small and simple environment.

In (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004) and (Chronis & Skubic, 2004, April) they use a sketched route map for direction mobile robots (see Figure 16). They model spatial relationships with force diagrams (see Figure 17). The relative position of a two-dimensional object A to another object B is represented by a function F^{AB}. The result is a force histogram which can be used to build spatial descriptions which provide a linguistic link to the user. This description can give an “opinion” about the position of A relative to B like right, left, above, below.

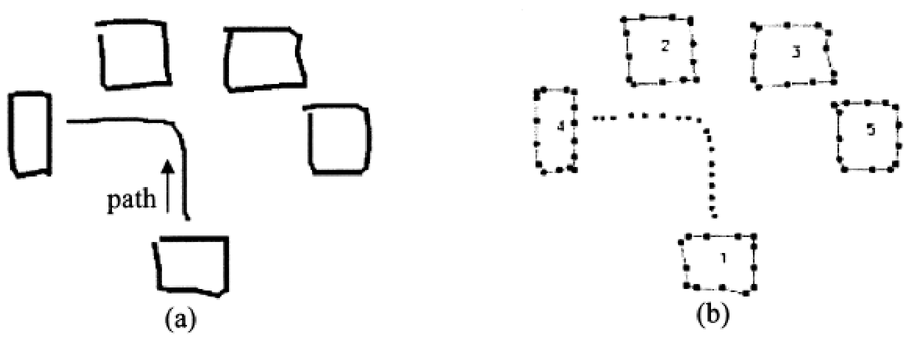


Figure 16 Sketch 1 (a) A route map sketched on a PDA. (b) The corresponding digital representation (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004)

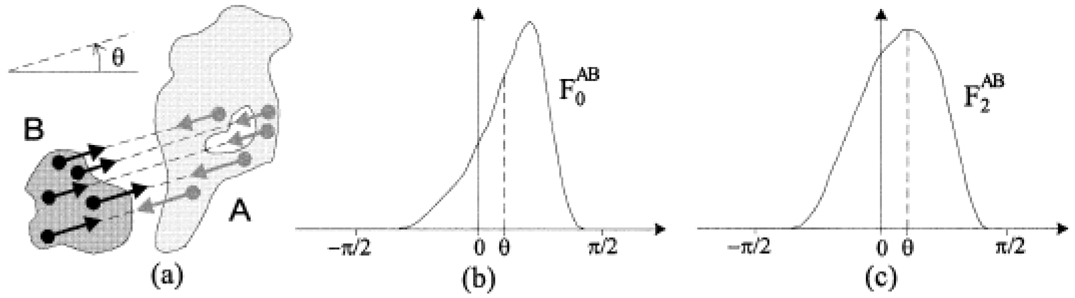


Figure 17 Force histograms. (a) the scalar resultant of forces (black arrows). Each one tends to move B in direction. (b) The histogram of constant forces associated with (A,B), i.e., the position of A relative to B. (c) The histogram of gravitational forces associated with (A,B) (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004)

From the sketch, points are extracted with a string of (x,y) coordinates. A delimiter separates the string of coordinates for each object in the environment. For each point along the sketched route, a view of the environment is built. For each object within the radius, a polygon is constructed using the boundary coordinates of the object as vertices. If any of the object points lies within the sensory radius, the entire object boundary is used as the object model. Then the histograms of forces are computed.

Besides the spatial information, movement along the sketched paths is also extracted. Here they want to track the orientation over time and compute significant changes. After this the landmarks states are determined like object 1 is behind the path, the movement is straight ahead. A system of fuzzy rules is used to extract the significant landmark states which are associated with the critical path node. The landmark states associated with the critical path nodes are used directly to generate a high-level linguistic description of the sketched route (see Figure 18). The description is generated as a sequence of path segments, each expressed in the form: When <landmark state> then <turn command>.

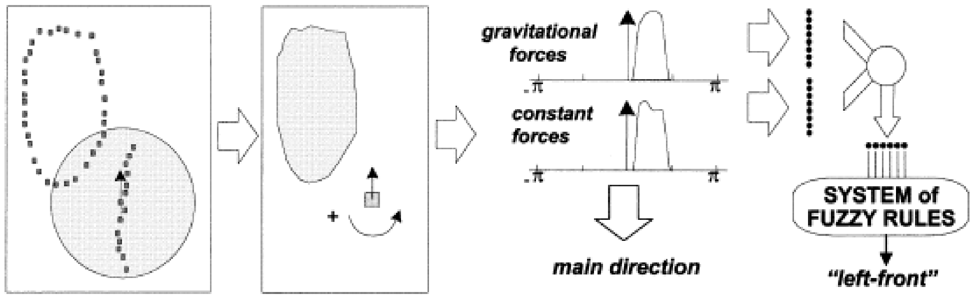


Figure 18 Synoptic diagram showing how spatial information is extracted from the sketch (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004)

2.6 Route feedback

Sometimes a robot get lost due to obstacles or other problems and because of the low sensory information and safety issues the robot stops. In cases like this a user has to decide what the robot needs to do next.

In (Fong, Thorpe, & Baur, 2003) they developed a teleoperation system model called collaborative control (Figure 19). The human functions as a limited resource for the robot, providing information and processing. The robot asks questions to the human in order to obtain assistance with cognition and perception. They designed a study to examine using collaborative control in the context of remote driving. The goals of the study were to observe how different users react to robot dialogue, to understand how work habits develop, and to evaluate system usability. The study revealed that dialogue, especially questions generated by the robot, is valuable for teleoperation. In particular, dialogue significantly helped beginners understand problems encountered by the robot. The study showed also that human assistance is a limited resource that must be carefully managed.

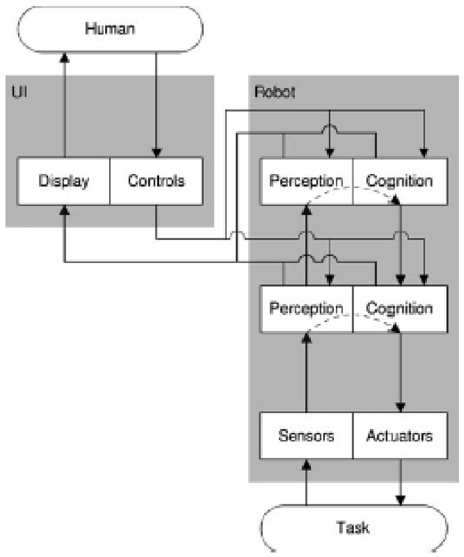


Figure 19 The collaborative control system model (Fong, Thorpe, & Baur, 2003)

2.7 Conclusion

There are a lot of techniques available for teaching a route to the robot or to learn the robot the route. The most techniques make use of a map which is currently not available on the robot. For the new robot a map will be available which provide an opportunity to use those new techniques which make use of a map. When a map is used, the routes can be learned by the robot or the routes can be made more reliable. After the field research should be decided if the techniques with a map can be used for the Discovery.

3 Field Research

During the research phase a prototype of the new version of the Lely Discovery was installed by a technician in a test barn. The old version was installed by another technician and a farmer. To find an answer on the research question, the technician and the farmer are observed while they programme the routes on user-friendliness and problems they encounter. In the related work different techniques for making and using a map are discussed. To decide if those techniques are useful, the layout of the barns are also observed.

Several questions could be asked about the difficulties and problems the technicians and the farmer encountered with programming the routes. Also could be observed how those barns are built to see how difficult it would be to make a map of those two barns. Installing the prototype on a test barn took the whole day an installing the Lely Discovery in a barn took several hours. In the following paragraph the points learned during the research are listed.

3.1 Point learned during the field research

User-friendliness

- Little changes in the actions are time consuming, before an action can be changed, the robot needs to be driven to the action to change it. The reason for this is that a change in one action can affect the rest of the route. When the action is changed, the other actions or new actions needs to be executed to drive the robot back to the charger. When this is done, the route needs to be tested if it works correct now. This is done by starting the route and return to the robot when the robot finished the route in the charger.
- If you want to copy actions to a new route from the charger to a specific point, you get a list of actions from a route with only the name of the action. It is difficult to see which route actions need to be copied, this because there is no visualization of the actions.
- The most difficult part for the farmer during the route programming with the service engineer was to press the right button. It was difficult for him to understand the difference between the buttons and their function. One reason for this problem is that the function for some buttons is inconsistent between different menus.

Problems

- It is time consuming to drive the robot back to the charger when the robot got stuck somewhere.
- The opinion of the service engineer was that it was better to make the robot a few hundred euro more expensive so that he could program al the routes. This because some farmers make for example routes where the robot drives to close to the beginning of a wall with an ‘L’ corner, this can cause the robot to drives against the wall (the bottom of the ‘L’) while the robot should follow the left side of the ‘L’ from the bottom to the top. Problems like this can cause the robot to get a lot of times stuck. When this is the case, the farmer need to bring the robot back to the charger for almost every day or several times a day. This will decrease the satisfaction and the motivation to go on with the robot.

- It was for the farmer difficult to know when he could start wall following with ultrasound. He also forgot every time to choose the right value for the distance between the wall and the robot.

Map

- More information from the sensors during the programming could be useful. This to get more insight in what the robot sees. The information from the sensors is already displayed during the execution of the actions when the action are programmed.
- A map can be useful to see which areas are cleaned by the routes.
- All the corners in both barns where corners of 90 or 270 degrees.
- The test barn has one path of almost 100 meter and it took more than 5 minutes to walk from the beginning to the end of the path with the robot.

Other points

- Time schedule could be more intelligent. If the robot is not enough charged while he need to drive a route, the route will be skipped.
- It took more than ten minutes to make a time schedule while the technician only wanted to ride one route every hour.
- Worldwide there are different Lely centers who sell the Lely products. Depending on the policy of the Lely center, the tester or the farmer programs all the routes. In the case of Lely Center Venray they install the charger and show how a route needs to be programmed, then the service engineer helped the farmer to program one route. The rest of the routes needed to be programmed by the farmer. The reason to choose for the option to let the farmer program the routes are the costs. When the farmer programs the routes, the robot can be sold at a considerably lower cost (several hundred euros).

3.2 Conclusion

From the field research can be concluded that the current route programming method works good for the technician. The user-friendliness of the current method is not very high, but this is not a problem for the technician because he needs to work with it every week. It took the users a lot of time to program one route and it was for a new user difficult to learn to program a route. Changing one action is time consuming, the user need to drive the robot to the specific action and then drive the robot back with the programmed actions. The layout of the two barns was very simple and could easily be drawn on paper.

4 Improving the current system or making a new system

The research question was “How can the user friendliness and effectiveness of the route teaching for the Lely Discovery platform be improved?”. In Appendix E a lot of possible solutions are discussed. In the field research is found that the main problem with the current route programming method is the time needed to program a route and making route modifications. The interface for the current route programming method is also not user-friendly, but technicians use the system every day so they learn to use it.

From the fact that the new version of the Lely Discovery will use a map and the layout of the most barns is simple, a decision is made to develop a totally new programming method for the Lely Discovery. A map offers possibilities to make a route without teaching the route. When users can generate a route with the map on the computer, users can possibly save a lot of time.

With the current route programming method an action sequence is generated. The action sequence is the input for the robot. The route programming method which will be developed should also generate an action sequence with the same action types as the current route programming method.

Users should still decide how the robot drives his rounds and they should be able to make different routes in the same barn. This to disturb the cows as less as possible (e.g. the farmer prefers the robot to drive in one alley always in one direction).

In the next chapter a prototype is shown of a route drawer, the requirements and the scope of the route drawer is shown. Then is explained how an action sequence can be generated from the input. A test plan is made for the prototype and the results are evaluated.

5 Prototype route drawer

To program the routes with a drawing tool, a prototype should be build where users can build a map, draw routes and generate route actions. In this section the interface design is showed and the functionality of the system.

5.1 Interface route drawer

The Interface is designed in such a way that users can easily step between the modules. In Figure 20 you can see on the top different tabs to step between the modules. This is done to separate all the information between the steps and to get a clean system output on the screen (see the center of Figure 20). On the bottom of the screen the buttons for the selected module are displayed.

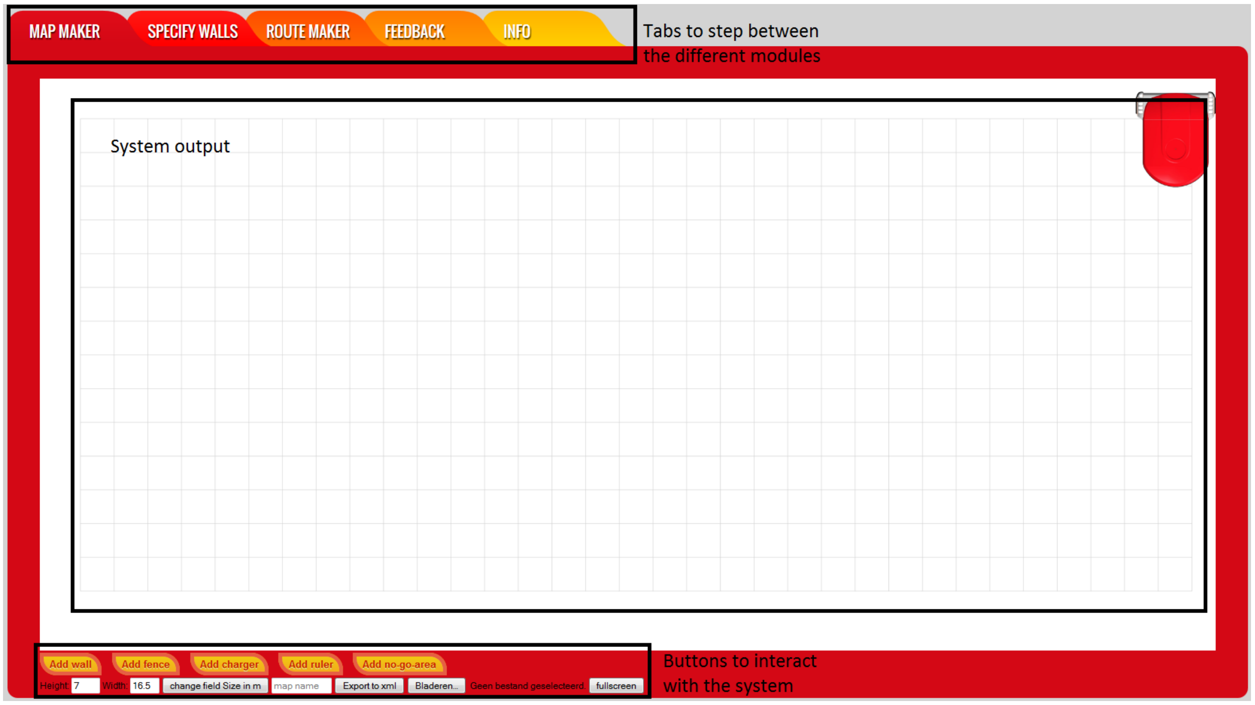


Figure 20 Interface design

5.2 Functionality route drawer

The system is for the users separated into four modules. In this section the functionality for every module is described.

5.2.1 Map Maker

The map maker is used to draw and modify the map of the barn. In Table 2 the functionality of the buttons can be found and how users can interact with help of the mouse.

Table 2

Button / mouse interaction	Functionality
Draw Wall	When users click on the button draw wall, they can click in the map to draw a new wall

Draw charger	When users have clicked on the button draw charger, they can click in the map to draw a new charger
Draw ruler	When users click on the button draw ruler, they can click in the map to draw a new ruler
Move points	Users can click on a point which is already drawn on the map, and drag it to another position. Users can also use the arrows on the keyboard to move a selected point
Move charger	Users can move a charger to a wall. The charger will connect to the closest point on the wall next to it.
Delete Points	Every point on the map can be deleted when selected by pressing the delete button on the keyboard
Export Map	The button export map will export the map to an xml-file where after the user can save it to the hard disk.
Import Map	Users can import a map from on xml file. When pressed on the import map button, they can select the file.
Change field-size	Users can change the height and width of the map in meters.

5.2.2 Specify Walls

With the specify walls module, users are able to define how the robot can use every wall. In Table 3 the functionality of this module is shown.

Table 3

Button / mouse interaction	**Functionality**
Bump on wall	When the bump on wall option is enabled, users can click on every wall to select or deselect if the wall is usable for bumping
Ultrasound on wall	When the ultrasound on wall option is enabled, users can click on every wall to select or deselect if the wall is usable for ultrasound
Wall follow	When the wall follow option is enabled, users can click on every wall to select or deselect if the wall is usable for wall following

5.2.3 Route Maker

With the route maker users can draw routes on the map. In Table 4 the functionality of this module is listed.

Table 4

Button / mouse interaction	**Functionality**
Add route	When users have clicked on the button add route, they can click in the map to draw a new route
Move point	Users can move a selected route point by dragging
Route on/off	Users can set a route on and off with a checkbox. When set off, the route is not displayed on the screen, and not included in the action export
Import map and routes	Users can import a map and routes from xml
Export map and routes	Users can export the map and the routes to xml

Export actions	Users can export the generated actions to an xml which can be used by the robot

The route maker supports users to draw route segments with a distance half of the width of the robot from the wall, so that the robot can drive with the center of the robot to a waypoint (e.g. Figure 21). When waypoints are drawn before a wall as the last waypoint in Figure 22, also with a distance of the width of the robot, the robot should drive to the wall and if possible bump. The distance between the center point and the front is larger than the half of the width of the robot. So when an obstacle is in front, the robot cannot drive with the center of the robot to the waypoint. If this is the case, the robot will drive to the wall and bump if possible. The waypoint is still placed with half of the width of the robot from the wall in the front to simplify the drawing.

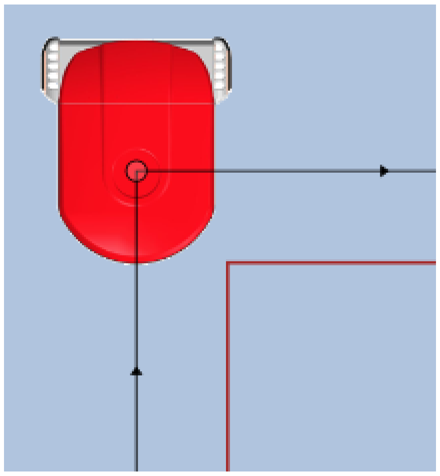


Figure 21 the robot should drive with the center point of the robot to every waypoint.

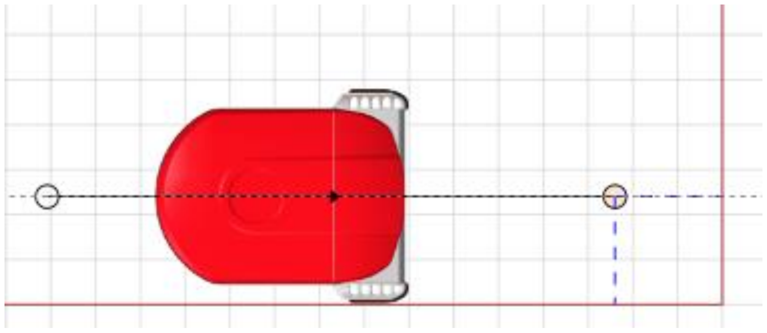


Figure 22 Path segment aligned with half the width of the robot from the wall

In order to overlap the routes for optimal cleaning, the route maker supports users to draw lines parallel to another path segment with the width of the robot minus some margin (see Figure 23).

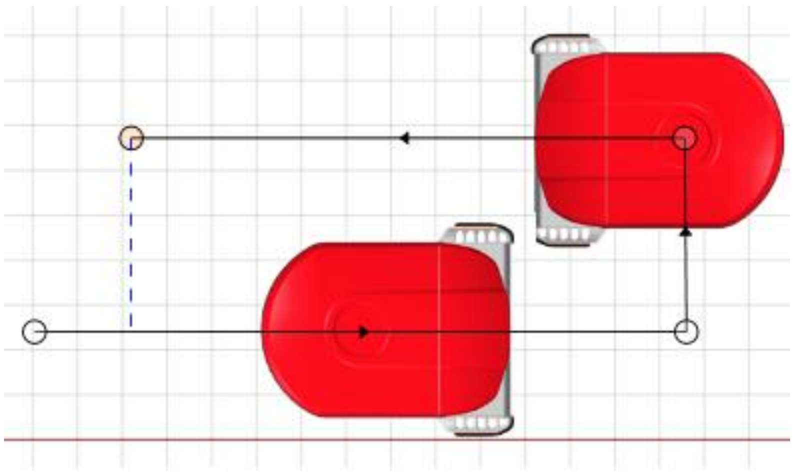


Figure 23 path segment aligned with the width of the robot minus a margin for overlap with another route path segment

The tool also helps users to align route segment by aligning the waypoints horizontal and vertical, which is shown by a black reference line (see Figure 24).

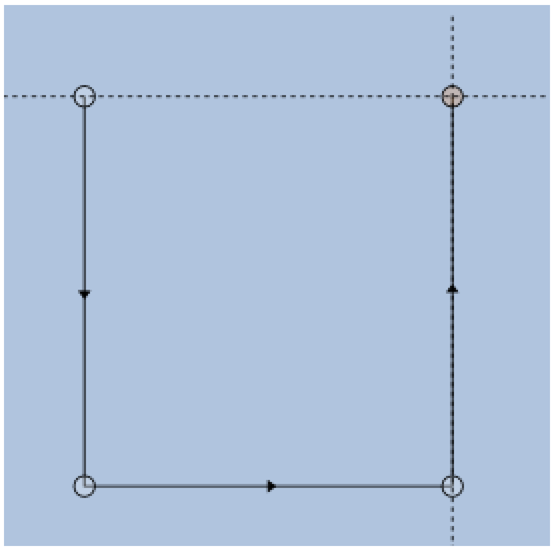


Figure 24 waypoints are horizontal and vertical aligned

The route should start and end in a charger, in Figure 25 a reference line is shown that the waypoint starts or ends in the charger. See Figure 26 for an example of a route which is drawn in the route maker.

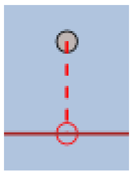


Figure 25 waypoint drawn in the charger

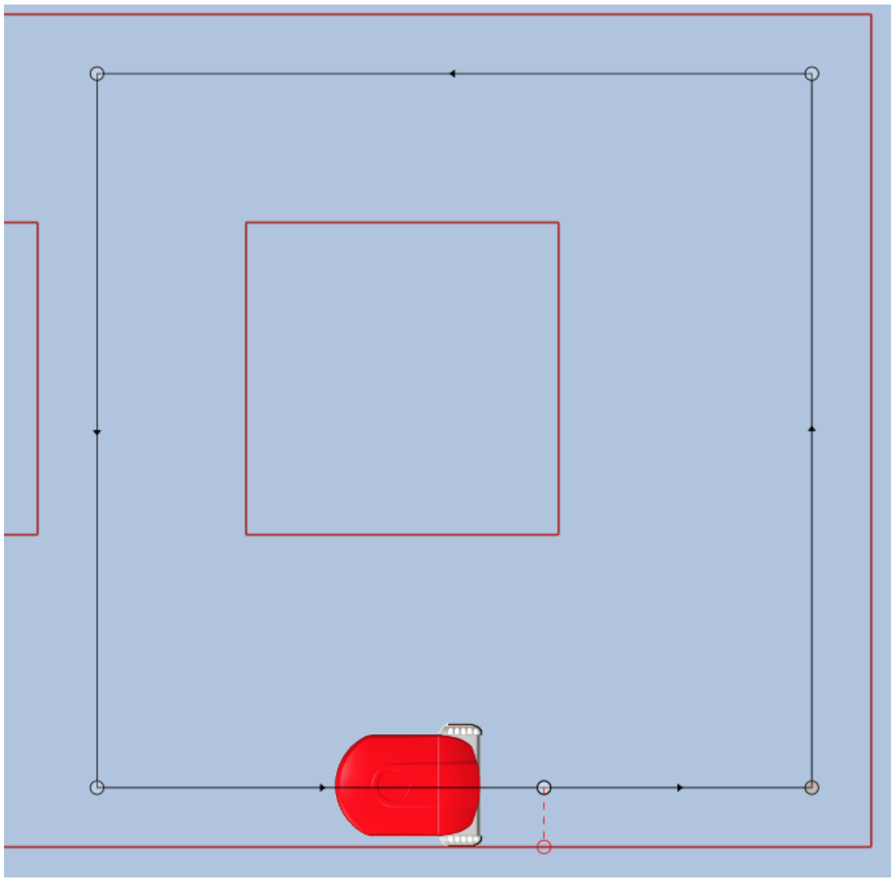


Figure 26 example route where the route start and ends in the charger

5.2.4 Feedback system

In Table 5 the functionality for the feedback system is described. Users can interact with the generated actions and change the actions. They can also see if the actions are calculated correclty by coloring the route line. The route line is green if it is calculated succefully, orange if it could be optimized by changing the route and red if it could not be calcualted (see Figure 27). Users can also play a visualization where the robot excecutes the actions. This can help users to decide if the route could work in practice.

Table 5

Button / mouse interaction	**Functionality**
Area cleaned	When hovered over a box area cleaned, the area that gets cleaned by the routes is displayed by a green color in the map
Action details	Users can select a route line to display the action details of that route segment
Change actions	Users can click on the action details to change the values

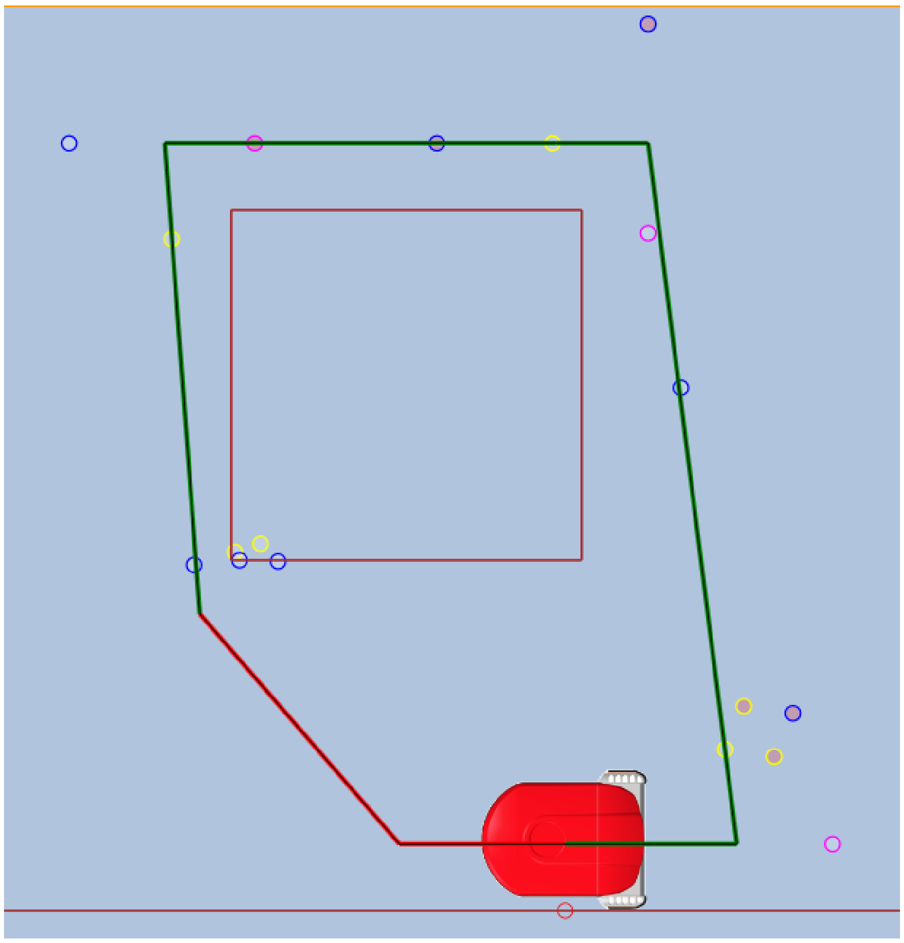


Figure 27 feedback after calculating the route. The last parts or colored red, because the route line is drawn to close to the wall and the corner

5.3 System data flow

In this section the data flow of the system is described on the basis of a data flow diagram which can be found in Figure 28.

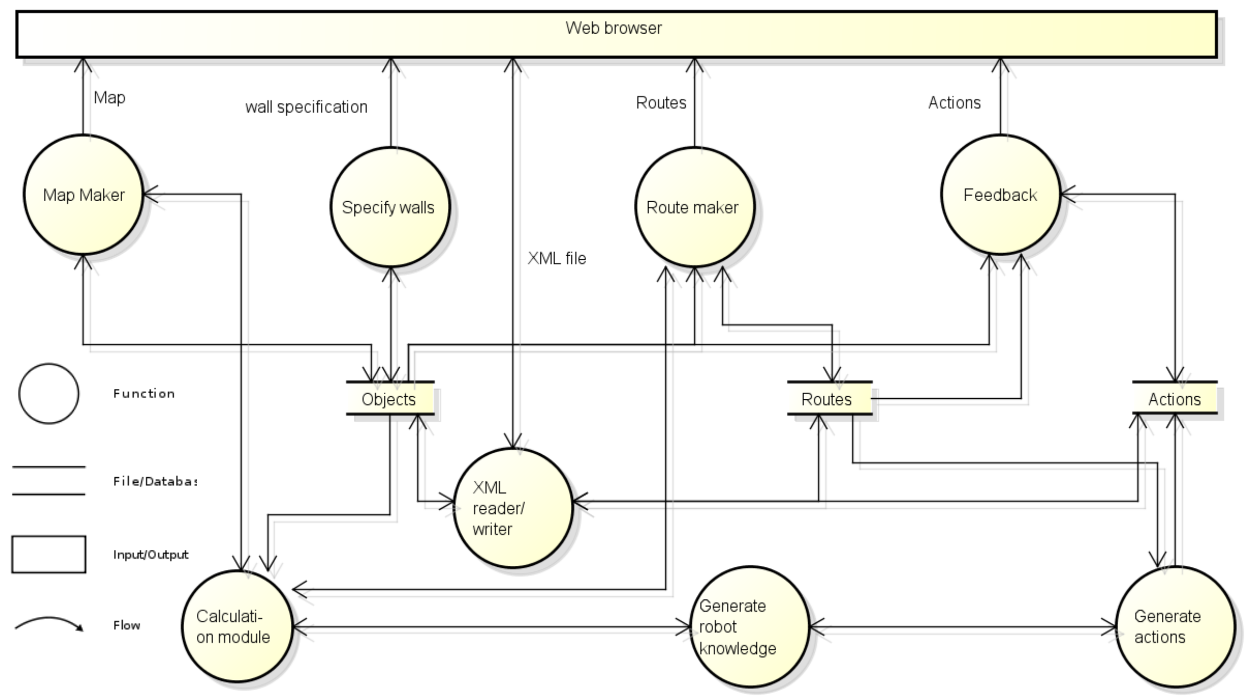


Figure 28 Data flow diagram

Web browser
The interface can be opened with the web browser. Users can interact with the system with help of the web browser.

Map maker
With the map maker, users can make a map by generating and modifying objects.

Specify walls
Users can specify for every wall if it is usable for bumping, ultrasound, and wall following by the robot.

Route maker
With the route maker, users can make different routes. The route maker helps users to place route points on logical places.

Feedback
The feedback module will give users feedback on all the created routes and the generated actions. Here users can also modify the generated actions.

Calculation module
The calculation module exist of a lot of basic and complex mathematically formulas, for example to calculate distances to objects from different position from the robot.

Generate robot knowledge

This module generates the knowledge for a specific location and robot orientation. Here all the knowledge about for example the closest obstacles in specific directions is calculated with help of the calculation module.

Generate actions
This module generate the actions for every route. This is done by first generating the robot knowledge on the start location with the orientation of the robot. Then a decision is made for the best action, with help of the cases and the robot knowledge.

Objects
The object database contains all the object generated with the map maker, like the walls, rulers, chargers.

Routes
The route database contains all the route points.

Actions
In the action database, all the generated and modified actions are stored.

XML reader/ writer
With the XML reader/ writer all the objects, routes and actions can be saved in an XML file or retrieved from a XML file.

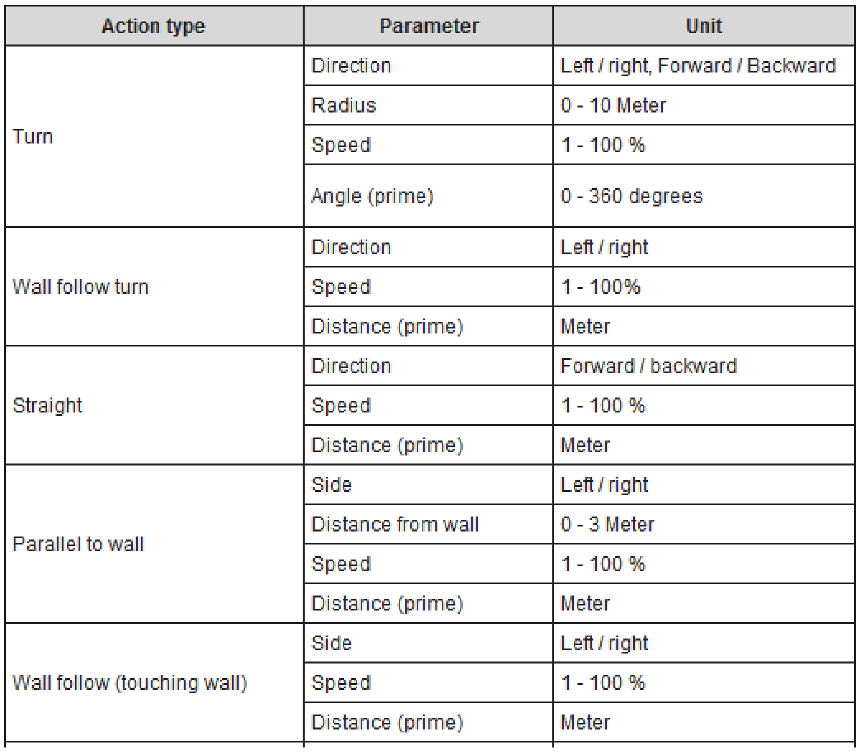
6 Requirements route drawer

In this section first some requirements of the Lely Discovery are specified. Then the requirements of the map drawer, route drawer and the action sequence as output are specified.

6.1 Assumptions and requirements for the Lely Discovery

- The set of actions are presented in Figure 29 from which users can choose
- Each action has a set of parameters that users can configure before starting
- Users can transfer a map in xml format to the machine and vice versa
- Users should start and end every route at the charger when programming
- The machine shall drive a pre-programmed route accurately enough to come back to the charger
- Inaccuracies while driving, caused by slipping wheels for example, shall be compensated by using the given map where possible. For example when the machine is driving parallel to cubicles and it passes the end of the cubicles (which is also drawn in the map) it shall correct the longitudinal position to the end of the cubicles
- Wall following is more reliable than a straightforward action and a parallel action
- A parallel action is more reliable than a straightforward action
- Wall turn is more reliable than a normal turn
- More bumping points in the route makes the route more reliable

Other relevant requirements for the Lely Discovery can be found in Appendix A



Action type	Parameter	Unit
Turn	Direction	Left / right, Forward / Backward
	Radius	0 - 10 Meter
	Speed	1 - 100 %
	Angle (prime)	0 - 360 degrees
Wall follow turn	Direction	Left / right
	Speed	1 - 100%
	Distance (prime)	Meter
Straight	Direction	Forward / backward
	Speed	1 - 100 %
	Distance (prime)	Meter
Parallel to wall	Side	Left / right
	Distance from wall	0 - 3 Meter
	Speed	1 - 100 %
	Distance (prime)	Meter
Wall follow (touching wall)	Side	Left / right
	Speed	1 - 100 %
	Distance (prime)	Meter

Figure 29 action types for the Lely Discovery

6.2 Map drawer

- There is functional support available so that users can draw a map with 1 cm accuracy
- There is functional support available to let users draw one or more walls
- There is functional support available to let users draw one or more chargers
- There is functional support available to let users draw one or more rulers, this to easily place different objects on the correct distance of each other
- There is functional support available to let users draw the walls straight
- A grid should be available to help users to draw the walls straight
- The system should connect points of the same object which are drawn by users on the same location. This to create from different walls that connect, one big object.
- A connected object, for example all the walls that overlap with the endpoints should easily be moved to another location in the map
- A connected object can be copied and pasted
- The system should support users to place the charger easily on a wall
- The system should support users to easily define on which side the charger is placed on the wall
- The length of a wall should be displayed
- The system should support users to delete all types of objects
- The system should have functionality to export a map to an xml file which can be imported by the robot
- The system should have functionality to import a map from an exported xml file
- Rulers are not included in the export
- The system should support users to zoom into the map and move through the map
- The grid size should change when zoomed in
- The system should have functionality to let users type the distance between the selected point and the new point
- When a distance is typed, a new point should be placed into one of the four direction from the selected point with the arrow keys

6.3 Route drawer

- The system should support users to draw one or more routes
- The route points should not merged to one point as in the map maker when drawn on the same place
- Every line/ route segment should show the direction of the route
- Route lines are defined by the middle position of the robot
- Route lines should be easily placed along the walls with a distance between the line and the wall of half the robot width
- Routes should start and end in the charger
- Route lines should be easily be placed along other route lines with the a distance between the two line of 0.9 * width of the robot
- Route lines should not intersect with walls
- Every route should get a checkbox where the user can set a route on and off
- If the route is set off, then the route is not displayed in the map and not exported to the robot

6.4 Route feedback

- When the route starts and end in a charger, users should get feedback on the reliability of every route segment
- Every route segment should get a color: green (reliable), orange (will work but it cannot be guaranteed that it will be reliable) and red (will not work).
- When hovered over a route segment, the calculated actions are displayed in text and an overlay over the map is shown with the route actions. The actions are in colors related to an action and the position of the robot which is not directly the same as the route segment.
- The calculated actions can be changed, where after the new actions are calculated from that point
- When hovered over a box called area cleaned, the area that is cleaned by the calculated route actions is colored green
- Time needed to drive the route is calculated and displayed
- Energy cost in percentage of the battery is calculated and displayed

6.5 Action output

- For every route an actions sequence should be created
- The action sequence exist of the same actions available to the robot
- The actions should include bump points on every place where possible
- The actions should follow the route segments in the same direction
- The most reliable actions possible on the route should be chosen

7 Scope route drawer

Before the algorithm can be designed and implemented, the scope for the Lely Discovery needs to be clear. In this chapter the layout of the most barns and the design of the discovery is shown. At the end the input for the algorithm is defined.

7.1 Barn layout

According to the farm management support department of Lely (FMS), the ideal barn should have the minimal distances displayed in Table 6 between a specific object and the first obstacle. This to give the cows enough space to pass other cows and to escape (see Figure 30).

Table 6 distances between object and the first obstacle

row of cubicle (resting box)	3 meter
feeding fence	4 meter
passageway + drinker	4 meter
milk robot	5 meter

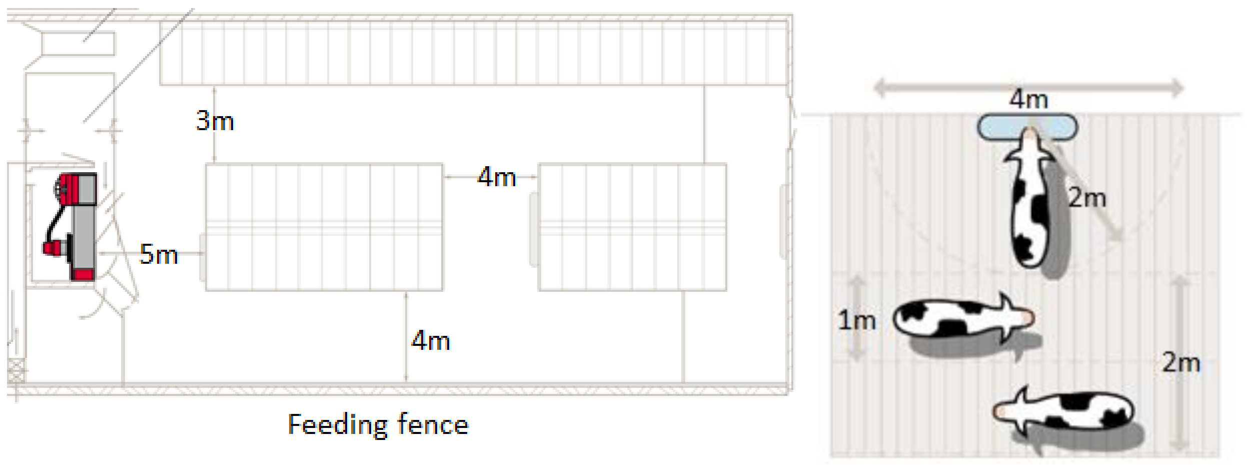


Figure 30 minimal distances between objects for the ideal barn

In Figure 31 you can find an example of a 2 + 2 barn and in Figure 32 a 4 + 4 barn. In a 2 + 2 barn there are two rows of cubicles in one part of the barn, and on the other site of the passageway there are again two rows of cubicles.

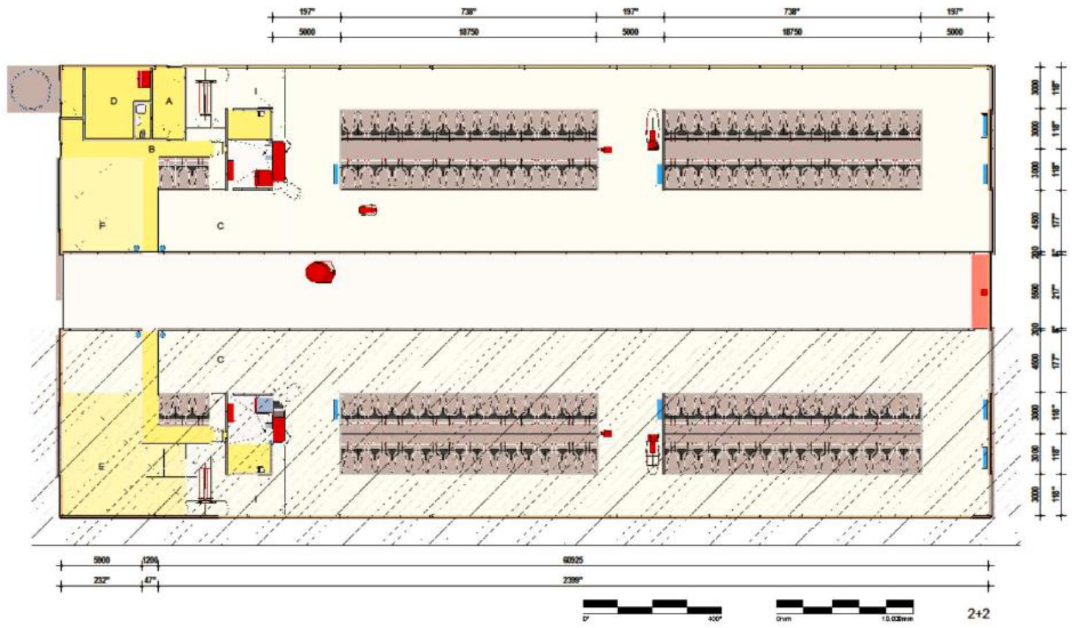


Figure 31 2 + 2 with 2 rows of cubicles on one side of the feeding path and 2 rows of cubicles on the other side

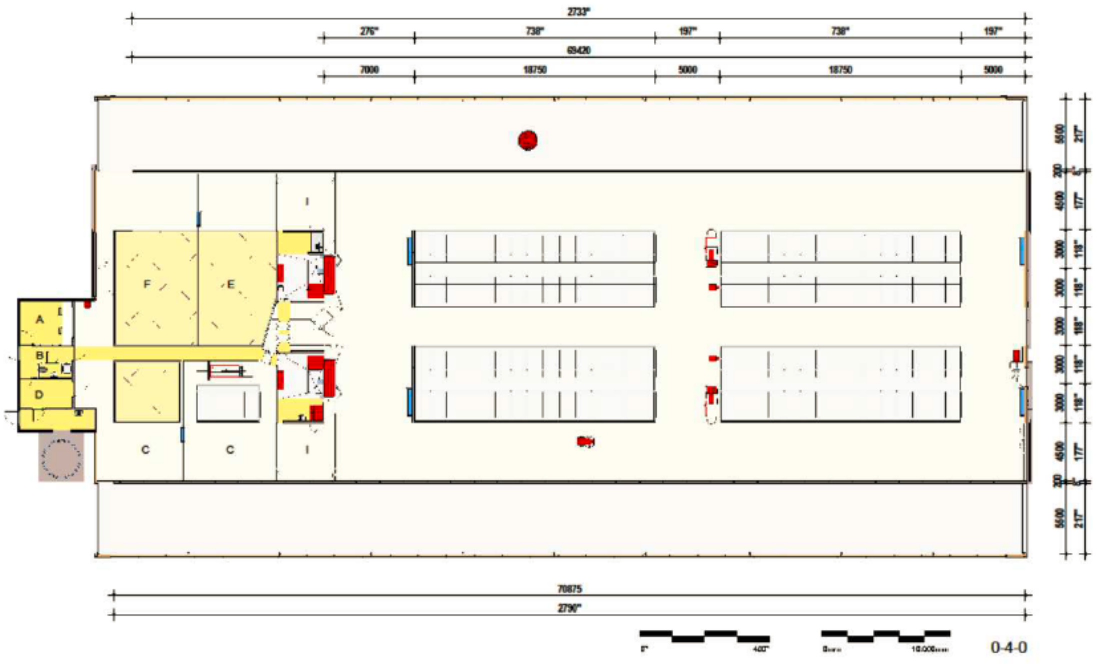


Figure 32 4 + 4 barn with 4 rows of cubicles on one side of the feeding path and 4 rows of cubicles on the other side (one side displayed)

According to FMS the majority of all the farms in the world are built with right angles between the walls which is approximately more than 90%. They expect that less than 10% of all the farms in the world satisfy the ideal farm described in Table 6. The sizes of the old barns are difficult to estimate, but there are barns which are built in such a way that some paths are width enough for one cow, but the variation is huge. So the minimal distance between the walls depends on the Lely Discovery. The Lely Discovery is 1.2 meter width, so when a margin is included, the minimal width of a path needs to be 1.3 meter to let the Lely Discovery drive on an accurate way every possible route. This minimal distance will be a requirement for the program where users can still draw a path and where the algorithm should find an action sequence. Almost every barn has some places where the robot can't follow the wall or can't bump on a wall. The reason for this is for example that there is placed a drinking trough against the wall, the wall is too low for wall following or the wall is not strong enough.

Overview barn layout:

- Most corners have right angles between the walls but there are always corners with other angles
- There are other obstacles then walls like drinking troughs
- Some walls are usable for bumping and some walls not
- Some walls are usable for wall following and some walls not
- The minimal distance between the walls where the Lely Discovery can drive between is 1.3 meter

7.2 Lely Discovery design

The Lely Discovery is designed in such a way that when he is aligned with a wall, still can turn from the wall without bumping on the wall when there is enough space on the other side. See Figure 33 how the Lely Discovery turns to the left from the wall without getting stuck on the wall.

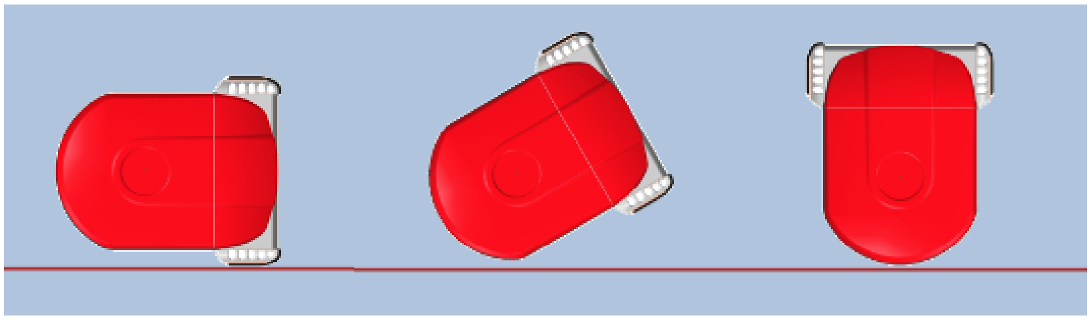


Figure 33 Lely Discovery turns from the wall without getting stuck on the wall

7.2.1 Control points

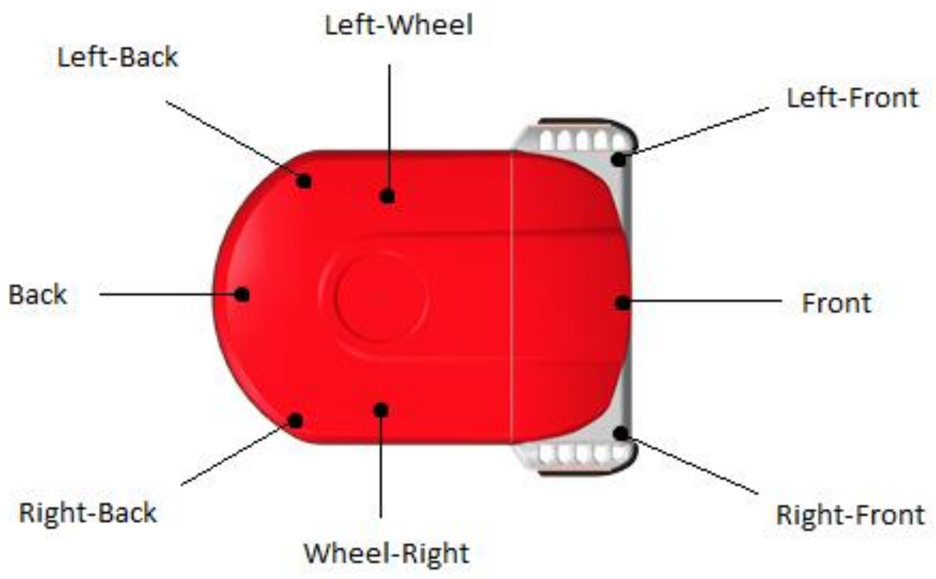


Figure 34 control points used for executing the actions

When a robot drives around in the world the robot can be represented by one single point. In this case it is quite simple on paper to move the robot from location x to a goal point y. In the real world, this is not so simple because the robot has a finite area in the plane and volume in three dimensions. For example, when the robot is represented by a single point on the front-center and there is an obstacle in the front left of the robot. With this representation, the obstacle is not seen by the robot, because in the front-center of the robot is not an obstacle. When driving forward, the robot will get stuck with the left-front on the obstacle. Evaluating the effect of a movement on every place on the robot is mathematically and computational quite complex. An alternative approach is to select a subset of points on the robot, called control points (H. Choset, 2005).

For executing the actions eight control points are defined for the Lely Discovery (see Figure 34). For almost every action one control point is used. The left- and right-front control points are used for wall-following and the front point is used when driving straight. When driving backwards the front points cannot be used, because when the robot slips for example with the left-wheel, the back of the robot will move to the right-side. This will cause the robot to turn around with 180 degrees and then moves forward instead of backwards. For this reason one of the three control points on the back should be used in this case. When making a turn to the left, the robot will use only the right wheel which will move the robot around the left-wheel. For this reason both center-points of the wheels are also used as control points. In Table 7 the control points which are used for the position during the execution of an action are displayed. The other control points are also used in the code of the robot but are not directly used for the end position.

For the algorithm of the route drawer, another subset of control points could be used to find obstacles. Minimal the left-front, right-front, left-back and right-back point should be used for finding the obstacles. To move to a waypoint, also the center point (point between the wheels) of the robot should be used.

The front-center and the back-center point are usable for calculating the position of the robot when an obstacle is in the front or at the back. So for the algorithm to generate the action with the route drawer will minimal use 7 control points in total.

Table 7 control points used for the position of the robot during the execution of an action

Action	Control point
Forward	Front
Turn left	Right-front
Turn right	Left-front
Backward	Back
Wall follow left	Left-front
Wall follow right	Right-front
Wall turn left	Right-front
Wall turn right	Left-front
Parallel to wall left	Left-front
Parallel to wall right	Right-front

7.3 Input for algorithm

The input for the algorithm is the map and the waypoints of every route. In the subsection barn map and route waypoints is shown how the input should look like and what the requirements are for the input.

7.3.1 Barn map

With the map drawer a set of walls and the position of the charger(s) is generated. Here is assumed that the map is drawn correctly which can be done with 1 cm accuracy. Every wall is specified correctly, if it is usable for wall-following, bumping or for ultrasound. The charger is placed on a wall with a minimal of 3 meter from one of the end points where the robot drives into the charger and minimal 1 meter from the second end point to let the robot drive from the charger. Also no obstacles are placed around the charger. This because the robot should be able to do wall-following until it finds the charger. If there is a small obstacle somewhere in the field, users have placed a no-go-area so that the robot can avoid the obstacle like a drinking trough.

Overview map input requirements:

- Drawn with 1 cm accuracy
- Charger is placed on a wall with a minimal distance of 3 meter from one end point of the wall and minimal 1 meter from the second end point
- Obstacles are enclosed within a no-go-area
- Walls are specified if they are useable for bumping, wall following and ultrasound

7.3.2 Route waypoints

The route maker provides a set of waypoints which are drawn with a minimal distance of half the width of the robot from every wall and the routes that start and end in a charger (see section Route Maker). Besides the waypoints, also every point on the route line should have a minimal distance of half the width of the robot from every wall. On this way the drawn routes automatically avoid obstacles, so that the robot will not get stuck on an obstacle when moves forward along a route segment. If there are points on the line that have a smaller distance than half the width of the robot from a wall, the route maker should not draw the line and should provide feedback. Waypoints drawn with a larger distance from the wall are also possible, but can result in a less accurate route. Waypoints which does not satisfy the described conditions can result the robot not being able to move from one waypoint to another. The route maker should show when a waypoint is placed incorrectly and violating the requirements. Although with those requirements users have to spend more time defining the route, it will result in a more simple and stable algorithm to generate a robust route.

Overview route input requirements:

- The route starts and ends in the charger
- Waypoints are draw with a minimal distance of half the width of the robot from every wall
- Every point on the line has a minimal distance of half the width of the robot from every wall

8 Algorithm design for generating an route action sequence

In the previous section is the input for the algorithm defined. In this section is shown what the output should be. Then an algorithm is presented which is able to find a correct set of actions. A decision is made that a case based system would be the best option, this because of the complex actions of the Lely Discovery. Then the pseudocode and the possible cases are shown.

8.1 Output

As input a set of waypoints is given, which represents the route. Also the map, start position and orientation of the robot is given.

The algorithm should generate an action sequence which ensures that the robot drive the drawn route in an accurate en reliable way. The algorithm should find the right places to bump on the wall and find the most reliable actions possible in every situation. For every action, the right parameters should be selected and defined.

When a waypoint is placed on a place which cannot be reached by the robot on a reliable or accurate way, the corresponding route line should be marked with an error specification. This error specification will be translated to a color in the simulation. With the color for every part of the route, uses can see if the robot can reach the waypoint with a reliable and accurate set of actions. If a part of the route line is not colored green, users should change the route until every part of the route is colored green in the simulation.

With the algorithm the position and orientation of the robot can be calculated after every action in perfect conditions. This can be used in the simulation to simulate the movements of the robot, which can help users to decide if the generated action sequence is what they want and if it the route cleans what they wanted to clean. If this is not the case, they can change the route on that location and so the input for the algorithm. When the route is to their wishes they can export the action sequence to an xml-file which will be the input for the robot.

8.2 Search algorithms

From the map and from the routes an action sequence should be generated, which moves the robot from the first waypoint to the next waypoints and finally to the charger. For this a search algorithm should be found which is able to find an action sequence that ensures the robot to drive the route robust and accurate.

To move from every waypoint to another waypoint there can be made a choice between a discrete set of possible type actions. In Figure 29 five action types are described which the robot use to drive around. Every action can be defined by the direction, speed and distance or by the direction, speed, angle and radius. The parameters distance, angle and radius are continuous so the set of actions is infinite. When there is an infinite set of actions, there cannot be searched for every possible combination of actions to move from one waypoint to the other waypoint. So an algorithm that search for example for a least-cost path by checking every possible action sequence is not possible.

The continuous parameters could be transformed to a discrete set of values. The angle can for example defined with a factor of 20 degrees which gives us 18 different actions. When every parameter is transformed to a discrete set of values, there is a finite set of actions.

Before the transformation there was an infinite set of actions but only 5 type actions. After the transformation there will be a large set of actions. A large set of actions will make it hard to calculate every possible path from the first waypoint to the last waypoint.

8.3 Case based algorithms

The algorithm doesn't have live input from the robot so the actions need to be simulated. In a simulator the movements of the robot can be simulated in ideal conditions. When the map and the position of the robot is known, the position of the robot can be calculated after executing an action.

With the map and the exact position of the robot, a knowledge base can be built about the position of the robot in the environment. For a set of points on the robot can be calculated what the distance is between that point and the next obstacle and what the angle of incidences is for that point on the obstacle. Also the distance and the angle to the next waypoint can be calculated. This information can be used to calculate which actions are possible and what the position of the robot will be after a specific action in the simulation. With the exact location of the robot, the position history of the robot is not necessary to find the most reliable action.

From the users a route is required as input which is simple to follow and already avoids obstacles. To get from one waypoint to another waypoint in ideal conditions, the robot needs to do one turn action and then one or more move actions to drive to the next waypoint. To find a correct action sequence in ideal conditions, only the relevant turn or movement action needs to be found on every position. So it is not needed to search for all the possible action sequences.

To find the relevant turn and movement actions, a set of rules can be used. In practice with the old route programming method, users find a path solution by explaining the problem with expressions as 'in such and such situation, I do this action first and then that action'. These expressions can be represented in if-then production rules (Negnevitsky, 2005). A system that uses production rules is also called a rule-based system.

A rule-based system is based on the idea that humans solve problems by applying their knowledge to a given problem representation with problem-specific information (Negnevitsky, 2005). In this situation every problem could be matched with a known case which describes a general situation, so a sort of case based system can be build which is a subset of a rule-based system. A case-based reasoning (CBR) system solves new problems by retrieving relevant cases and adapting them to fit new situations (Leake, 1994). For this a set of cases needs to be found which overlap with every possible situation. Then the algorithm can iterate over the cases, until it finds the case which is relevant to a specific situation, and return the corresponding action.

In Figure 35 the basic structure is shown of a rule-based system. In the long-term memory all the rules or cases are stored and in the short-term memory the problem-specific information and the facts are stored.

With the information in the short-term memory, the system can find the relevant case from the long-term memory with the corresponding action to execute. When is known which action can be executed, the continuous parameters of the action can be calculated like the distance or the angle. This can be calculate with the rules specified for that case in the long-term memory and with the variables in the short-term memory like the distance to a wall in front.

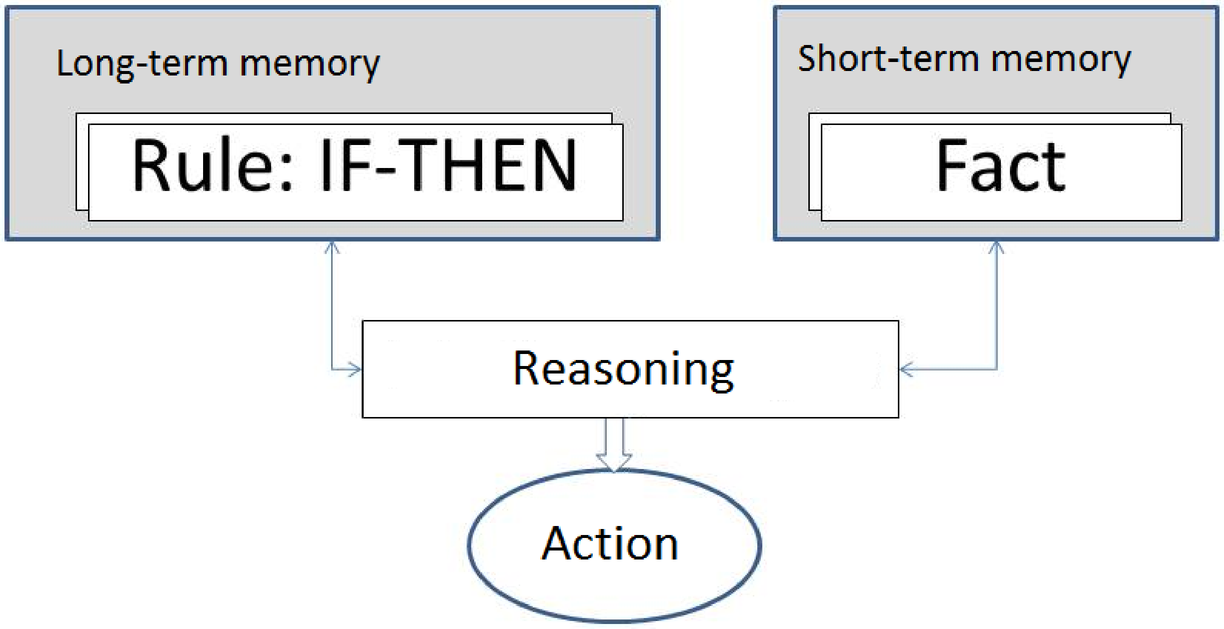


Figure 35 basic structure of a rule-based reasoning system (Negnevitsky, 2005)

For a complex case specific rules can be defined with certain thresholds. When for example the angle between the front of the robot and the wall in front is between 80 and 100 degrees and if the wall is usable for bumping, the robot can move forward until the robot bumps onto the wall. With this simple case a lot of different situations are covered, where a wall is in front with an angle between 80 and 100 degrees.

The advantage of a case based system is that with a finite set of cases all known complex situations are covered, which are solvable with a standard solution. Every case can cover a large set of situations where the situations are bounded by the thresholds of the rules.

The algorithm has as input a route which is simple to follow and already avoids obstacles. In ideal conditions the robot only need to turn to the next waypoint and then move in the direction of the next waypoint. So in the simulation the problem can be solved by selecting the most reliable action that is possible. The possible actions of the robot are a combination of the type actions and the discrete parameters, direction and stop-condition. For example there are 4 possible actions for the type action wall-follow, wall-follow left or right and as stop condition distance or wall-bump. From this it follows that there can be maximal 18 possible actions and from the fact that there are no obstacles in ideal condition, there are 18 cases when the robot only drives in ideal conditions.

In the barn the conditions are not ideal. The actions needs to be corrected or actions needs to be added to avoid obstacles next to the route, see for more information the section Action sequence evaluation. For every combination of actions the algorithm need to look if it can go wrong when there are obstacles next to the route. This cannot follow from the analysis because situation can be very specific, so those cases needs to be found by testing.

The disadvantage of a case based expert system is that it is possible that not every case is covered. Only the cases which follows from the possible actions are known and the cases which are found with testing.

In the scope is found that more than 90% of the barns has right angles between the walls. When there are only right angles between the walls, then the barn is not a very difficult environment for the robot. So for 90% of the barns the total sum of possible complex cases will be small and can be found by testing in a test barn. For other barns the total sum of cases can be much larger, but every case can be added afterwards by an expert when there is a high change that the case can be found in more barns.

8.4 Pseudocode

With the algorithm the movements of the robot are simulated. Every movement is done by a specific action. A route is defined by a set of waypoints which will be translated to a set of actions which will be simulated in the simulation.

To generate an action sequence for a route, where the robot moves from the first waypoint to all other waypoints, the algorithm needs to iterate over all the waypoints. For every waypoint the robot begins with a position and an orientation and the goal point which is the current waypoint. With the map and the position and orientation of the robot the position knowledge can be calculated, which contains information of the environment, like the distance to the first wall from the front of the robot.

With the position knowledge can be searched for the best case which fits the situation by iterating over all the known cases in the long-term memory, until a relevant case is found that satisfies all the conditions. Then the action from the case can be added to the action sequence. While the robot is not at the goal point, the position knowledge is again generated for the new position and an iteration is done over the cases to find the next action.

In the simulation are no uncertainties, so in the simulation only the most accurate action sequence in the best conditions is calculated. In practice the robot can encounter some problems when the conditions are worse. After an action sequence is made, the action sequence is evaluated and updated. This to make the action sequence robust so that the route also works in practice. In the section Action sequence evaluation this is explain in more detail.

In Figure 36 the pseudocode for generating the action sequence for a route can be found.

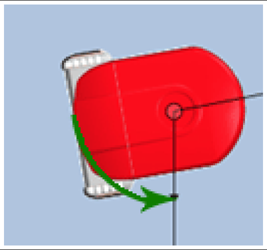
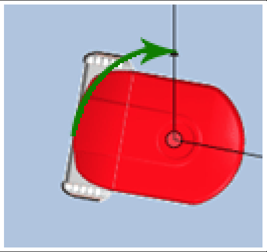
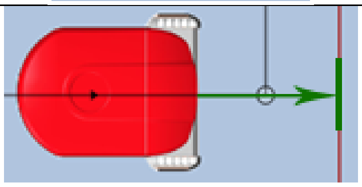
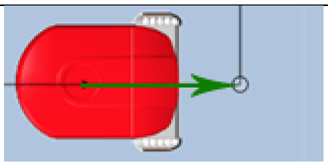
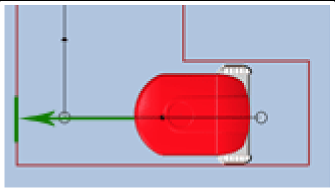
```
generateRoute(waypoints, startPositionRobot, startOrientationRobot, map){
        var routeactions = array()
        var position = startPositionRobot
        var orientation = startOrientationRobot
        foreach waypoints(waypoint){
                While ! at waypoint {
                        var action = generateAction(position, orientation, waypoint)
                        routeActions add action
                }
        }
        evaluateActions(routeActions)
        checkRoute(routeActions)
}
generateAction(position, orientation, goalPoint){
        var positionKnowledge = buildPositionKnowledge(position, orientation, goalPoint)
        case if (conditions) then { update position and orientation and return action}
        ….
        return error no case found
}
evaluateActions(actionSequence){
        evaluateCase1(actionSequence)
        evaluateCase2(actionSequence)
        ….
        evaluateCase10(actionSequence)
        …
}
evaluateCaseX(actionSequence){
        for( i = 1; i + 1 < actionSequence.length; i++){
           if actions [i] is wall following{
                if actions [i – 1] instance of … && …{
                        change distance of action … && change distance of action …
                }
           }
        }
}
checkRoute(routeActions){
        if last action does not end in charger, add path error
        if route does not start in the charger, add path error
        …
}
```
Figure 36 pseudocode for generating the route actions

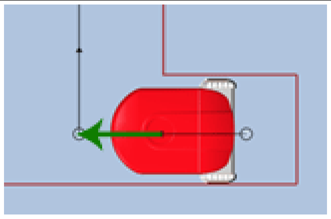
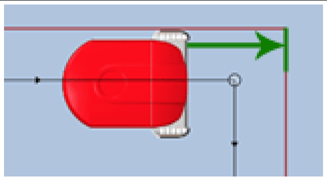
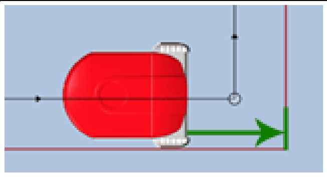
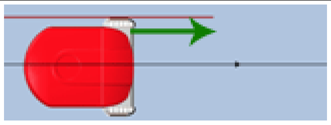
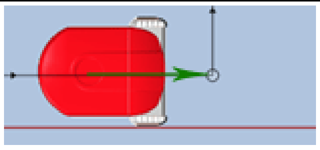
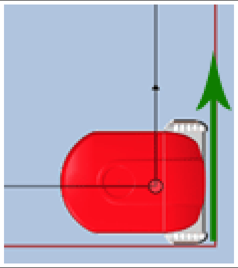
8.5 Route cases

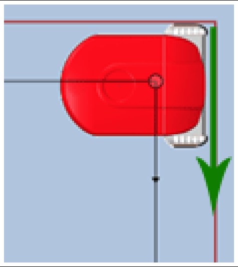
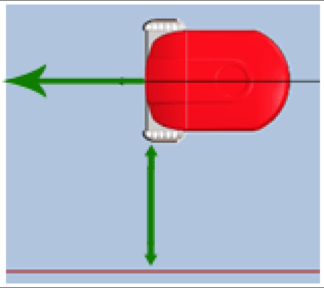
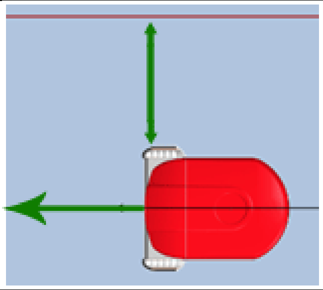
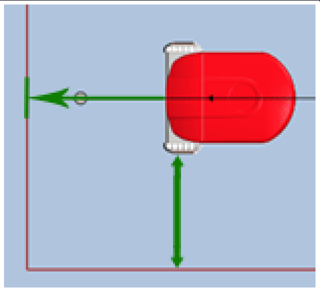
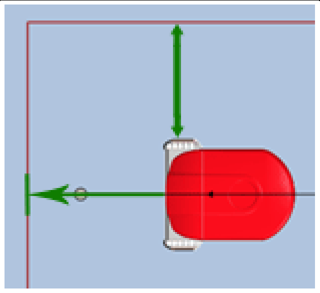
It is important that the system can find the correct action in every situation where the Lely Discovery can drive. In this section all the cases which can be matched with every situation in ideal conditions are listed.

In the scope is the barn layout defined, where is found that more than 90% of the barns have right angels between the walls. Also is defined how route points should be placed and that the route lines already should avoid obstacles. With this information the possible set of cases can be limited to the movement possibilities of the robot in ideal conditions. 18 possible movement types can be defined which are displayed in Table 8. From the test done in the workshop is concluded that only the first 16 cases are needed, so only the first 16 cases are used in the algorithm.

Table 8 cases for simple situations

#	Situation	Case
1		**Goal:** Change direction of the front of the robot to the next waypoint **Environment:** Direction is to the left and no obstacle on the left side to turn **Action:** Turn Left
2		**Goal:** Change direction of the front of the robot to the next waypoint **Environment:** Direction is to the right and no obstacle on the right side to turn **Action:** Turn Right
3		**Goal:** Move forward to the next waypoint and bump on the wall **Environment:** If in the correct direction and there is a wall at the end which can be used to bump on **Action:** Straight forward until bump
4		**Goal:** Move forward to the next waypoint with the center of the robot **Environment:** If in the correct direction and there is no wall at the end which can be used to bump on **Action:** Straight forward with distance
5		**Goal:** Move backwards to the next waypoint and bump on the wall **Environment:** If obstacle on the left, right and front and if there is a wall at the end which can be used to bump on **Action:** Straight backward until bump

6		**Goal:** Move backwards to the next waypoint with the center of the robot **Environment:** If obstacle on the left, right and front **Action:** Straight backward with distance
7		**Goal:** Follow the wall on the left to the next waypoint and bump on the wall in the front **Environment:** If in the correct direction, wall on the left within 1cm which can be followed and if there is a wall at the end which can be used to bump on **Action:** Wall follow left until bump
8		**Goal:** Follow the wall on the right to the next waypoint and bump on the wall in the front **Environment:** If in the correct direction, wall on the right within 1cm which can be followed and if there is a wall at the end which can be used to bump on **Action:** Wall follow right until bump
9		**Goal:** Follow the wall to the end of the wall **Environment:** If in the correct direction, wall on the right within 1cm which can be followed **Action:** Wall follow left with distance
10		**Goal:** Follow the wall on the right to the next waypoint with the center of the robot **Environment:** If in the correct direction, wall on the right within 1cm which can be followed **Action:** Wall follow right with distance
11		**Goal:** Change direction to the left with help of the wall in front **Environment:** If direction is left and wall within 1 cm in the front which can be used to bump on **Action:** Wall turn left with distance

12		**Goal:** Change direction to the right with help of the wall in front **Environment:** If direction is left and wall within 1 cm in the front which can be used to bump on **Action:** Wall turn right with distance
13		**Goal:** Follow the wall on the left parallel until at the next waypoint or the end of the wall **Environment:** If in the correct direction, wall on the left which can be followed with 1 – 3 m distance between the robot and the wall **Action:** Parallel to wall left with distance
14		**Goal:** Follow the wall on the right parallel until at the next waypoint or the end of the wall **Environment:** If in the correct direction, wall on the right which can be followed with 1 – 3 m distance between the robot and the wall **Action:** Parallel to wall right with distance
15		**Goal:** Follow the wall on the left parallel until the robot bumps on the wall in the front **Environment:** If in the correct direction, wall on the left which can be followed with 1 – 3 m distance between the robot and the wall and wall in the front which can be used to bump on **Action:** Parallel to wall left until bump
16		**Goal:** Follow the wall on the right parallel until the robot bumps on the wall in the front **Environment:** If in the correct direction, wall on the right which can be followed with 1 – 3 m distance between the robot and the wall and wall in the front which can be used to bump on **Action:** Parallel to wall right until bump
17	-	**Action:** Backward turn left
18	-	**Action:** Backward turn right

8.6 Action sequence evaluation

When an action sequence is generated with the algorithm, which follows the waypoints in an accurate way, the set of actions should be correct to finish the route in perfect conditions. When the conditions are worse the robot will end on some other place then in the simulation after a movement. For example, when the robot has some slip, the robot can stop too early which result in another end position than in the simulation.

The drawn routes already avoid obstacles, but can be drawn next to an obstacle. So when the actions are evaluate, the algorithm mainly needs to look to obstacles next to the route like corners of obstacles. Corners can be avoided by adding margins to a route action, so that the distance between the robot and the obstacle will be larger. Some corners can only be avoided by first turning away from the obstacle, move forward and then turning back, see case 3 from Table 9.

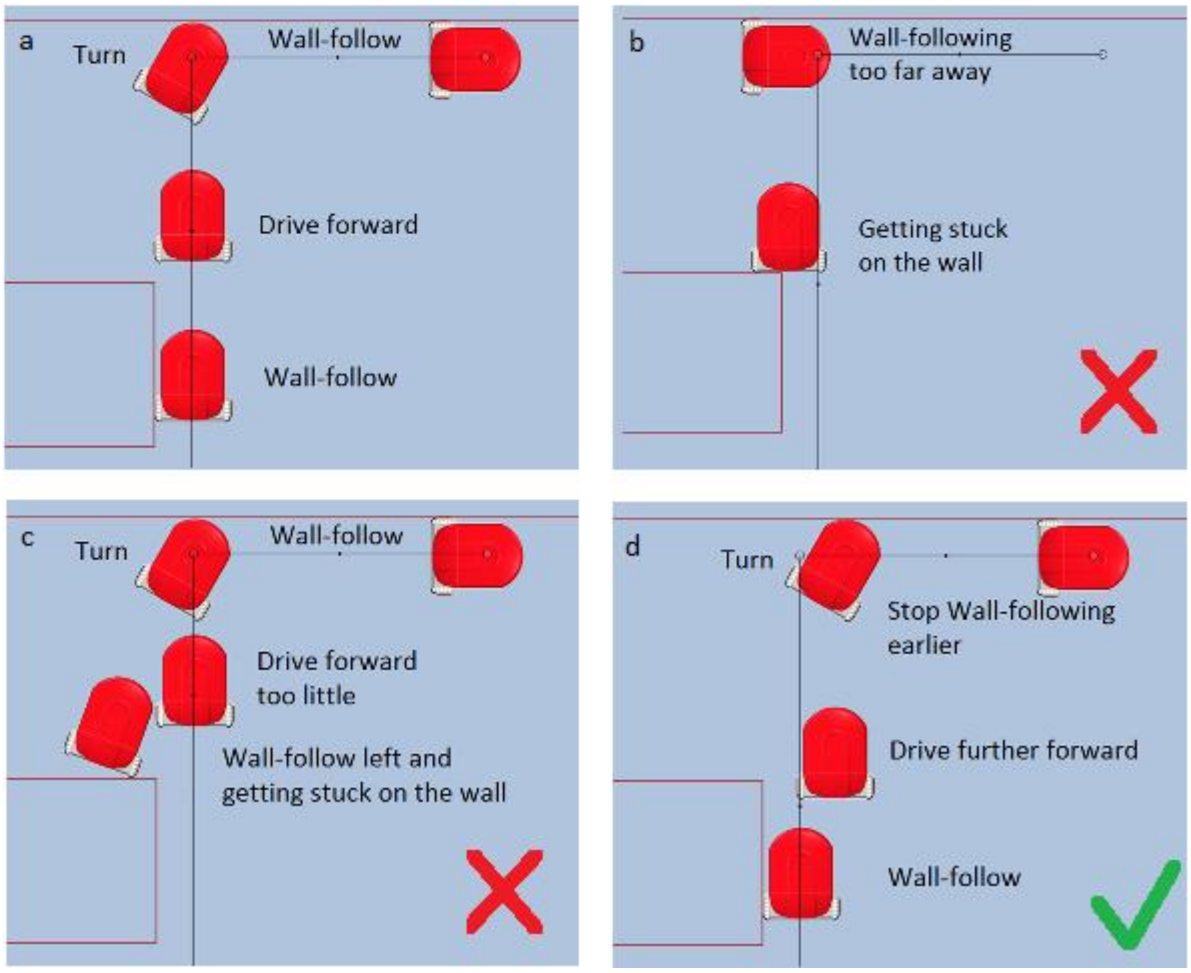


Figure 37 (a) generated action sequence which needs to be evaluated, (b + c) route goes wrong in practice (d) desired action sequence

In Figure 37 a, the movements of the robot in ideal conditions is shown of an example route where the robot moves in the simulation from the first waypoint to the second waypoint with a wall-following action.

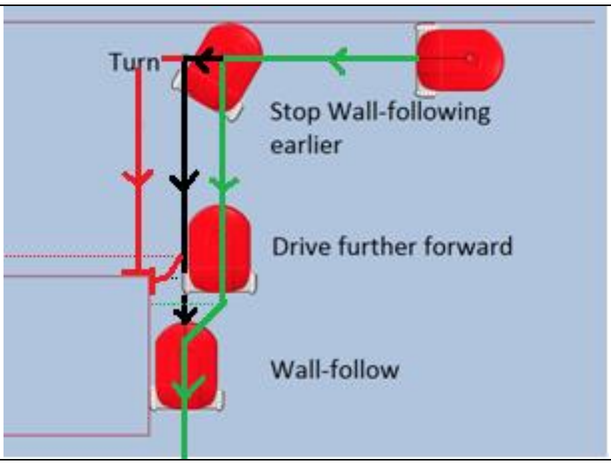
Then the robot makes a turn left to the next waypoint. The robot drives forward until it finds a wall on the right. Then the robot can follow the wall on the right.

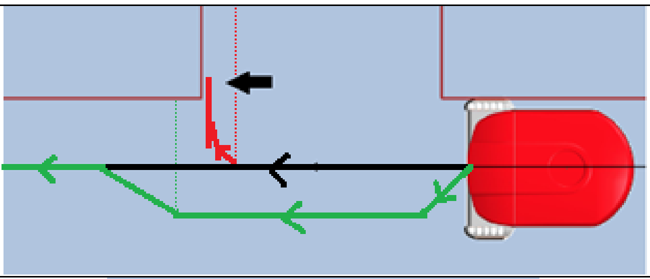
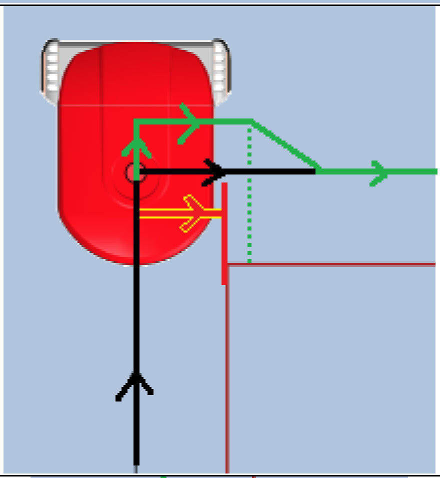
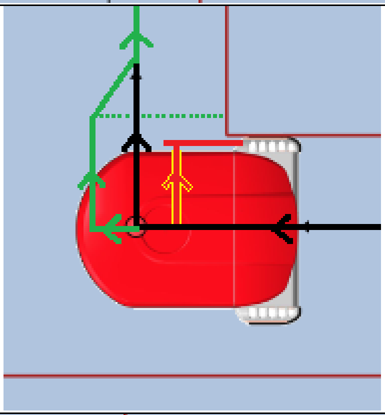
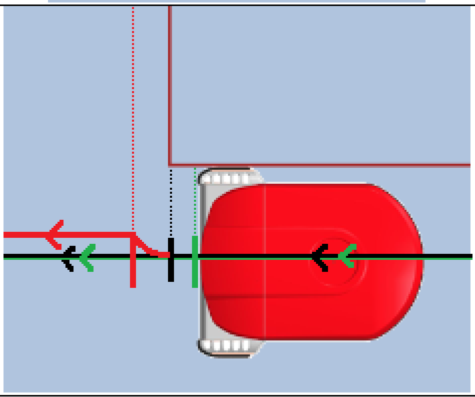
For the simulation the set of actions is correct, but in practice the robot will get stuck on the wall most of the time. If the robot ends after the second waypoint and then turn and drive forward, the robot will get stuck on the corner of a wall as in Figure 37 b. To avoid the corner a margin can be subtracted from the first wall-following action so that the robot never get stuck on the wall. With the drive forward action, the robot can end too early. This will give some problem to the next wall-following action, which will drive forward and drive to the left until it finds the wall as in Figure 37 c, but this time the robot will get stuck again on the wall. The desired action sequence which is generated after the action evaluation is shown in Figure 37 d, where the robot stops wall-following earlier and drive with a larger distance forward so that the robot will find the wall on the right when the robot wants to do wall-following again.

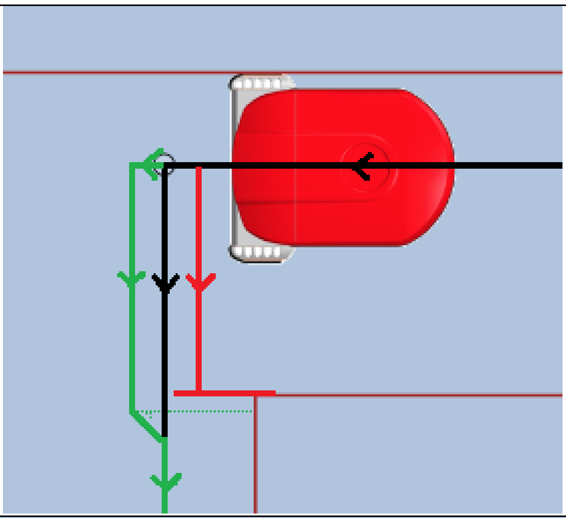
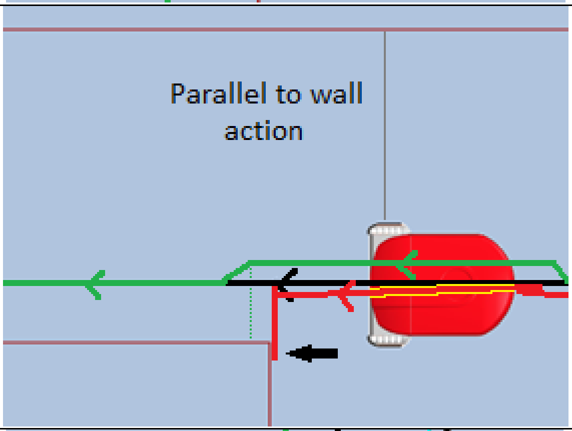
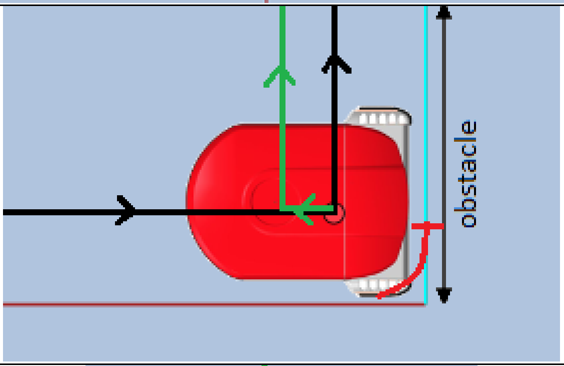
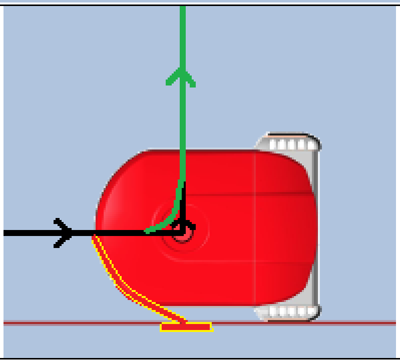
For a barn with right angles between the walls, the set of complex cases will be small. This because only one type of corner needs to be avoided. In Table 9 all the complex situations are described which are found for barns with right angles between the walls which covers approximately more than 90% of the barns. For more complex barns with different type of corners, it will be harder to find all complex cases, those barns are out of the scope for this project. Every type of corner can be avoided if there is enough room for the robot to move around, so it should be possible to find all the cases for other type of corners.

In Figure 36 the pseudo-code can be found for evaluating the action sequence. For every case which is described in Table 9, an iteration should be done over all the actions to determine if the case applies on a specific part of the action sequence. If a part of the action sequence is found which satisfies the conditions of the case, some parameters should be changed like the distance of an action or a turn action should be added before and after the action depending on the case. From the cases described in Table 9 can be found if some actions need to be changed to increase the robustness of some part of the route.

Table 9 cases for complex situations

Situation	Case	#
 Turn Stop Wall-following earlier Drive further forward Wall-follow	If one of the next actions is wall follow, drive less forward so that the robot can follow the next wall	1
	If next action is wall follow, drive further forward so that the robot can follow the next wall when the robot has a lot of slip	2

	If there is an obstacle next to the route add a small turn action with some distance to the other side so that the robot can safely drive around the obstacle	3
	If there is an obstacle on the side and the robot needs to turn to that side, drive further forward	4
	When the next action is turn, drive further backward so that the robot doesn't get stuck on the wall when the robot did not drive enough backwards	5
	If the action end at the end of the wall, drive less forward, so that the robot does not try to do wall-follow after the end of the wall when the robot drives to far	6

	If one of the next actions is wall follow, drive further forward so that the robot can follow the next wall	7
	If there is an obstacle next to the route decrease the distance between the robot and the wall to avoid the obstacle	8
	If the robot needs to turn but there is an obstacle in front, the robot first needs to drive backwards and then make the turn	9
	When making a turn and there is an obstacle on the other side of the robot. Add a radius to the turn, to prevent scrapping with the back of the robot the wall	10

8.7 Conclusion

In this chapter is described that the actions of the Lely Discovery has an infinite set of possible actions. When the continuous parameters are transformed to discrete values, a large set of actions is retrieved which makes it hard to calculate every possible path.

In 8.3 is looked at a case-based reasoning (CBR) algorithm which fits a new situation with known cases to find the relevant action. The advantage of a CBR is that it can handle complex situations and that it make use of a finite set of cases. The variation of the situations which fits those cases will be very large.

The disadvantage of a CBR is that it is possible that not every case is covered. With testing the most cases will be found but some cases can be missed. The possible set of cases in ideal conditions is limited and follows directly from the possible action types. The set of complex cases where corners need to be avoided is not limited, but for 90% of the barns only one type of corner need to be taken into account. When there is one corner type, the set of complex cases is also limited and can be found by testing.

There is decided that a CBR would be the best solution. This algorithm is implemented and the test plan and results can be found in the next chapters.

9 Test plan

The research is about how the user-friendliness and effectiveness of the route teaching for the Lely Discovery platform can be improved. A complete new route programming method is designed and implemented. To compare the current system and the route drawing system on user-friendliness and effectiveness a user study is done and the routes of the users are tested on the robot. First the results of the pilot are shown. Then is described how the user study is performed. At the end is shown how the routes of the users are tested on the robot.

9.1 Pilot

With the pilot is checked if the generated action sequence works in the workshop and if more cases should be implemented. Also is checked if users where able to draw a route with the tool and if the tool missed some features.

During the pilot 5 route are tested on the robot which can be found in Appendix C. The first four tests ended successfully in the charger. The last test ended after the charger because the route line of the drawing did not directly follow the wall. This caused that the algorithm added two wall following actions where the second wall following action was only 1 cm and ended in the charger. When those two actions are combined to one action, then the robot will be able to end in the charger. During some parts of the route when the robot turned away from the wall, the back of the robot scraped against the wall. This could be solved by adding two new cases, case 9 and 10 in Table 9. Also a bug in the code was solved which caused that the robot made a turn of 540 instead of 180 degrees. The robot was able to drive the routes very accurately and did not get stuck on some obstacles. With those 5 tests the author gained a lot of trust in the system and that it would work in practice.

Three users tried to draw a route with the tool. After some explanation and some trials, they were able to draw their own route. The route of the last test in the workshop (see Appendix C) was also drawn by one of the users. The visualization of the movements of the robot in the feedback module and the actions where difficult to understand for users. Without this visualizations and details about the actions, users where able to make a good route. When the user is able to visualize the movements and is able to understand the actions, the route will be more robust and will contain less problems. To help the user a little bit, route errors are added with some explanation about the error in the tool. Those errors are generated by a route checker which evaluate every route part. This method can only indicate known problems, but for the prototype this is a good alternative for users when they cannot visualize the movements automatically and when they don’t get detailed information about the actions. With some instructions before the user test, users should be able to understand how the tool works.

9.2 User Study

In the user study, users should draw a route on paper and then draw the route in the tool. Users will get 4 maps with different configurations of the workshop where different parts of the workshop are selected and different obstacles are added. The last map is a map which is used for a test barn. Drawing the map is not part of the test, a correct drawing of the map is given.

9.2.1 Participants
To compare the results of the route drawing tool with the current route programming method, minimal 10 people will be recruited which already have some experience with the current route programming method. Also 10 students will be recruited which have no experience with the current route programming method. So there will be 20 users for the user study.

9.2.2 General objective
The current route programming method and the route drawing tool should be analyzed on user-friendliness and time needed to program the routes. The routes users wants to draw should be compared with the routes users actually draw in the tool.

9.2.3 Task
The participants should draw one route which covers the whole ground of the barn. First they should draw the route on paper and then draw the same route in the tool. They get 5 maps with 4 configurations of the workshop where different obstacles are added and 1 map of a test barn. After they have drawn a route, users are asked to generate the action sequence and check in the feedback tab if the generated action sequence will work in practice.

9.2.4 Preparation
Before the test starts, instructions for the users about the tool and the test should be prepared. The following instructions and preparations needs to be done:

1) Minimally three days before the test, every user gets an explanation of how the tool works.
2) 5 maps with different configurations of the workshop will be drawn in the tool and exported to a xml-file.
3) The instruction document is printed including the maps on paper.
4) Users are instructed that they should draw one route that covers the whole ground on the given map by the robot.
5) Users are instructed that they should draw the same route in the tool as on paper.
6) Users are explained that after the second drawing on paper, they can get again an explanation on how they should draw the routes.
7) Users are instructed that every route should follow the walls, drawn parallel to walls and use walls in front for bumping as much as possible.

9.2.5 Procedure
When the preparations are done and users are instructed three days before, the test can be started. Only one user can do the test at the same time. The procedure of the test will be the following:

1) The user gets the instruction document including a print of all the maps (same order for every user, from simple to difficult), a pencil, ruler and eraser from the experimenter.
2) The user answer the questions in the instruction document about their background. If they have experience with the current route programming method, they answer the questions about the current route programming method for the Lely Discovery
3) The user draws a route on paper.

4) The experimenter imports the map in the tool while the user draws the route on paper.
5) If this is the second drawing, the users receives explanation how the tool works again.
6) The user is asked to draw the route in the tool.
7) The experimenter start a timer.
8) When the user indicates that the generated action sequence should work, is green in the feedback tab on every part of the route and the route covers the whole ground of the barn, the experimenter stops the timer and writes down the time.
9) The user export the route to an xml-file.
10) The user can give some comments about the route in the instruction document.
11) Step 3 – 10 is repeated for all the maps.
12) The user answers the questions in the instruction document about the route drawing tool.

9.2.6 Instructions and questions

Before the test the instruction document is given to the users. In the instruction document the procedure and questions are described (See Appendix C for the instruction document).

In the instruction document first some questions are asked about their background, so that can be measured whether specific skills affect the results. If this is the case, it is possible that only certain groups of users can use the tool. The rest of the document consists of two parts where the first part is about the current route programming method and the second part about the route drawing tool. This is done to separate the questions of both methods. Some people don't have experience with the current route programming method and they can't use the current system, they can skip the first part. The questions about the current route programming method is in the first part so that users are not influenced by the user test of the route drawing tool, on the moment that they answers the questions about the current route programming method.

To compare the usability of the current route programming system with the route drawing tool, a standard computer system usability questionnaire (Lewis, 1995) for both systems is included.

For the route drawing tool the usability of every module can be measured. The current tool consist of two modules for the users, a route drawing module and a feedback module. To measure the usability of those two modules, a Component-Based Usability Questionnaire (CBUQ) is used to measure the perceived usability of a specific part of an interactive system (Brinkman, Haakma, & Bouwhuis, 2009).

The user study cannot be done with a lot of people, so the results will not be very significant. For this reason some open questions are included for both parts.

9.3 Routes from users on robot

9.3.1 General objective

When the users have drawn the routes with the drawing test, the routes can be evaluated whether the Lely Discovery drives the route as drawn and whether the Lely Discovery can finish the route without problems.

9.3.2 Task
With the user test, 20 users have created a route for 4 different configurations of the workshop. This delivers us 80 routes which can be used to generate an action sequence and which can be exported to the Lely Discovery. The Lely Discovery can drive the route by executing the action sequence. When the Lely Discovery executes the action sequence, it can be checked whether the Lely Discovery is able to finish the route, drive the route without problems and drives the route accurately.

9.3.3 Preparation
Before the routes are tested on the robot, the routes should not contain any green or orange parts in the tool. During the user test, the simulation of the robot movements and the visualization of the actions was turned off, this because the simulation for now is too simple and the visualization of the actions are too difficult for users to understand. The experimenter will simulate the route in the tool and locate possible problems before the route is exported to the robot. For every route the following preparations should be done:

1) Check if the route is displayed green or orange, when a part is displayed red, simulate the route in the tool and try to fix the problem and write it down.
2) When the route is displayed green or orange in the tool, simulate the route in the tool and check for possible problems. Write down the possible problem and the position of the problem in the route.
3) Print the map and the route on paper for every route which is displayed green and orange in the tool, the print should be used to draw the driven route of the robot in the workplace.

9.3.4 Procedure
To test the routes on the robot, the action sequence needs to be exported from the tool to the robot. Then the actions can be executed on the robot and checked how the Lely Discovery drive the route, whether the Lely Discovery encounter problems during the route and whether the Lely Discovery is able to finish the route. The procedure is the following:

1) Run the route on the robot by exporting the actions from the tool to the robot.
2) While the robot drives the route, write down the driven route on the printed map by drawing a route line of the position of the front of the robot during the test.
3) Write down on the same paper where the robot must correct for problems during the route by drawing a circle on the map.
4) Check whether the robot is able to finish the route in the charger and write it down.
5) Save the data from the robot which can be played in a simulation and shows the estimate of the position of the robot during the route.
6) Repeat step 1 – 5 for all routes which are displayed green and orange in the tool.

10 Evaluation

In this chapter the results are shown of the user test and the results of the test on the robot. First some information about the background of the participants is given. Then the results of the system usability questionnaire, open questions and the Component-Based usability questionnaire are presented. After this the results are shown of the routes in practice which are drawn by the users. At the end the results of the user study and the results of the routes are combined and an overview of the interaction between education and other factors which have effect on the result are given.

10.1 Background information participants

The user test is done by 20 participants where 13 participants have experience with the current route programming system. See Table 10 for the information about the participants.

Table 10 Background information participants

User	Job Title	Education	Experience current system
1	Student	VWO	No
2	Tester	HBO	Yes
3	Student	BSC	No
4	Test coordinator	MSC	Yes
5	Embedded software engineer	HBO	Yes
6	-	HBO	Yes
7	Software developer	MSC	No
8	Service support	HBO	Yes
9	Embedded software engineer	MSC	Yes
10	Test/service technician	MBO	Yes
11	Product specialist (TSS)	-	Yes
12	Software engineer	PHD	No
13	Manager product development	HBO	Yes
14	Student	VWO	No
15	Student	VWO	No
16	Principle engineer	HBO	Yes
17	Student	VWO	No
18	Embedded software engineer	MSC	Yes
19	Test service engineer	MBO	Yes
20	Technical service support	-	Yes

10.2 Route results

In the user study, 20 users drew a route for 4 different configurations of the workplace. 79 routes where drawn by users which have been tested on the robot. One user had no time for the fourth route. First the changes on the routes made by the author are shown. Then the results and the problems which the robot encountered during the routes are represented.

10.2.1 Route changes

Before the routes where tested on the robot, some routes of the users where changed. The changes are categorized into two types, simple and complex changes. Simple changes are changes which could be automatically corrected by the tool and complex changes are changes to avoid bugs in the robot software or in the tool, or changes which should be found by the checker in a next version of the tool and should result in a red line.

Table 11 user route changes

Map	Routes	Routes without changes	Routes with simple changes	Total Simple changes	Routes with complex changes or not possible	Total Complex changes
1	20	11	9	11	0	0
2	20	14	6	8	1	1
3	20	2	6	7	18	26
4	19	7	2	2	11	8
Total	79	34	23	28	30	35

For the routes in the first two maps, mostly the start and end point are corrected. It was for the users not clear how they should end and start in the charger. Map 3 and 4 contained a very complex corner. The checker in the tool was not advanced enough to see the problems in this corner, so the most routes in this corner needed to be corrected. See Figure 38 for an example of a route in the complex corner. In the simulation the robot drives into the gap, and then needs to turn to the left. In the simulation the robot turns 270 degrees to the right where the robot will bounce on the corner right of the robot in practice. In practice it was also not possible for the robot to drive into the gap from the top in a reliable way. See Table 31 in Appendix D for all the changes in the routes.

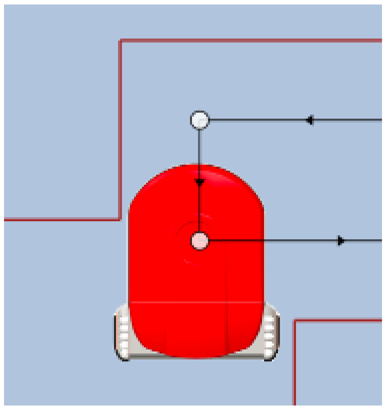


Figure 38 complex corner in the workshop

10.2.2 Results

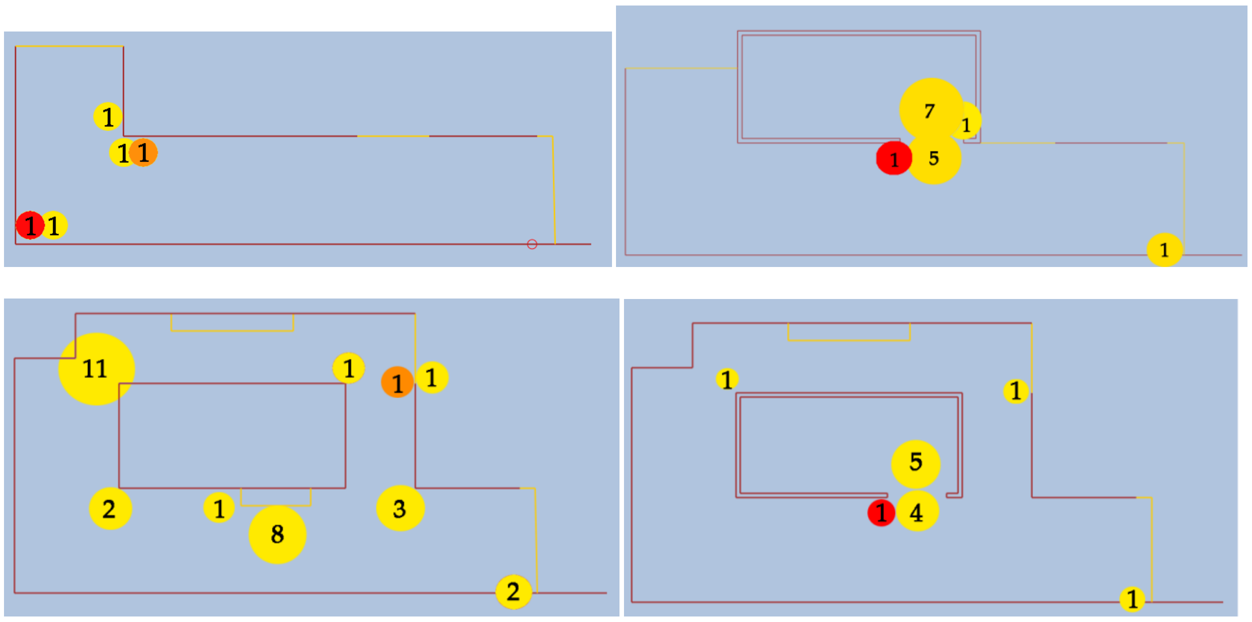
From the 79 routes drawn by the users, 68 routes were usable for testing on the robot. Some of the other 11 routes are tested but the author concluded that it was not possible for the robot to drive into the gap from the top as in Figure 38. From the 68 tested routes the robot was 64 times able to finish into the charger, 94% of the tested routes and 81% of all drawn routes. See Table 12 for the results.

Table 12

Map	Routes	Finished in charger	Finished without problems	Finished with small problems	Finished but missed part due to problems	Did not finish	Not possible	Bug in robot software
1	20	19 / 20	15	3	1	1	0	0
2	20	19 / 20	10	9	0	1	0	0
3	20	13 / 13	4	8	1	0	7	0
4	19	13 / 15	6	7	0	1	4	1
Total	79	64	35	27	2	3	11	1

10.2.3 Route problems

The robot encountered some problems during the routes. In 51% of the tested routes, the robot finished in the charger without problems. In 91% of the tested routes, the robot finished in the charger without missing a part of the route. During the routes on map 2 and 4, the most occurring problem was going in and out of the middle of the cubicle. There was almost no margin on the left and right side of the robot so with a small error, the robot already bounced on the corner of the opening. In the routes on map 3, the biggest problem was the corner of the top left large cubicle (the complex corner of Figure 38) and the corner of the drinking trough as obstacle. See Table 30 for all the problems which the robot encountered.



In three cases the robot could not finish the route and in one case the robot could not finish the route due to bug in robot software. See Figure 39 for the route where the robot crashed in map 1. As you can see in the simulation right, the robot drives a little bit too far (action of 8 cm) which cause the front right of the robot to move into the wall. In practice this was not a problem because the robot needed to drive

20 cm backwards to make a turn more reliable for the robot, the routes are not corrected afterwards for driving 20 cm backwards before every turn when close to a wall. So the problem in the simulation was not the problem in the test on the robot in practice. In practice the robot software wanted to drive further forward while the robot already bounced on the wall. To drive more forward the software tried to correct and used avoid algorithms which caused that the robot got lost.

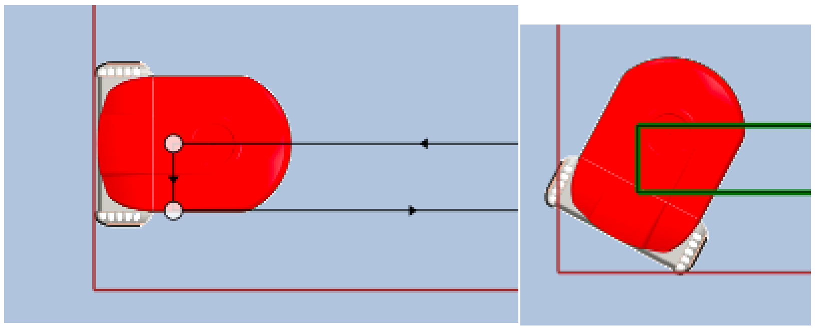


Figure 39 Route 13 in map 1 where the robot crashed, right the simulation where the robot should end after the move forward action for 8 cm

In map 2 and in map 4 the robot crashed one time on the wall of the large cubicle when going inside of the middle of the cubicle. Mostly the users first followed the wall and then turned into the cubicle which is more reliable. In route 11 of map 2 the robot had an error of 0.5 meter which the robot could not correct before going into the cubicle. See Figure 40 and Figure 41 of the routes where the robot was not able to get inside the cubicle.

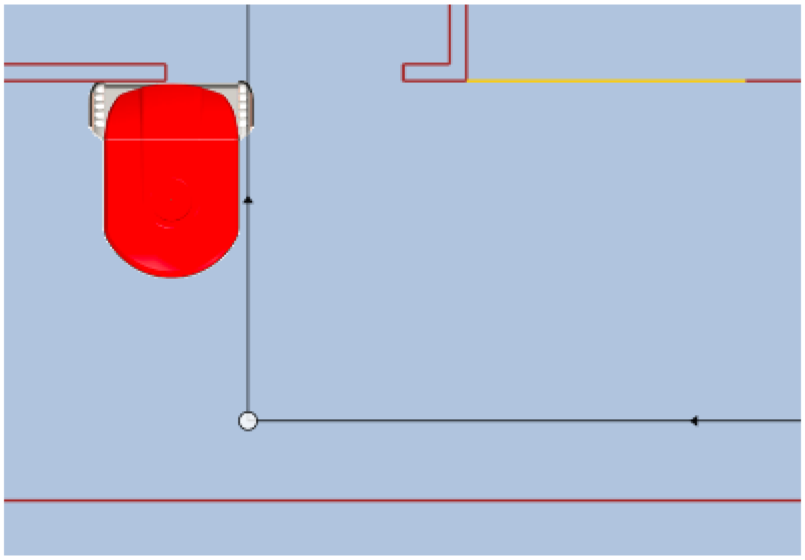


Figure 40 Route 11 in map 2 where the robot did not get inside the cubicle

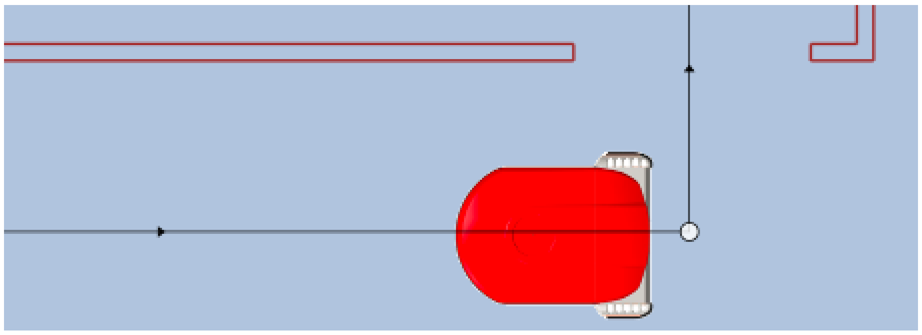


Figure 41 Route 2 in map 4 where the robot did not get inside the cubicle

10.2.4 Conclusion

Before the routes of the users where tested on the robot, a lot of changes needed to be made on the routes. The most changes, the simple changes, can be solved by improving the tool by automatically correcting the routes. When the simple changes automatically can be solved with the tool, still 38% of the routes needs to be corrected, the complex changes.

Some of the complex changes are done to avoid bugs in the algorithm or in the robot software. The most complex changes are to avoid problems in a very difficult corner (see Figure 38). This problem need to be solved to make the tool useful. A solution can be to improve the feedback checker in the tool, this can help users to find in an early stage problems and display a red line in this corner. Another solution is to deliver users a better insight in how the robot drives the route and help them to see how reliable a move is. This can be done by automatically simulate the robot movements in the tool or by making a reliability calculation for every route part and display the results in the tool.

During the test on the robot, the robot encountered in 49% of the routes some problems. The maps contained 2 parts, in and out of the center and the difficult corner on the top left, which can’t be driven by the robot on a reliable way. The instruction for the users was to clean every part of the barn so they did not avoid those difficult parts. Those parts are the cause of 56% of the total problems, which will also be a problem for the current route programming method. The most other problems shouldn’t be there and are mostly the result of a strange or wrong drawing of the users. One of those strange drawings also resulted in a crash of the robot. This can be solved by training the users and by automatically align route parts with walls and check if users draw in the red area.

The results are not convincing but provides a reasonable result to work on a second prototype. With the second prototype the tool should automatically correct the simple changes and a solution should be implemented to avoid the complex changes. To see if the route problems can be reduced, the second prototype should automatically align the route parts with the walls and show users that they drew in the red area. Also a better training should be given to reduce the problems or by making a help function and instructions in the tool.

10.3 Route draw and drive time

During the user study the time the users needed to draw the routes in the tool was measured. Also the time which the robot needed to drive the route was measured during the test on the robot. In Table 13 the result can be found. From the results can be seen that the factor drive time/ draw time increases for almost every map. This can be explained by the learning curve of the users and the overlap of the maps. When users draw a route on a place where they already have drawn a route, they already know how they need to draw the route on that place.

Table 13 Average draw and drive time

	Map 1	Map 2	Map 3	Map 4	Map5	Sum avg map 1-4
Avg drawing time	300	358	430	373	620	1462
Avg drive time	616	839	1001	1189	-	3646
Drive time / draw time	2,05	2,34	2,33	3,19		2,49

It can be assumed that making a route with the current system cost minimal the drive time of the robot. From the results can then be concluded that making a new route with the drawing system takes 2-3 times less than the current system. The time to draw a route with the route drawing system can be optimized. This can be done by adding functionality to copy route parts, optimize drawing functionality and automatic aligning with the walls.

When the draw time is compared with education, one significant result can be found. In Figure 42 you can see the scatter plot for the interaction between the draw time of map 4 and the computer experience of the users. When users have more computer experience, the draw time is much lower. For the first three maps there was no significant difference between the draw time and computer experience of the users. The learning curve of more computer experienced user is steeper so after some routes done, they draw the routes faster. It can be expected that less computer experienced users also will learn to draw a route faster and will reach a factor of more than three between drive time and draw time after more routes done.

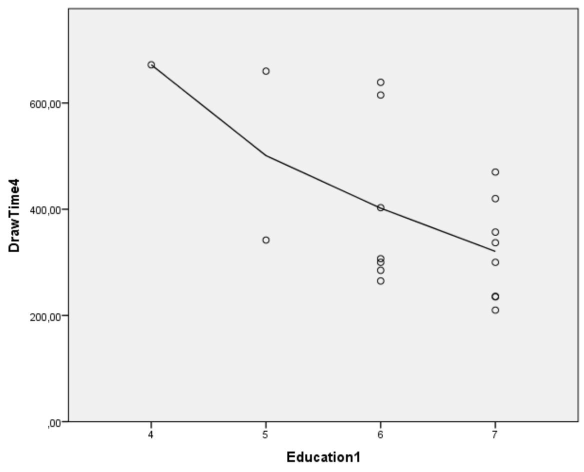


Figure 42 Interaction between computer experience and the draw time for map 4

10.4 Route drawing on paper compared

Before the users drew the route in the tool, they drew the route on paper. This gives the possibility to see if users can draw a route in the tool how they want a route to be done.

Table 14 Average route parts drawn on paper and in the tool

	Map 1	Map 2	Map 3	Map 4
Average route parts drawn on paper	13,2	24,5	28,45	36.35
Average route parts drawn in tool	14	25,2	30,5	38,12
Average same parts on paper and in the tool	10,6	21,55	22,4	30.82

In Table 14 can be seen that the average difference in route parts drawn on paper and in the tool is not so large. The users were able to draw more than 80% of the route parts on paper also in the tool. The other 20% of the route parts are drawn differently. When users drew the route in the tool, they got more insight in the area coverage, so they changed the route to cover every part of the barn. This explains also why users drew more route parts in the tool then on paper.

10.5 Area cleaned

In the tool users could see which parts are cleaned by the route. The pixels of the area which are covered by the route are counted and divided it by the total pixels in the whole area. In Table 15 you can find the average area coverage.

Table 15 Average area cleaned for every map in the drawing tool

	Map 1	Map 2	Map 3	Map 4
Average area cleaned	99.74%	99.11%	97.53%	97.82%

The average area cleaned by the routes was more than 98%. In some situations users avoided specific corners or obstacles. This is why users not always reached 100% coverage. The lowest area coverage of one user was 83.1% for the third map.

The results shows that users are able to make a route which cleans almost every part of the barn.

10.6 System usability

Users could rate the current system usability and the usability of the drawing system in the instruction document with 19 questions on a 7 point Likert scale. There was a group of 7 students which has no experience with the current system, they only rated the usability of the drawing system. The other group of 13 people rated first the usability of the current system and after they finished the user test, they rated the usability of the drawing system.

10.6.1 Results

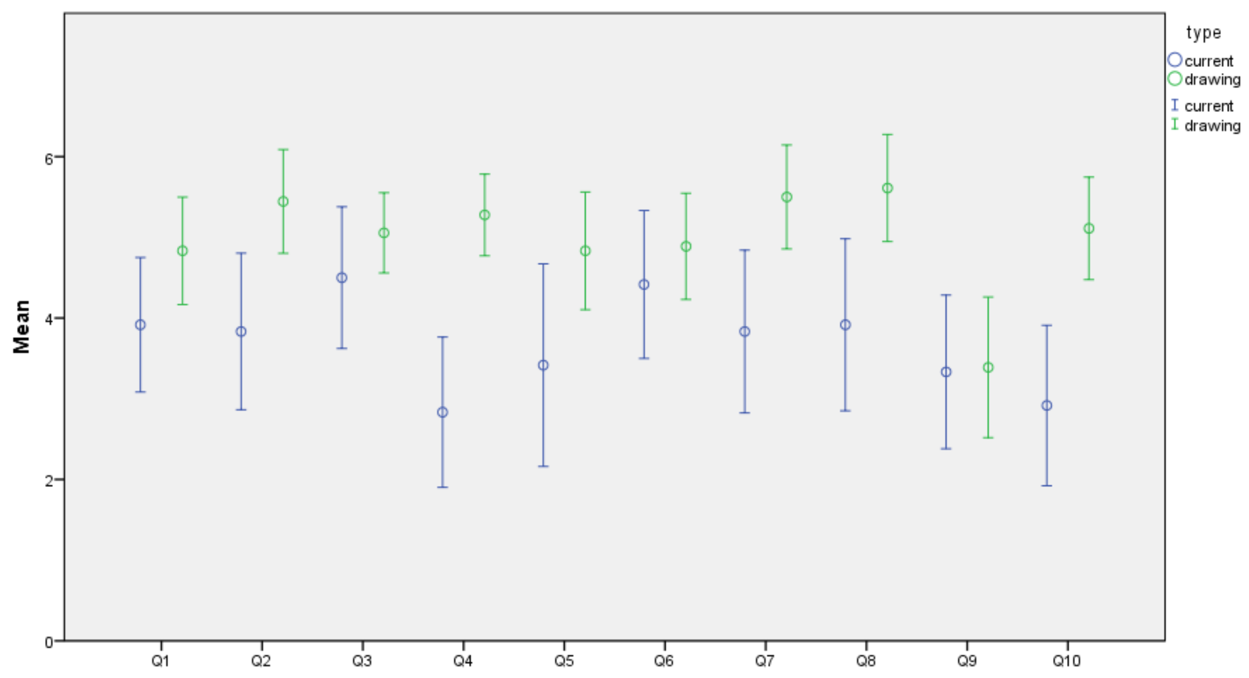
The means of the usability for the two systems are compared, this is done with a T-test. The two groups the current system and the drawing system are related because the participants of the first group are also in the second group. When two groups are related, a paired sample T-test needs to be done to compare the means. In Table 21 in Appendix D the results of the paired sample T-test can be found and in Table 16 the questions can be found.

Table 16 usability questions

1.	Overall, I am satisfied with how easy it is to use this system
2.	It was simple to use this system
3.	I can effectively complete my work using this system
4.	I am able to complete my work quickly using this system
5.	I am able to efficiently complete my work using this system
6.	I feel comfortable using this system
7.	It was easy to learn to use this system
8.	I believe I became productive quickly using this system
9.	The system gives error messages that clearly tell me how to fix problems
10.	Whenever I make a mistake using the system, I recover easily and quickly
11.	The information (such as online help, on-screen messages, and other documentation) provided with this system is clear
12.	It is easy to find the information I needed
13.	The information provided for the system is easy to understand
14.	The information is effective in helping me complete the tasks and scenarios

15.	The organization of information on the system screens is clear
16.	The interface of this system is pleasant
17.	I like using the interface of this system
18.	This system has all the functions and capabilities I expect it to have
19.	Overall, I am satisfied with this system

For all the pairs can be said that when $p < 0.05$, the difference in mean is significant. From the results in Table 21 can be seen that the difference in mean for question 4,7,8,10,15,16,17 is significant. For all those questions the mean of the route drawing system is 1 to 2 points higher than the current system. See Figure 43 for the means for every question on both systems.



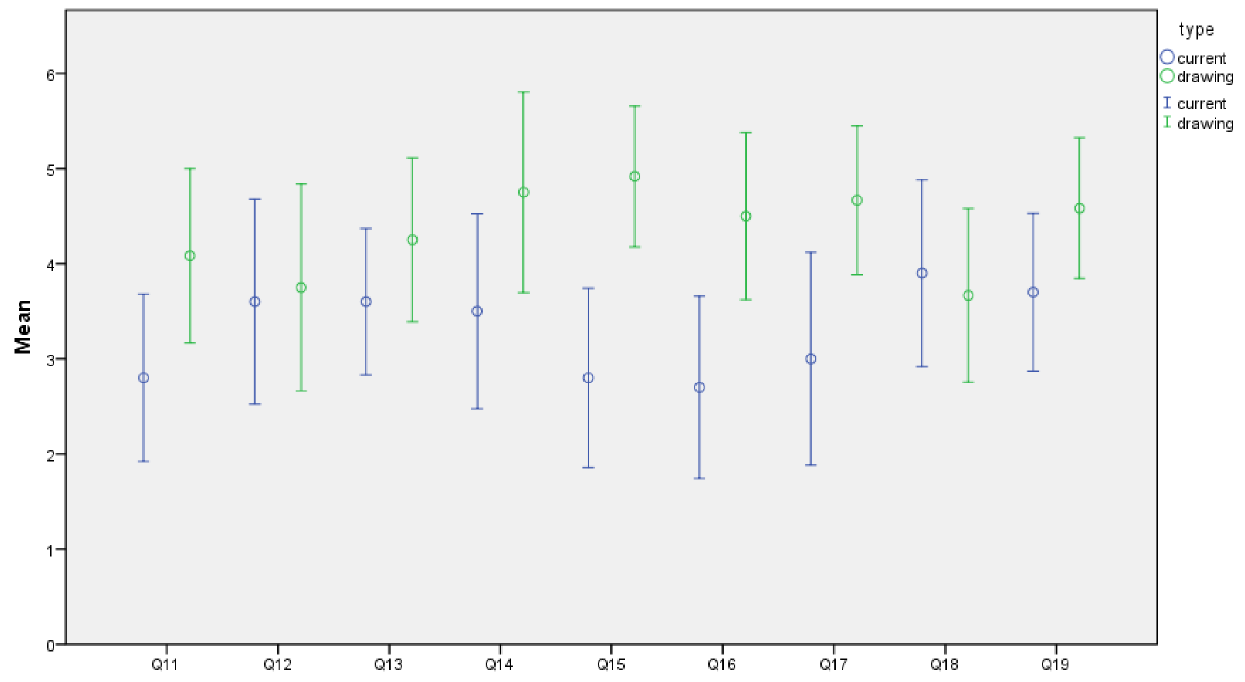


Figure 43 means usability questions of both systems

10.6.2 Conclusion

According to the result of the usability questionnaire users can complete their work more quickly, learn to use the system more easily, are more productive, recover more easily and quicker when making a mistake. In chapter 10.3 can be seen that users are able to complete their work more quickly. This is in line with the results of the usability questionnaire.

Users also thinks according to the usability questionnaire that the organization of information of the system is more clear, the interface is more pleasant and they like the interface of the route drawing system more than the current system. In the chapter Route drawing on paper compared can be seen that users are able to draw the same route in the tool as they drawn on paper. They are also able to clean the most parts of the barn, see the chapter Area cleaned. With the interface of the route drawer, users are able to complete their task.

10.7 Open questions

During the user test, users filled in answers to the open questions in the instruction document. In Appendix D the answers to all the questions can be found.

From the answers to the open questions can be concluded that most negative aspects of the current route programming method are turned into positive aspects with the route drawing method, like programming speed, route modification and route overview. The most negative aspects of the route drawing method are zooming, unspecific feedback and miss clicks. Those negative aspects can be solved by improving the tool.

The positive aspects of the current route programming method is mostly what users miss in the route drawing method. With the current route programming method, you see directly what you get, but with the route drawing method, users have no idea about how the robot will drive the route. A good simulation of the route could replace this positive aspect of the current route programming method.

10.8 Component-Based Usability Questionnaire

With the Component-Based Usability Questionnaire (CBUQ) it is possible to measure the perceived usability of a specific part of an interactive system (Brinkman, Haakma, & Bouwhuis, 2009). This method is used in the user test to test the usability of two components, the drawing component and the feedback component of the drawing tool.

10.8.1 Reliability Usability scales

Table 17

	Cronbach's alpha
Drawing component	0,84
Feedback component	0.91

The Cronbach's Alpha is used as an estimate for the reliability of the measurements. To simplify the analysis and because of the high cronbach's Alpha the analysis was done with the average rating for each component.

When the Cronbach's Alpha is > 0.7, the reliability is good and when the Cronbach's Alpha > 0.9 the reliability is Excellent. In Table 17 can be seen that Cronbach's alpha for both components > 0.7, the reliability of the test scores is good.

10.8.2 Usability rating - Hypothesis testing Component rating

Table 18

One-Sample Statistics

	N	Mean	Std. Deviation	Std. Error Mean
AverageDrawingComponent	20	5,12	,789	,177
AverageFeedbackComponent	20	5,07	,977	,218

In Table 18 can be seen that the mean for both components is around the 5 and the std. deviation < 1.

10.8.3 One-Sample t-Test

With the one-sample t-test the mean can be compared with a hypothetical value, and can be seen if the difference of the mean is significant. If the significance is lower than the critical value alpha (0.05), then the hypothesis can be rejected and can be concluded that the difference between the means is significant.

The hypothetical value is 5.29 which is based on (Brinkman, Haakma, & Bouwhuis, 2009), where 5.29 was the breakeven point of the probability that a rating would come from distribution of a set difficult to use component or from an ease to use components.

Table 19

One-Sample Test

	Test Value = 5.29					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
AverageDrawingComponent	-,982	19	,338	-,173	-,54	,20
AverageFeedbackComponent	-,984	19	,337	-,215	-,67	,24

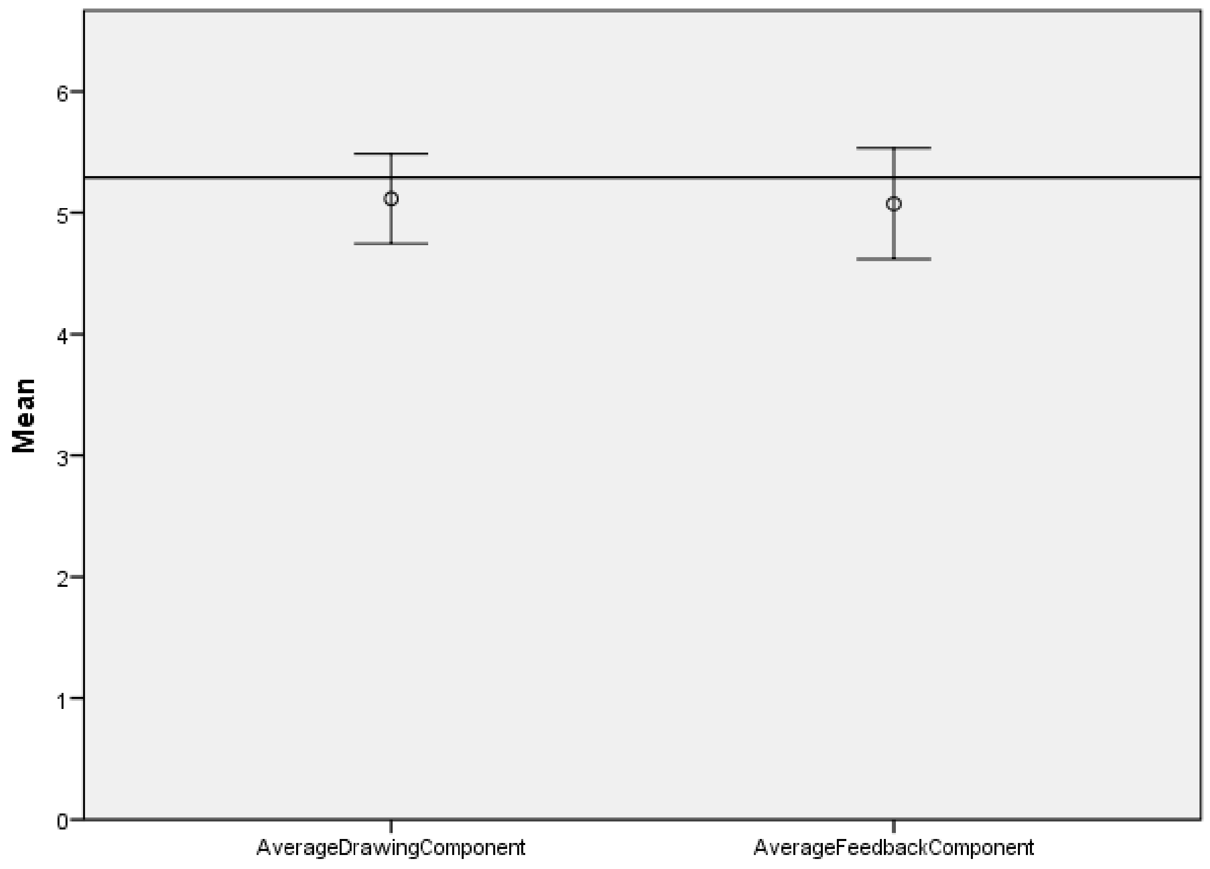


Figure 44

From Figure 44 and from Table 19 can be seen that the mean of both components is lower than 5.29, but the difference is not significant because alpha > 0.05. For both components there cannot be made a conclusion that it is difficult or easy to use the drawing or feedback component.

10.9 Interaction between background and results

The background data of the users is used to search for an interaction between education and the results. First the correlation is shown between education and the results of the questions about the current system. Then the correlation between education and the result of the questionnaire about the drawing system is shown. At the end is looked at the correlation between education and the results of the routes and the time needed to draw the routes. See Table 20 for the different education types which are compared with the results.

Table 20 background data types

Education 1	Computer experience
Education 2	Experience with programming routes on the Lely Discovery
Education 3	Auto cad experience
Education 4	Experience with drawing tools
Education 5	Diploma, MBO = 1, VWO =2, HBO = 3, BSC = 3, MSC = 4, PHD = 5

10.9.1 Interaction between education and current system

When is searched for significant results between education and the results on the questionnaire about the usability of the current system, 5 significant results are found (see Table 22).

For three questions there seems to be a correlation with experience on programming the routes on the Lely Discovery. It seems that more experience with programming routes on the Lely Discovery has a positive influence on how easy and simple it is to use the system and how quickly they can complete their work with the system. See Figure 45 for a scatter plot of the relation between education 2 and the three questions.

From the scatter plot in Figure 45 the correlation between the first question and education 2 seems not very strong. The correlation between the question how simple to use the system and if users can complete their work quickly seems to be more strong but is also not very strong. When users have more experience with programming the routes on the Lely Discovery, they seems to find it simpler to use the system and they experience that they can complete their work more quickly than other users.

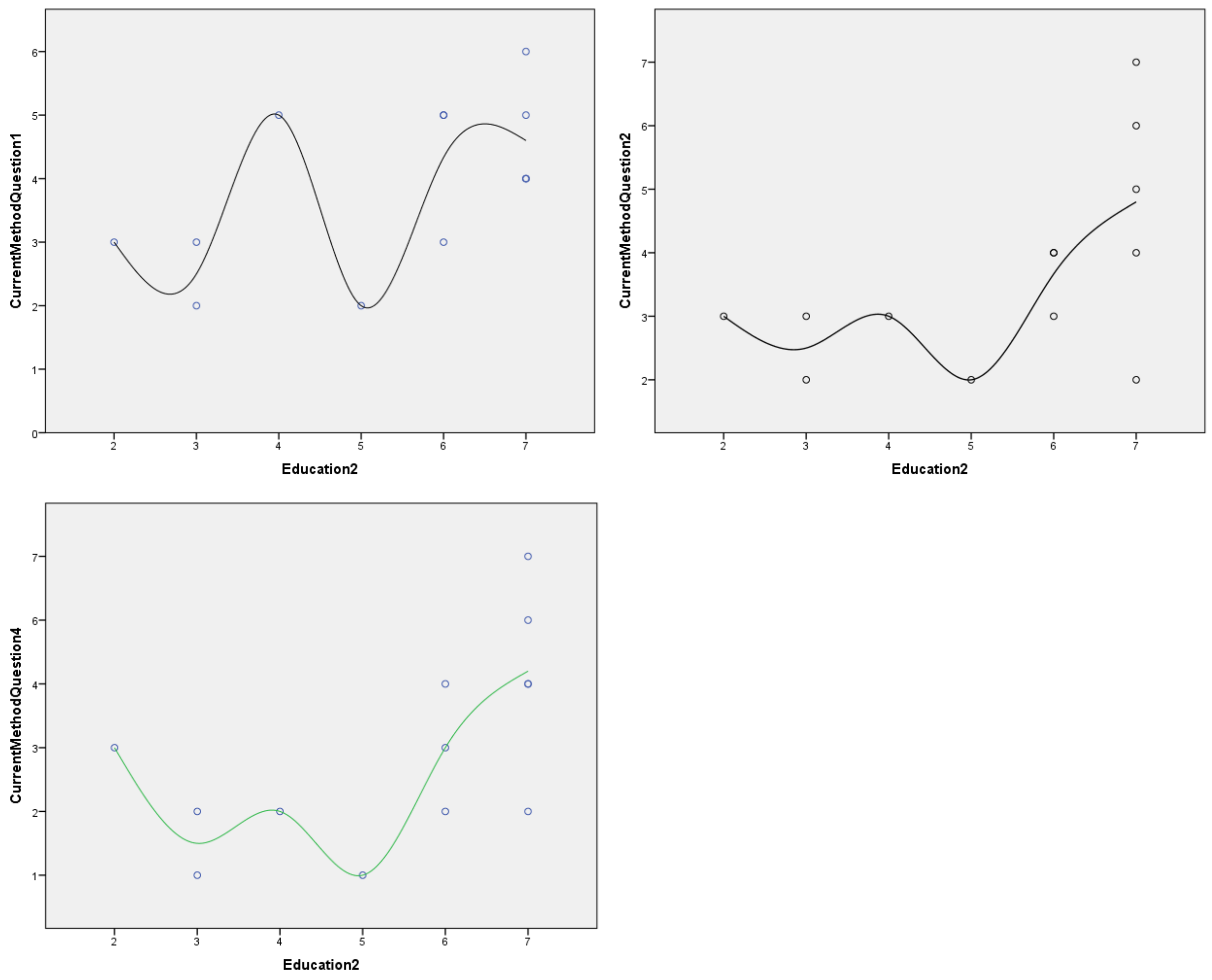


Figure 45 Interaction between experience with programming routes on the Lely Discovery and current system, a = Overall, I am satisfied with how easy it is to use this system, b = It was simple to use this system, c = I am able to complete my work quickly using this system

In Figure 46 the scatter plot is shown for interaction between the diploma received and if users feel comfortable using the current system and if they find the interface pleasant. The correlation between diploma received and both questions seems strong. In Table 28 there seems to be no relation between education 2 and 4. When users have a higher diploma they are less comfortable using the current system and find the interface less pleasant.

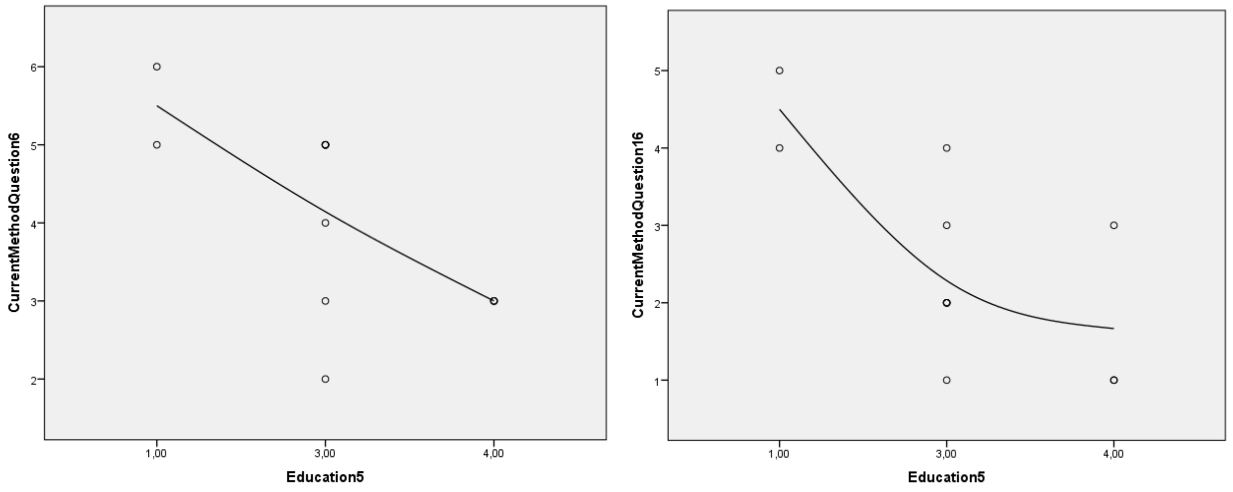


Figure 46 Interaction between diploma and current system, a = I feel comfortable using this system, b = The interface of this system is pleasant

10.9.2 Interaction between education and drawing system

When searched for significant results between education and the results on the questionnaire about the usability of the current system, 4 significant results can be found (see Table 24).

Three of the 4 significant results are between experience with the current system and the questions about the drawing system. In Figure 47 can be seen that there is a negative relation between education 2 and that it was easy to learn the system, to become productive with the system and if they expect that the system has all the functions and capabilities needed. Education 4, experience with drawing tools has a positive effect on how fast they become productive with the system.

When users have more experience with the current programming system, they find it more difficult to learn the new system and they think that they don't get quickly productive with the new system. When users have more experience with drawing tools, they think that they become more quickly productive with the new system. The new system use some features of drawing tools which makes it easier to learn for users which have experience with drawing tools.

Users with more experience with the current programming system are less positive about the functions and capabilities that the drawing system has. This can explained by the fact that they know what is necessary for a good route and don't see everything back in the route drawing system.

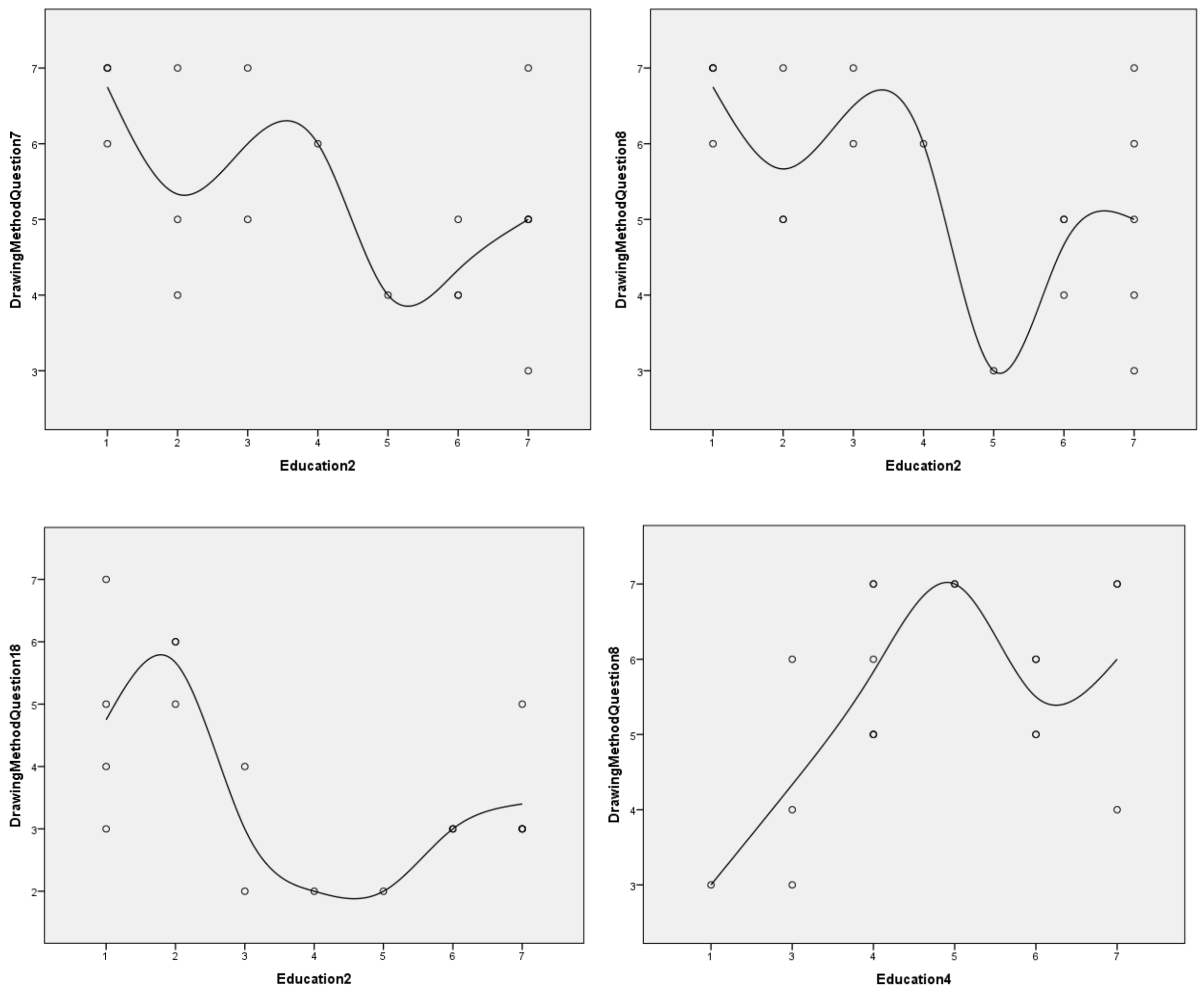


Figure 47, a = It was easy to learn to use this system, b and d = I believe I became productive quickly using this system, c = This system has all the functions and capabilities I expect it to have

10.9.3 Interaction between education, route success and draw time

For significant results between education, route success and draw time, 5 significant results can be found (see Table 26).

In Figure 48 can be seen that computer experience has a positive influence on the draw time and the route success. Also the highest diploma received has a positive influence on the draw time. When is looked to Figure 49, it can be seen that the influence of computer experience and highest diploma received has almost the same influence on the draw time for route 4. Route 4 is a combination of 2 other routes, so it is possible that users with more experience and a higher education are more easily able to reproduce route parts from earlier routes drawn.

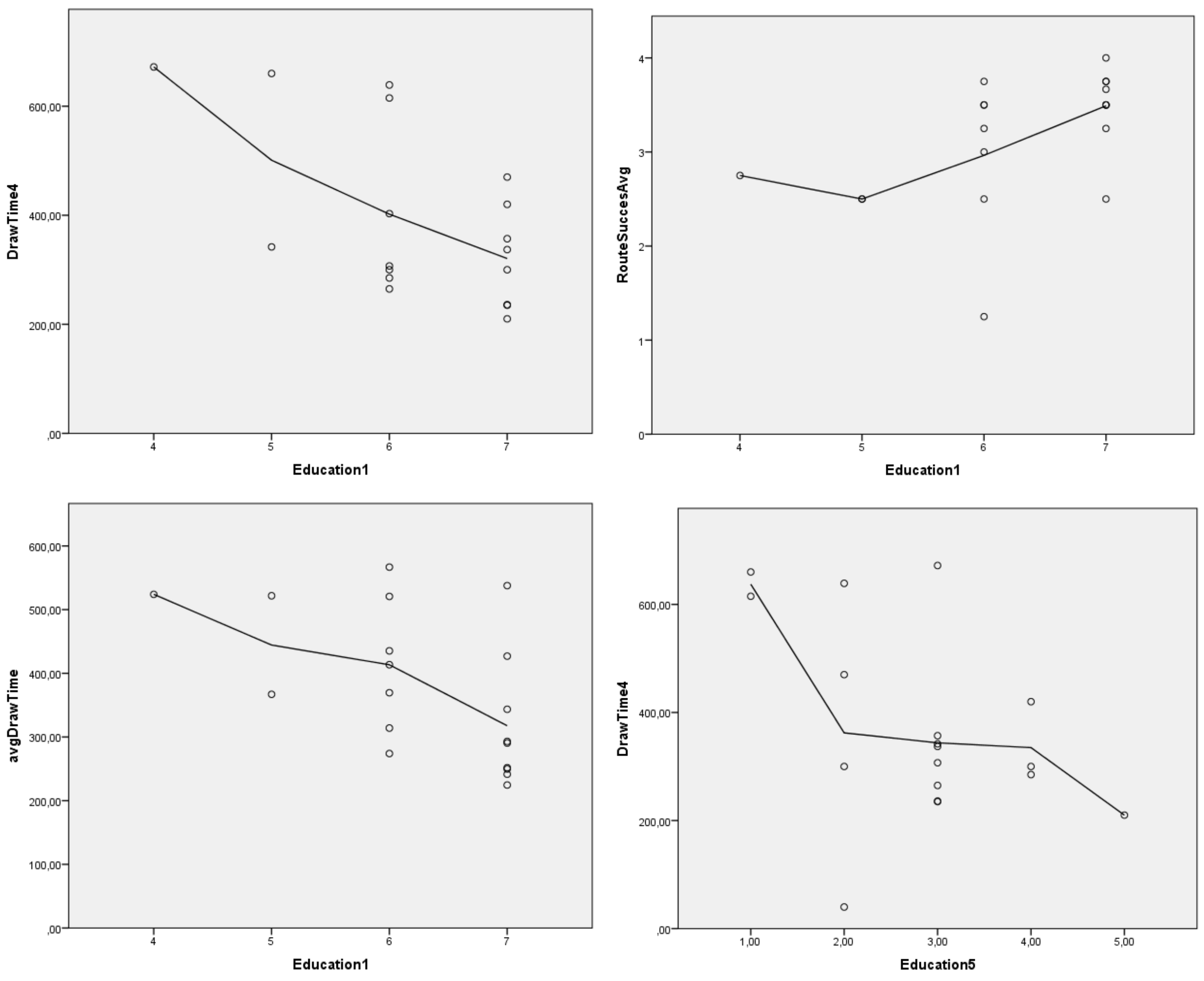


Figure 48 a-c = Interaction between computer experience (education 1) and draw time and route success. d = Interaction between draw time and experience with drawing tools.

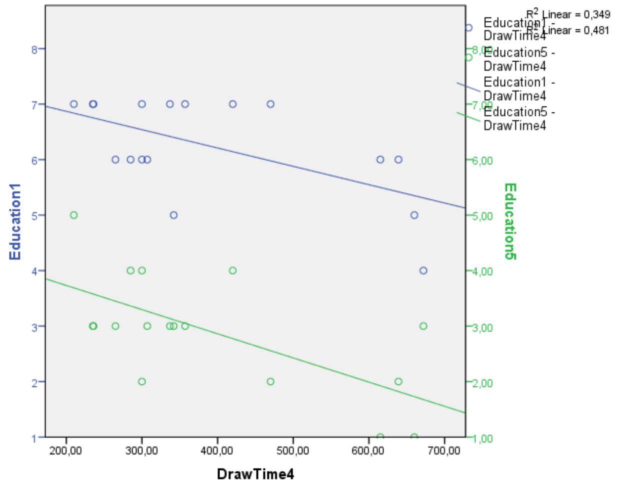


Figure 49 Interaction between draw time and computer experience (education 1) and diploma (education 5).

From this can be concluded that computer experience and the highest diploma have a positive effect on the learning curve and so the average drawing time. The computer experience of the users has also a positive effect on the success of the routes.

10.9.4 Conclusion

Some significant results are found between the background of the users and the results of the user-study. Experience with the current route programming has a positive influence on the user experience on how simple it is to use the current system and they think they can complete their work quickly with the current route programming method then less experienced users.

When users have more experience with the current route programming system, they find it more difficult to learn the new system and they think that they don't get quickly productive with the new system. They are also less positive about the functions and capabilities that the drawing system has then users with less experience with the current route programming system.

Computer experience has a positive influence on the draw time and the route success. Also the highest diploma received has a positive influence on the draw time.

11 Conclusion and recommendations

A prototype of a route drawer is shown where users can generate an action sequence for the Lely Discovery by drawing the route in the tool. The route drawing method can replace the teaching method. The main problem with the teaching method is that it is time consuming to program a route and to modify a route. It is also difficult to learn to use the teaching method.

A user study is done where 20 users drew 5 routes in the route drawing tool, 4 of the 5 routes are tested on the robot. Users where able to generate a route 2-3 times faster with the drawing method than the minimal time needed to program the routes with the teaching method. Users first drew the route on paper, they were able to draw more than 80% of the route parts on paper also in the tool. Users where able to clean the whole barn with the routes, the average area coverage for every map was > 97.5 %.

The results of the user friendliness of the drawing method and the reliability of the routes are not convincing but provides a reasonable result to work on a second prototype. With the second prototype the tool should automatically correct the simple changes and a solution should be implemented to avoid the complex changes. To see if the route problems can be reduced, the second prototype should automatically align the route parts with the walls and show users that they drew in the red area. Also a better training should be given to reduce the problems or by making a help function and instructions in the tool.

For generating the action sequence from the routes and the map, a case-based algorithm is implemented. The algorithm is divided into two parts, a part where the action sequence is generated and a part where the action sequence is evaluated and made more reliable. The robot drove the routes as calculated with the algorithm very accurately. When a second prototype is build, it is recommend to improve the cases of the second part of the algorithm and to add cases for actions that where not implemented in the first part of the algorithm.

When the suggested improvements are made, the most small route problems should be solved and the problems which will cause the robot not can finish in the charger are reduced. The robot was in 91% of the routes able to finish in the charger. When this number cannot be increased, the drawing method will be in 91% of the routes 2-3 times faster in programming the route than with the teaching method. In 9% of the cases can be expected that users need some more time to program a route.

The route drawing method can be a step towards more advanced techniques for programming the routes. More advanced techniques could be that the routes are automatically generated in the tool. More reliable action types can be made which make use of the map. Route problems and logs can be shown in the tool on the map.

12 Future work

When the route drawing method is successfully implemented, more research can be done to make the method more simple and some steps can be automated. In this chapter is first shown how new cases can be added to the system and then how the step of drawing the routes can be automated.

12.1 Adding cases

When the system is in use, it is possible that new cases are necessary for some type barns. Those cases need to be added to the system. In the next subsections is shown how new cases can be added to the system.

12.1.1 Learning from new cases

When no relevant cases are found by the algorithm, the system should learn from the new situation. The system can't learn from new cases by itself. In (Leake, 1994) they describe the case based reasoning (CBR) cycle (see Figure 50), where they retain at the end of the cycle "parts of this experience likely to be useful for future problem solving". The expert should decide if the new situation is likely to be useful for future situation, and if useful add the learned case to the general knowledge of the system. For this an module should be made, where the situation is send to the expert when the system did not find an relevant case. On this way the expert will be informed when a user defined a case that is not covered by the system.

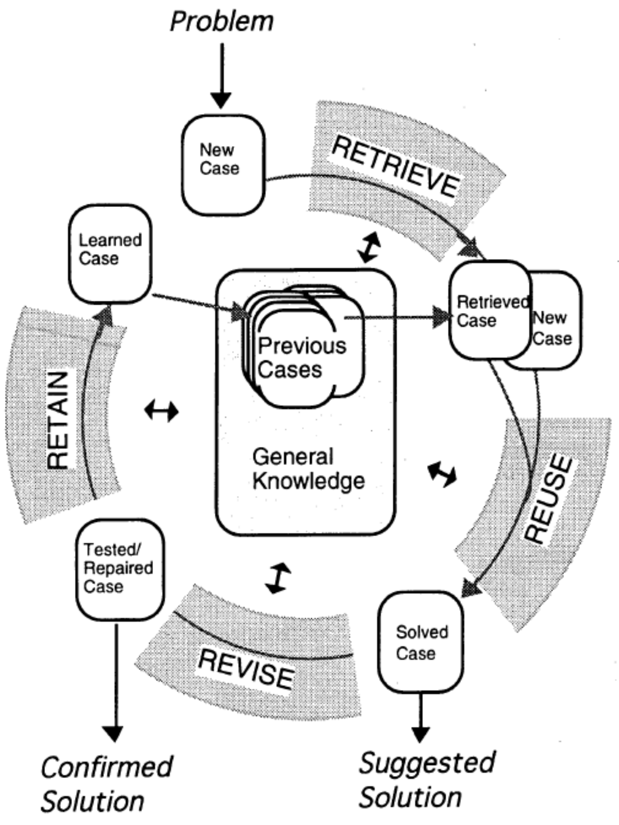


Figure 50 The CBR cycle

12.1.2 Opaque relations

When the expert decided to add a new case to the system, he should check if the new rules does not infer with the other rules. The individual production rules can be relatively simple but the logical interaction with the large set of rules may be opaque (Negnevitsky, 2005). For example you first add a case where the waypoint ends in the charger and the robot can wall-follow to the charger. The action will then be wall following until the robot is in the charger. After this you add the case wall following when a wall is on the side of the robot before the case wall follow to charger. Now in every situation that the case wall follow to charger applies, the case wall following when a wall is on the side does also apply. In every situation where the waypoints ends in the charger only the action from the other case is retrieved because that case is checked first. This is a simple example but in practice, the relations between the cases can be very opaque. This can be checked by using a test suite. The test suite should consist all relevant cases and a large set of test routes with the corresponding actions. When a new case is added, the expert should run the test suite to check if all the relevant cases and routes still return the corresponding actions. If this is not the case, there could be some opaque relation between the new rule and the old rules.

12.2 Generate automatically routes

When the route drawing tool successfully is implemented, more research could be done by applying advanced techniques like automatically generating routes in the tool. In the chapter Route planning is already mentioned some techniques for automatically planning a route. With the map and knowledge about the type routes, for example a route along feeding fence, a minimal cost function could be build. Waypoints can be placed into the tool and the cost of a route can be calculated with a cost function. When the waypoints are placed, the route actions can be automatically generated with the implemented algorithm.

When the routes can be automatically generated, users can select a generated route or make some modification to a generate route. This can save a lot of time and prevent users from making inefficient routes.

13 Bibliography

Beeson, P., MacMahon, M., Modayil, J., Murarka, A., Kuipers, B., & Stankiewicz, B. (2007, March). Integrating Multiple Representations of Spatial Knowledge for Mapping, Navigation, and Communication. *In Interaction Challenges for Intelligent Assistants*, (pp. 1-9).

Brinkman, W., Haakma, R., & Bouwhuis, D. (2009). Theoretical foundation and validity of a component-based usability questionnaire. *Behaviour and Information Technology*, 28(2), 121 - 137.

Castellanos, J. A., Tardos, J. D., & Schmidt, G. (1997, April). Building a global map of the environment of a mobile robot: The importance of correlations. *In Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on. Vol. 2*, pp. 1053-1059. IEEE.

Chronis, G., & Skubic, M. (2004, April). Robot navigation using qualitative landmark states from sketched route maps. *In Robotics and Automation. Proceedings. ICRA'04. 2004 IEEE International Conference* (pp. 1530-1535, Vol. 2). IEEE.

Davison, & Andrew, j. (2007). MonoSLAM: Real-time single camera SLAM. *Pattern Analysis and Machine Intelligence* (pp. 1052-1067). IEEE Transactions.

Dijkshoorn, N. (2012). *Simultaneous localization and mapping with the AR.Drone.* University of Amsterdam.

Discovery Team. (2014). Software requirements Discovery. Maassluis.

Fong, T., Thorpe, C., & Baur, C. (2003). Robot, asker of questions. *Robotics and Autonomous systems*, 42(3), 235-243.

Gat, E., & Dorais, G. (1994, May). Robot navigation by conditional sequencing. *In Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on* (pp. 1293-1299). IEEE.

Guo, C., & Sharlin, E. (2008, April). Exploring the use of tangible user interfaces for human-robot interaction: a comparative study. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 121-130). ACM.

H. Choset, K. M. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementation.* Boston: MIT Press.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. (pp. 100-107). IEEE Transactions on 4.2.

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2012). RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, (pp. 647-663).

Hristu-Varsakelis, D., & Andersson, S. (2002). Directed graphs and motion description languages for robot navigation. *In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Vol. 3*, pp. 2689-2694. IEEE.

Leake, D. B. (1994). Case-based reasoning. *The knowledge engineering review*, (pp. 61-64).

Lewis, J. R.-C.-7. (1995). *Computer System Usability Questionnaire*. Retrieved from http://hcibib.org/perlman/question.cgi

Maeyama, S., Ohya, A., & Yuta, S. (1996, November). Outdoor landmark map generation through human route teaching for mobile robot navigation. *In Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on. Vol. 2*, pp. 957-962. IEEE.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems.* Pearson Education.

Park, C. S., Kim, S. W., Kim, D., & Oh, S. R. (2011, November). Comparison of plane extraction performance using laser scanner and Kinect. *Ubiquitous Robots and Ambient Intelligence (URAI),* (pp. 153-155). IEEE.

Park, K., Bell, M., Kaparias, I., & Bogenberger, K. (2007). Learning user preferences of route choice behaviour for adaptive route guidance. *IET Intelligent Transport Systems*, 1(2), 159-166.

Project Tango. (n.d.). Retrieved february 28, 2014, from https://www.google.com/atap/projecttango/

Rawlinson, D., & Jarvis, R. (2008). Ways to Tell Robots Where to Go-Directing autonomous robots using topological instructions. *Robotics & Automation Magazine, IEEE*, 15(2), 27-36.

Shah, D. C., & Campbell, M. E. (2013). A qualitative path planner for robot navigation using human-provided maps. 32(13), 1517-1535.

Shah, D. C., Schneider, J., & Campbell, M. E. (2010, October). A robust sketch interface for natural robot control. *In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* (pp. 4458-4463). IEEE.

Silver, D., Bagnell, J. A., & Stentz, A. (2010). Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, 29(12), 1565-1592.

Sisbot, E. A., Marin-Urias, L. F., Alami, R., & Simeon, T. (2007). A human aware mobile robot motion planner. Robotics. *IEEE Transactions on*, 23(5), 874-883.

Skubic, M., Anderson, D., Blisard, S., Perzanowski, D., & Schultz, A. (2007). Using a hand-drawn sketch to control a team of robots. *Autonomous Robots*, 22(4), 399-410.

Skubic, M., Blisard, S., Bailey, C., Adams, J. A., & Matsakis, P. (2004). Qualitative analysis of sketched route maps: translating a sketch into linguistic descriptions. Systems, Man, and Cybernetics. *Part B: Cybernetics, IEEE Transactions on*, (pp. 34(2), 1275-1282).

Sprunk, C., Tipaldi, G. D., Cherubini, A., & Burgard, W. (2013, November). Lidar-based teach-and-repeat of mobile robot trajectories. *In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (pp. 3144-3149). IEEE.

Wang, Y., Huber, M., Papudesi, V. N., & Cook, D. J. (2003, October). User-guided reinforcement learning of robot assistive tasks for an intelligent environment. *In Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* (pp. 424-429, Vol. 1). IEEE.

Wieland, A., & Wallenburg, C. M. (2012). Dealing with supply chain risks: Linking risk management practices and strategies to performance. *International Journal of Physical Distribution & Logistics Management*, 42(10).

Yamada, S. (2004). Recognizing environments from action sequences using self-organizing maps. *Applied Soft Computing*, 4(1), 35-47.

14 Appendix A

14.1 Requirements Lely Discovery

A selection of the requirements for the Lely Discovery:

1) The user is responsible to provide a set of routes and timetable that ensures the coverage of the desired cleaning area and also provides the cleaning method in each area.

Installation

During installation of the machine, the user (service technician) shall be guided through a number of steps required to make the machine operational.

After that the user shall perform the following actions (which can be done with multiple machines in parallel):

1) Perform tests to verify that machine was not damaged during transport and correctly assembly at farm
2) Configure the machine, perform calibrations and confirm that they are correct
3) Create a map and download it into the machine
4) Program routes and confirm that they are correct
5) Fill in the timetable and put the machine in operation

Managing routes

2) The user can program a new route and modify an existing route. When programming (parts of) existing routes can be copied to allow reuse. When modifying, the user does not need to reprogram the complete route.
3) The system shall guide the user while programming a route.
4) A set of actions are presented from which the user can choose.
5) Each action has a set of parameters that the user can configure before starting.
6) The user explicitly starts the action.
7) When the action was started, the user can pause and resume.
8) Meanwhile (a part of) the map with the machine location is shown on the UI device.
9) When the action is stopped or finished the user shall explicitly accept the action.
10) The action will be saved with the learned prime parameters.
11) The user can transfer a map from the UI device to the machine and vice versa. The user can view, create and edit the map, but not necessarily on the UI device.
12) The user can make and restore a backup of the routes, map and timetable. A backup will be stored on the UI device.
13) The user can manage the timetable, i.e. create, modify and delete entries (start time for a specific route). The user can see the used capacity of the machine, i.e. how much time the

machine will spend driving and how much time it will spend charging according to the configuration in the timetable.

14) The user can manually start a route and can decide if the machine shall go to the in operation state afterwards. A route that was started by the user overrides all other tasks.
15) The user should start and end every route at the charger when programming. The machine shall give a warning when starting programming of a route when it's not at the charger or when ending programming of a route not at the charger.
16) Among the actions programmed in the route by the user, it is included the control of the cleaning.

Navigation

17) The machine shall drive a pre-programmed route accurately enough to come back to the charger.
18) In case an obstacle like a cow (leg) or bucket blocks the pre-programmed route, the machine shall try to circumvent it in order to complete the route. The machine has no way of detecting an obstacle before colliding with it, so it shall not be able to prevent such collision.
19) Inaccuracies while driving, caused by slipping wheels for example, shall be compensated by using the given map where possible. For example when the machine is driving parallel to cubicles and it passes the end of the cubicles (which is also drawn in the map) it shall correct the longitudinal position to the end of the cubicles.
20) The machine shall always try to charge the battery when connected to the charger.
21) The machine shall beep when driving autonomously. The user can configure the beep interval and even turn of the beep completely.

15 Appendix B

15.1 Usability test of route programming on the Lely Discovery

15.1.1 Introduction

Thank you for participating in this test. With this test we want to compare a new route programming method with the current route programming method for the Lely Discovery. You can skip part 1 if you don’t have experience with the current route programming method on the Lely Discovery.

15.1.2 Background

Name:
Job Title:
Highest level of education you have received:

Please rate your computer experience by placing an X in one of the seven gray cells after each statement.

			1	**2**	**3**	**4**	**5**	**6**
1.	I have experience with using computers	Strongly disagree						
2.	I have experience with programming routes on the Lely Discovery	Strongly disagree						
3.	I have experience with auto cad	Strongly disagree						
4.	I have experience with drawing tools	Strongly disagree						

15.1.3 Part 1 Current route programming with the E-Link or Smartphone

This part of the usability test is about the current route programming method. You can skip this part if you don’t have experience with the current route programming method on the Lely Discovery.

15.1.3.1 Current route programming method usability

Please rate the usability of the current route programming method on the Lely Discovery by placing an X in one of the seven gray cells after each statement. Try to respond to all the items.

	(System = current route programming method)		**1**	**2**	**3**	**4**	**5**	**6**
1.	Overall, I am satisfied with how easy it is to use this system	Strongly disagree						
2.	It was simple to use this system	Strongly disagree						
3.	I can effectively complete my work using this system	Strongly disagree						
4.	I am able to complete my work quickly using this system	Strongly disagree						
5.	I am able to efficiently complete my work using this system	Strongly disagree						
6.	I feel comfortable using this system	Strongly disagree						
7.	It was easy to learn to use this system	Strongly disagree						
8.	I believe I became productive quickly using this system	Strongly disagree						
9.	The system gives error messages that clearly tell me how to fix problems	Strongly disagree						

10.	Whenever I make a mistake using the system, I recover easily and quickly	Strongly disagree							
11.	The information (such as online help, on-screen messages, and other documentation) provided with this system is clear	Strongly disagree							
12.	It is easy to find the information I needed	Strongly disagree							
13.	The information provided for the system is easy to understand	Strongly disagree							
14.	The information is effective in helping me complete the tasks and scenarios	Strongly disagree							
15.	The organization of information on the system screens is clear	Strongly disagree							
16.	The interface of this system is pleasant	Strongly disagree							
17.	I like using the interface of this system	Strongly disagree							
18.	This system has all the functions and capabilities I expect it to have	Strongly disagree							
19.	Overall, I am satisfied with this system	Strongly disagree							

List the most **negative** aspect(s) about the current route programming method on the Lely Discovery:

1.	
2.	
3.	

List the most **positive** aspect(s) about the current route programming method on the Lely Discovery:

1.	
2.	
3.	

List the three most occurring problems that you encountered while programming a route with the current method:

1.	
2.	
3.	

Please provide any other comments on using the current route programming method:

15.1.4 Part 2 Instructions and questions about the new route drawing tool

15.1.4.1 Introduction

Below you will find step-by-step instructions on how to carry out the test. Please follow them carefully in order to ensure consistency between the procedure followed by the different testers.

15.1.4.2 Preparations

1. If you did not receive any instructions 3 days ago, please contact the experimenter
2. Make sure that you have received an pencil, ruler and eraser from the experimenter

15.1.4.3 Tasks

1. Draw with a pencil in map 1 of the attachments a route that starts and ends in the charger and covers the whole ground.
2. Try to draw the same route in the tool if possible, the map is already opened.
3. Check if the route is ok.
4. Update the route in the tool if a route part is displayed red or orange.
5. Check the route on possible problems by simulating the route with the robot in the route drawing module, update the route until the route is ok.
6. Save the route and map to xml in the route drawing module.
7. Please provide any comments about the problems you encountered while drawing the route in Table 1.
8. Repeat step 1– 7 for the other 4 maps.

Table 1 Time needed for drawing the route and comments

Map	Time	Comments about problems
1		
2		
3		
4		
5		

15.1.4.4 Route drawing module usability

Below are six statements about the route drawing module. Please rate the degree to which you disagree or agree with each statement. Please indicate your rating by placing an X in one of the seven gray cells after each statement.

			1	2	3	4	5	6
1.	Learning to operate the route drawing module would be easy for me.	Strongly disagree						
2.	I would find it easy to get the route drawing module to do what I want it to do	Strongly disagree						
3.	My interaction with the route drawing module would be clear and understandable	Strongly disagree						
4.	I would find the route drawing module to be flexible to interact with.	Strongly disagree						
5.	It would be easy for me to become skilful at using the route drawing module	Strongly disagree						

6.	I would find the route drawing module easy to use	Strongly disagree						

15.1.4.5 Feedback module usability

Below are six statements about the feedback module. Please rate the degree to which you disagree or agree with each statement. Please indicate your rating by placing an X in one of the seven gray cells after each statement.

			1	2	3	4	5	6
1.	Learning to operate the feedback module would be easy for me	Strongly disagree						
2.	I would find it easy to get the feedback module to do what I want it to do	Strongly disagree						
3.	My interaction with the feedback module would be clear and understandable	Strongly disagree						
4.	I would find the feedback module to be flexible to interact with	Strongly disagree						
5.	It would be easy for me to become skilful at using the feedback module	Strongly disagree						
6.	I would find the feedback module easy to use	Strongly disagree						

15.1.4.6 Route drawing tool usability

Please rate the usability of the route drawing tool for the Lely Discovery by placing an X in one of the seven gray cells after each statement. Try to respond to all the items.

			1	2	3	4	5	6
1.	Overall, I am satisfied with how easy it is to use this route drawing tool	Strongly disagree						
2.	It was simple to use this route drawing tool	Strongly disagree						
3.	I can effectively complete my work using this route drawing tool	Strongly disagree						
4.	I am able to complete my work quickly using this route drawing tool	Strongly disagree						
5.	I am able to efficiently complete my work using this route drawing tool	Strongly disagree						
6.	I feel comfortable using this route drawing tool	Strongly disagree						
7.	It was easy to learn to use this route drawing tool	Strongly disagree						
8.	I believe I became productive quickly using this route drawing tool	Strongly disagree						
9.	The route drawing tool gives error messages that clearly tell me how to fix problems	Strongly disagree						
10.	Whenever I make a mistake using the route drawing tool, I recover easily and quickly	Strongly disagree						
11.	The information (such as online help, on-screen messages, and other documentation) provided with this route drawing tool is clear	Strongly disagree						
12.	It is easy to find the information I needed	Strongly disagree						
13.	The information provided for the route drawing tool is easy to understand	Strongly disagree						
14.	The information is effective in helping me complete the tasks and scenarios	Strongly disagree						
15.	The organization of information on the route drawing tool screens is clear	Strongly disagree						
16.	The interface of this route drawing tool is pleasant	Strongly disagree						
17.	I like using the interface of this route drawing tool	Strongly disagree						

18.	This route drawing tool has all the functions and capabilities I expect it to have	Strongly disagree							
19.	Overall, I am satisfied with this route drawing tool	Strongly disagree							

List the most **negative** aspect(s) about the route drawing tool:

1.	
2.	
3.	

List the most **positive** aspect(s) about the route drawing tool:

1.	
2.	
3.	

List the three most occurring problems that you encountered while programming a route with the drawing tool:

1.	
2.	
3.	

Please provide any features you missed in the route drawing module.

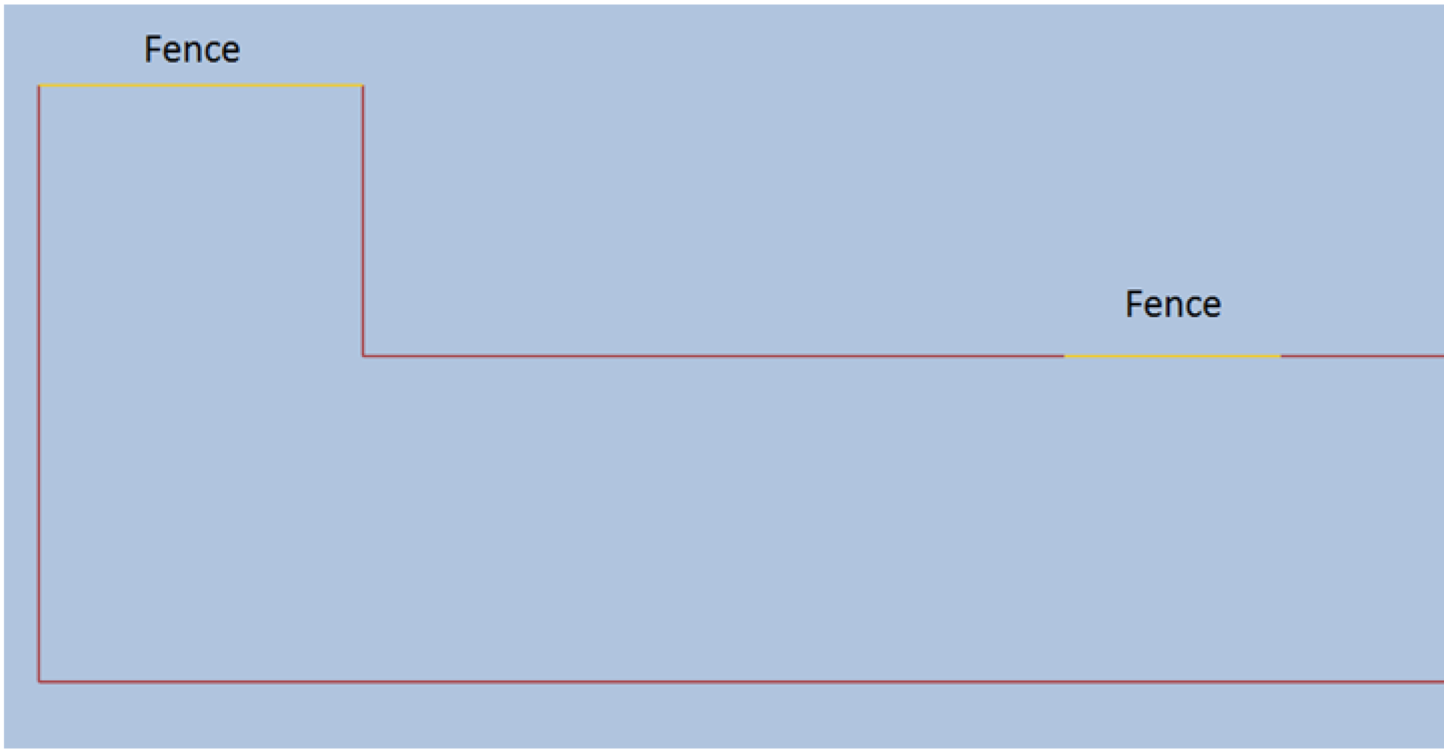
Please provide any features you missed in the feedback module.

Please provide any other comments on using the route drawing tool.

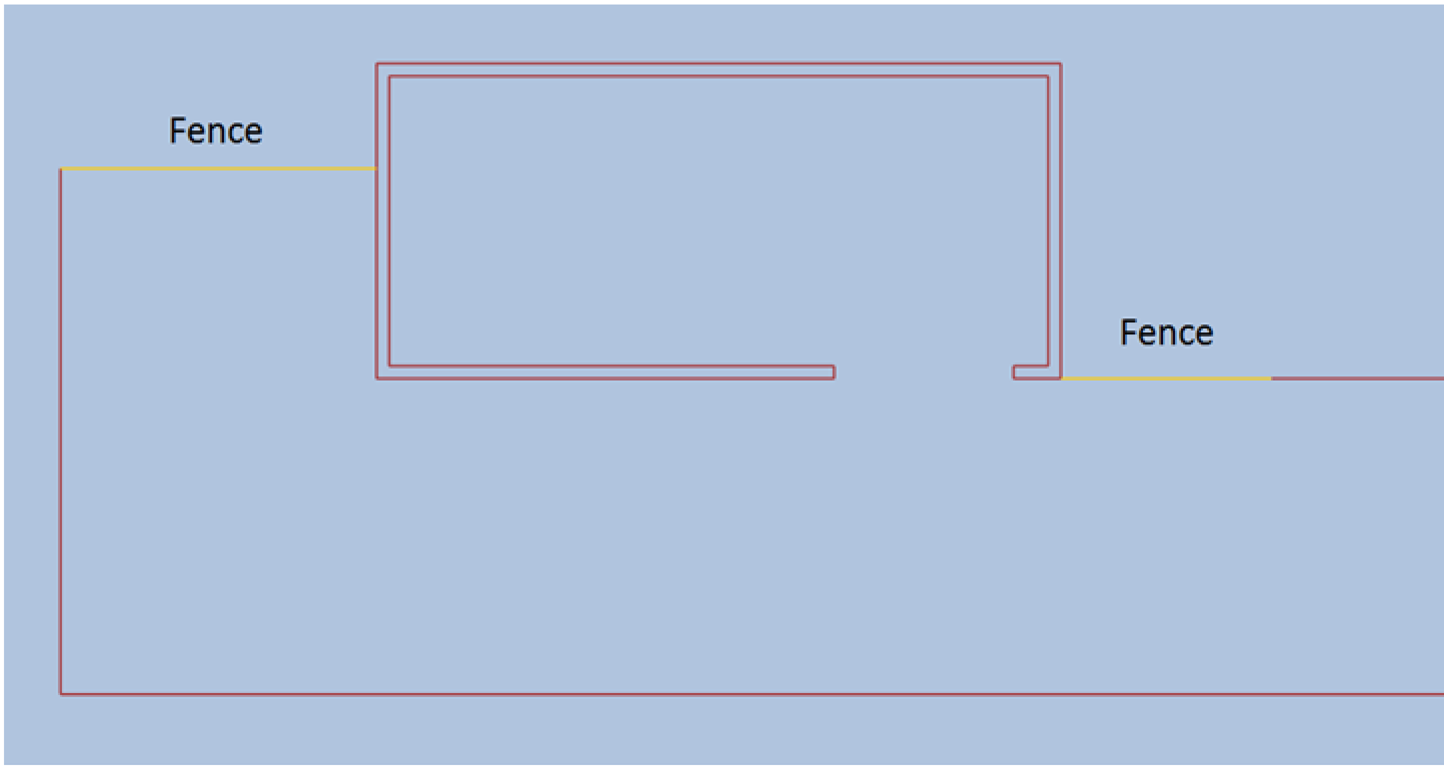
Thank you for your time and effort,

Jan Pieter Mars

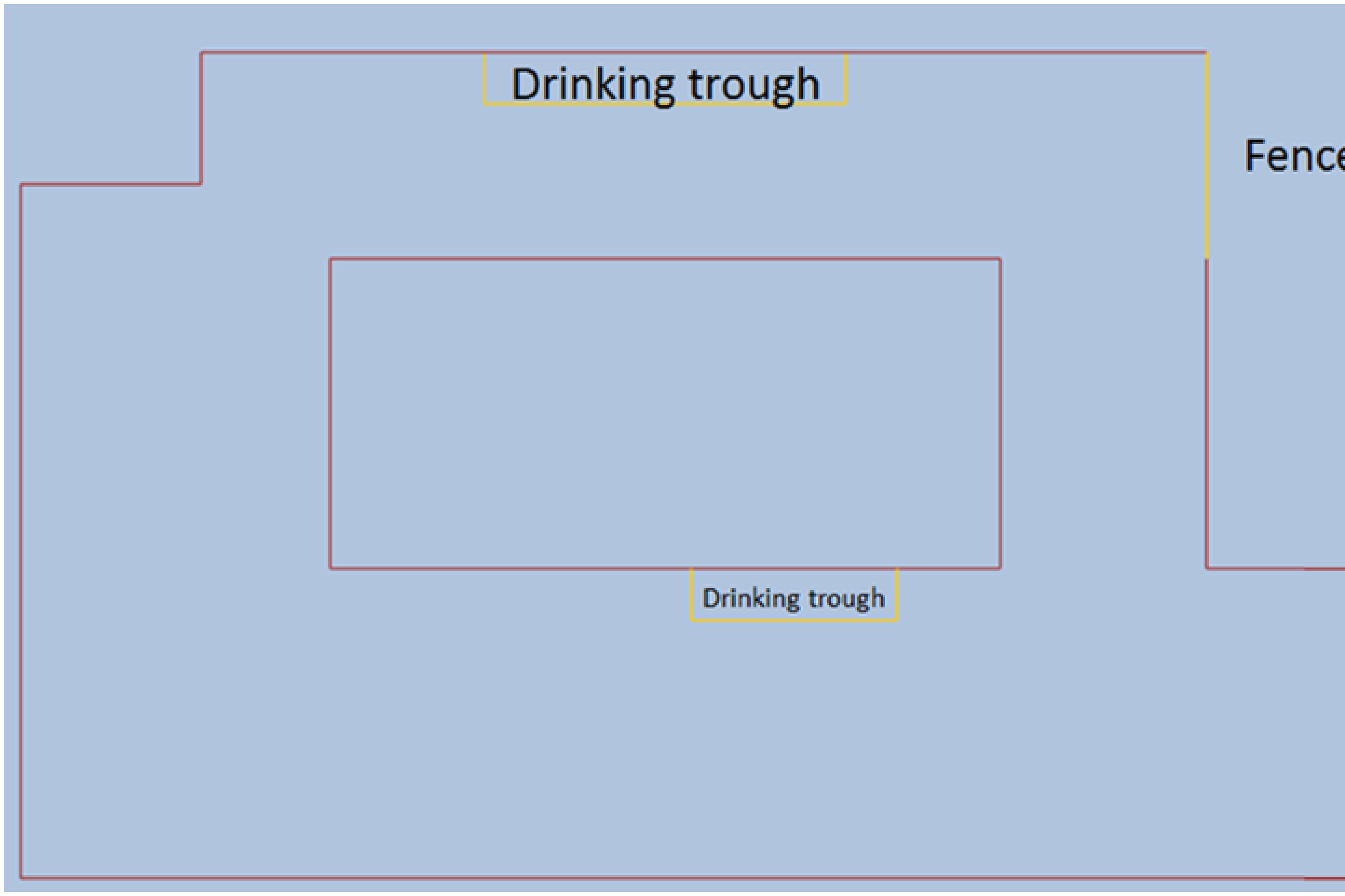
15.1.5 Map 1



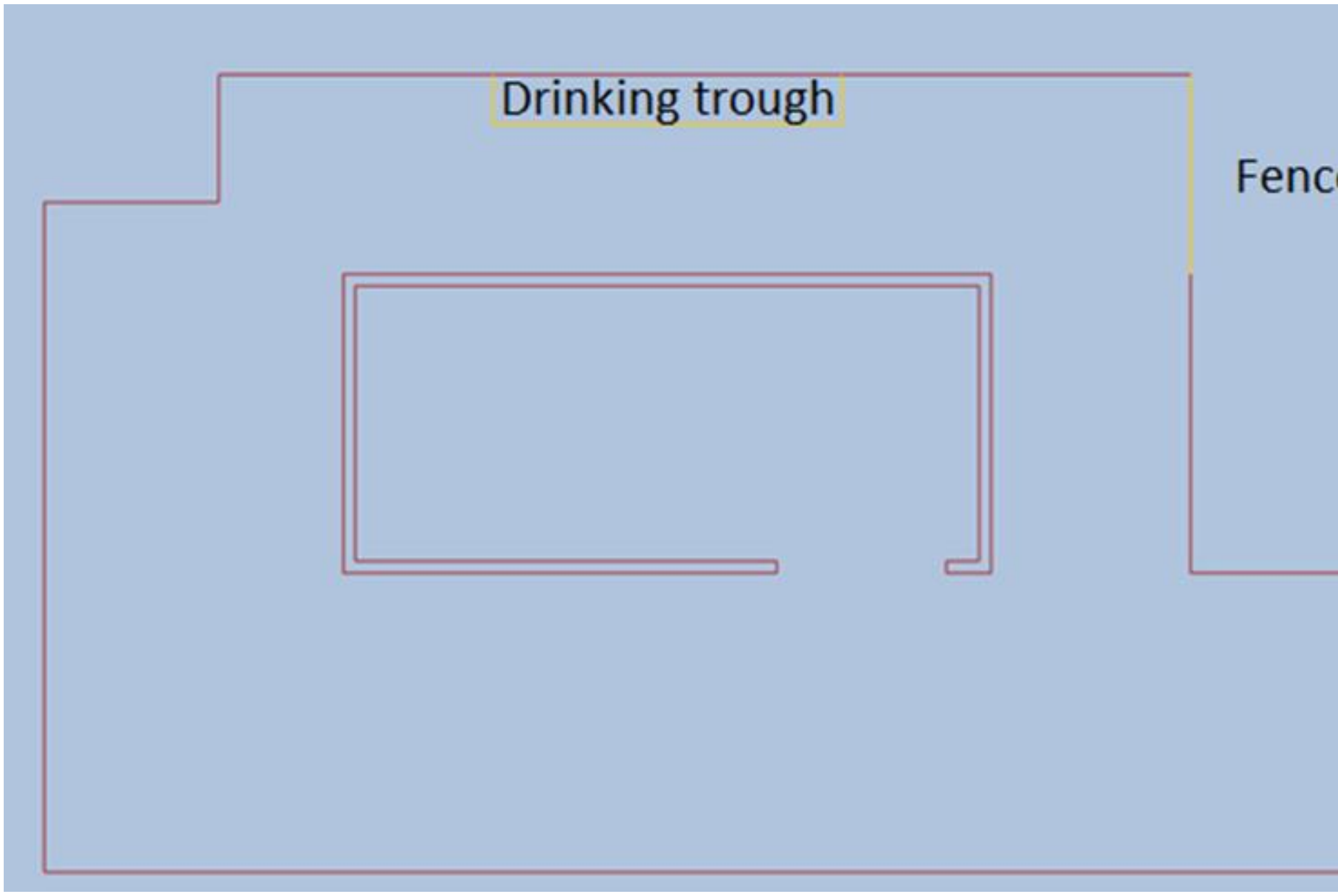
15.1.6 Map 2



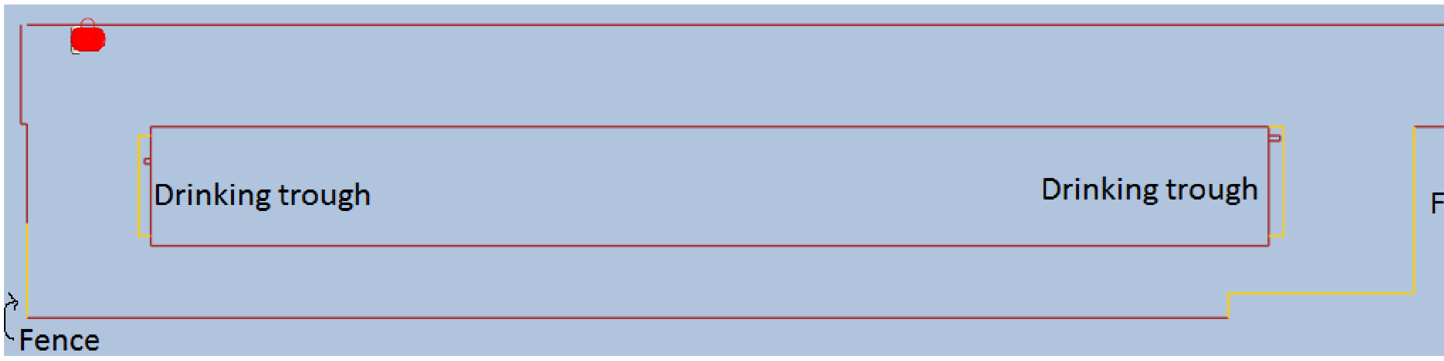
15.1.7 Map 3



15.1.8 Map 4



15.1.9 Map 5



16 Appendix C

Routes tested in the pilot:

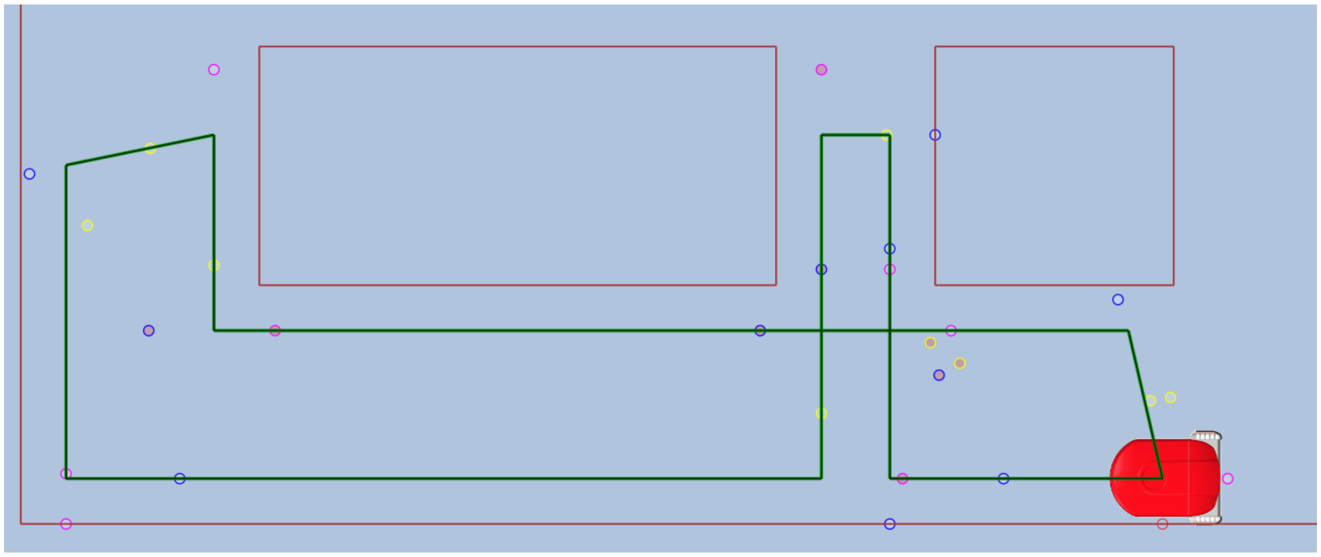


Figure 51 Test 1, normal route

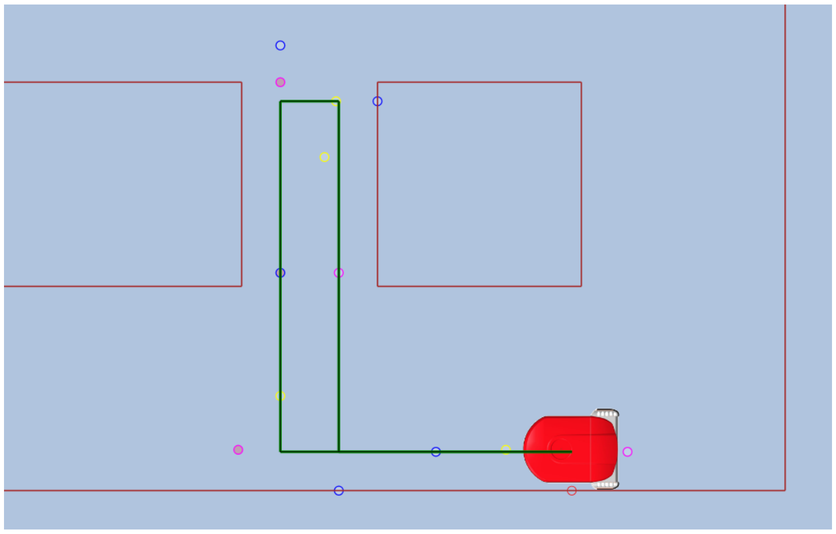


Figure 52 Test 2, turn at end of alley

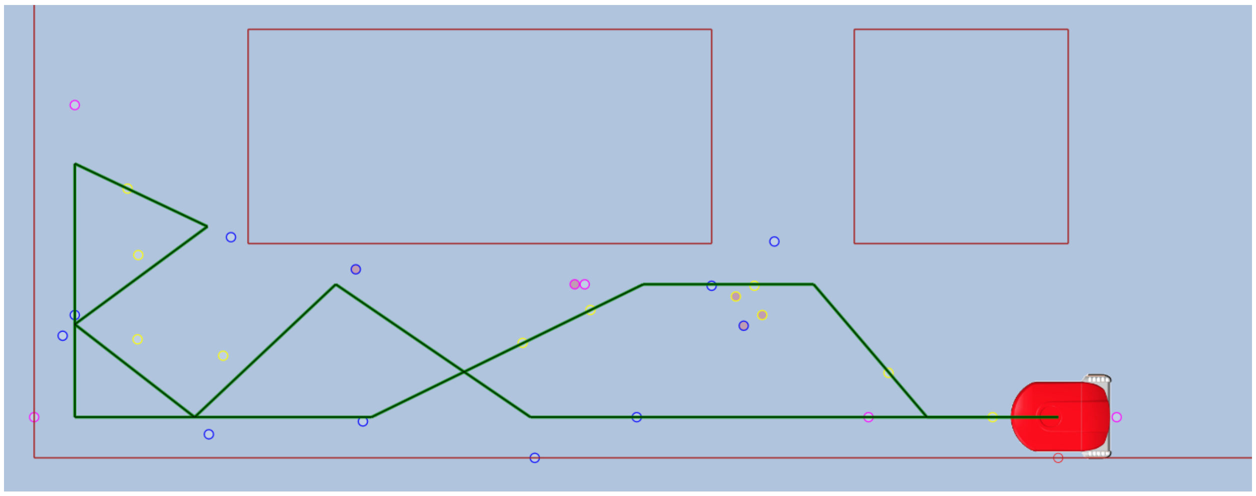


Figure 53 Test 3, bouncing with different angels on the walls

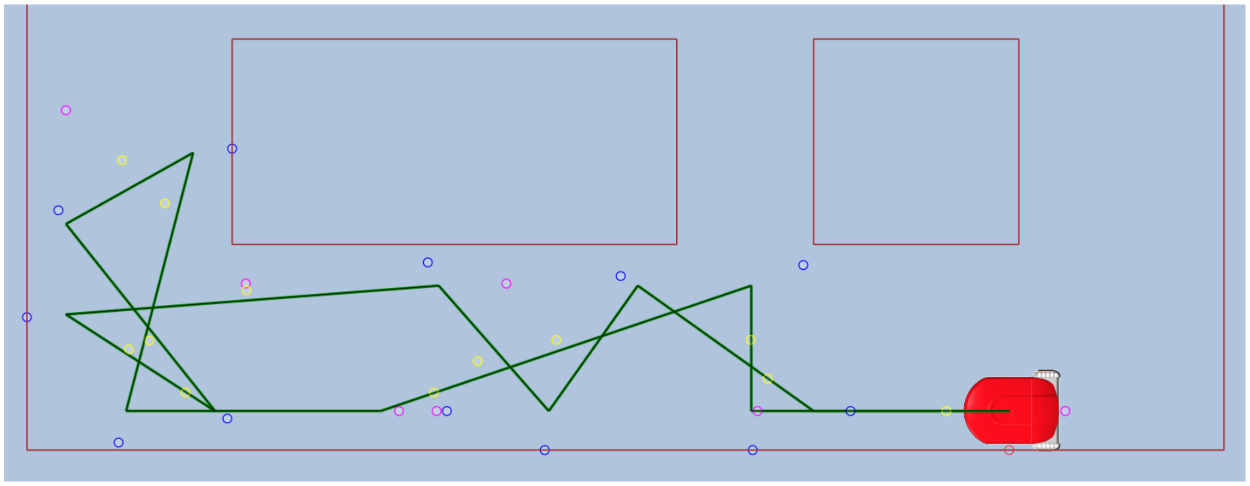


Figure 54 Test 4, bouncing with different angels on the walls

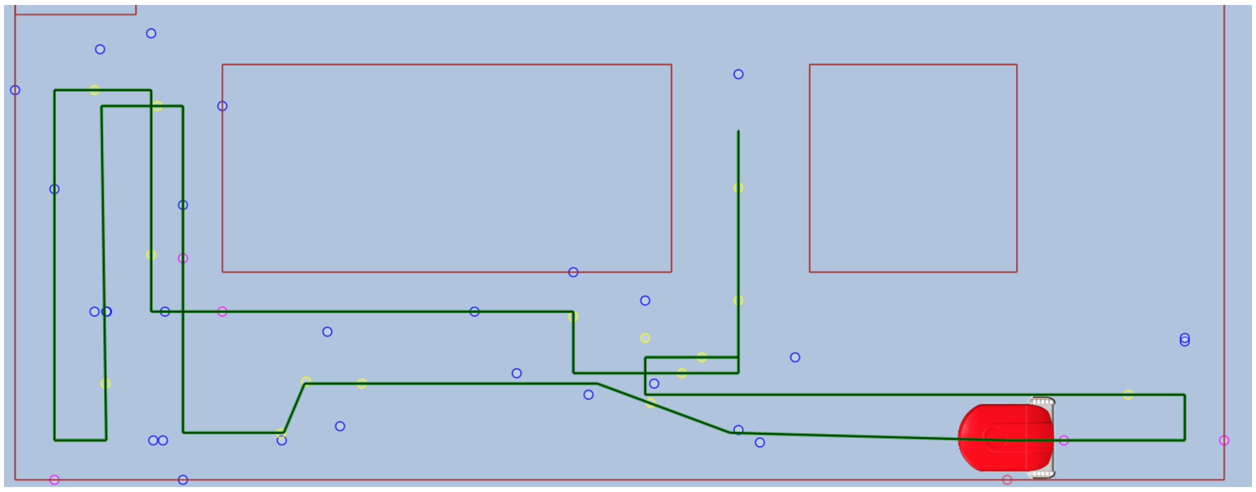
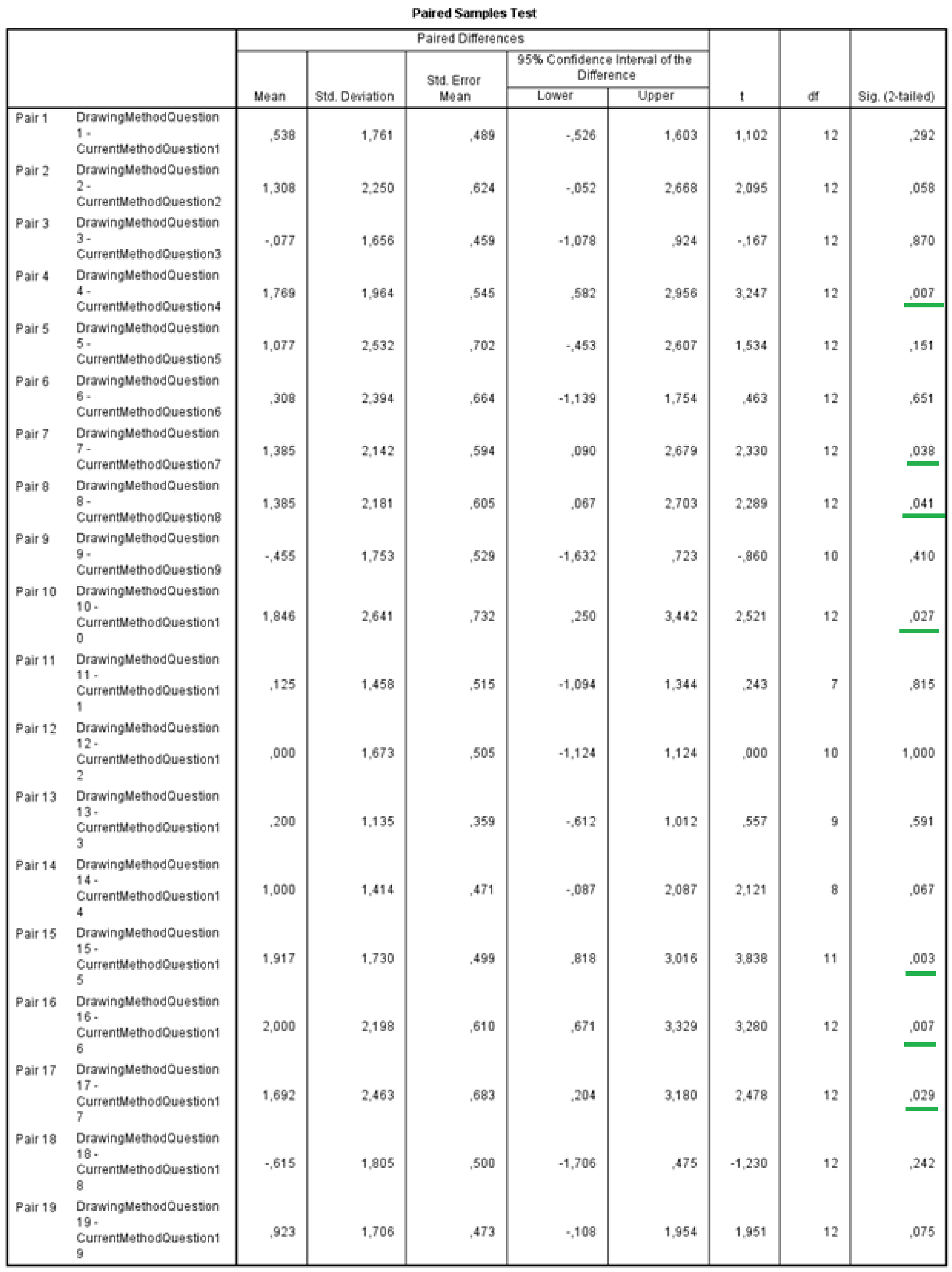


Figure 55 Test 5, route drawn by Mauro

17 Appendix D

17.1 Correlation between factors questions and results

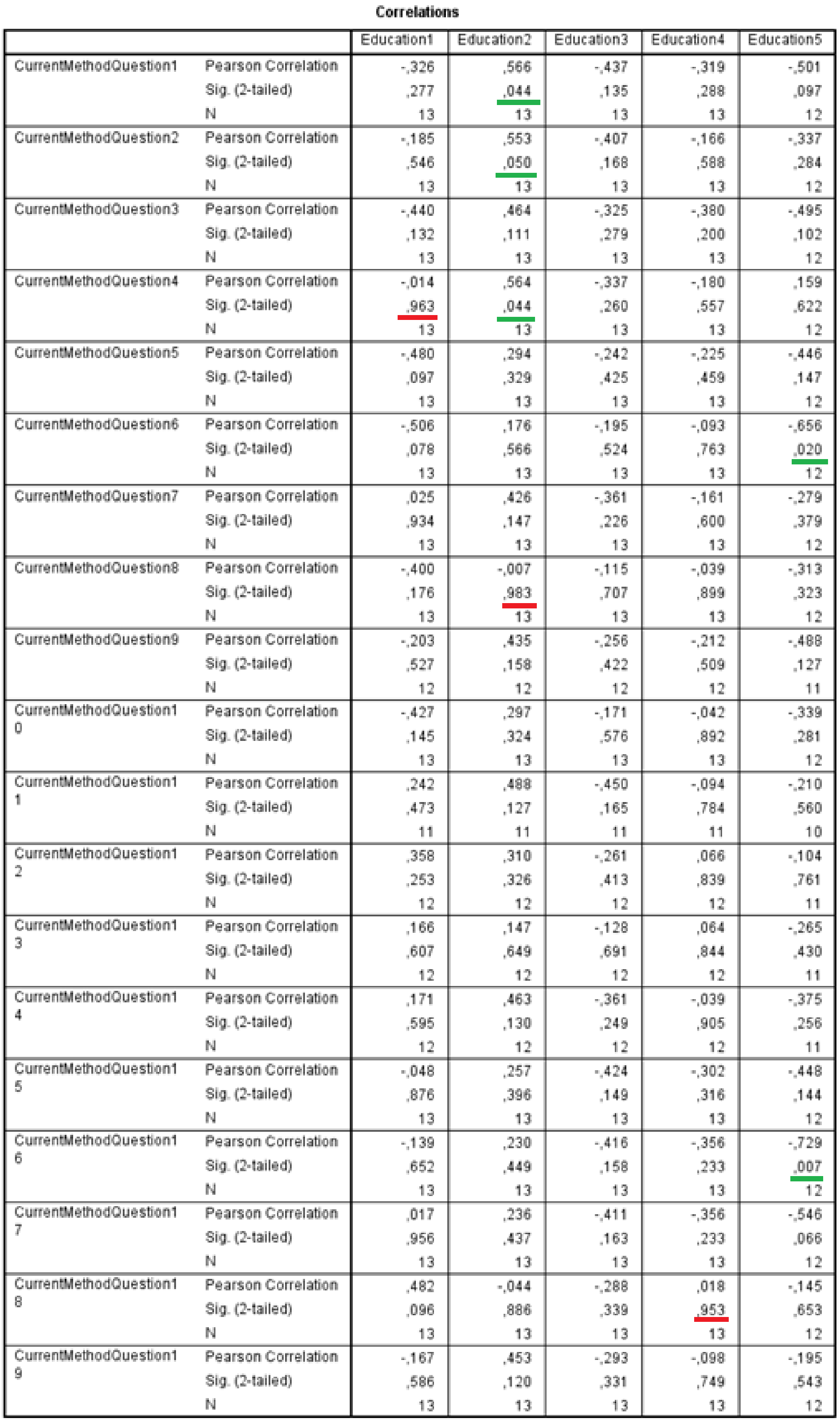
Table 21 paired samples test between current system and route drawing system



Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	DrawingMethodQuestion 1 - CurrentMethodQuestion1	,538	1,761	,489	-,526	1,603	1,102	12	,292
Pair 2	DrawingMethodQuestion 2 - CurrentMethodQuestion2	1,308	2,250	,624	-,052	2,668	2,095	12	,058
Pair 3	DrawingMethodQuestion 3 - CurrentMethodQuestion3	-,077	1,656	,459	-1,078	,924	-,167	12	,870
Pair 4	DrawingMethodQuestion 4 - CurrentMethodQuestion4	1,769	1,964	,545	,582	2,956	3,247	12	,007
Pair 5	DrawingMethodQuestion 5 - CurrentMethodQuestion5	1,077	2,532	,702	-,453	2,607	1,534	12	,151
Pair 6	DrawingMethodQuestion 6 - CurrentMethodQuestion6	,308	2,394	,664	-1,139	1,754	,463	12	,651
Pair 7	DrawingMethodQuestion 7 - CurrentMethodQuestion7	1,385	2,142	,594	,090	2,679	2,330	12	,038
Pair 8	DrawingMethodQuestion 8 - CurrentMethodQuestion8	1,385	2,181	,605	,067	2,703	2,289	12	,041
Pair 9	DrawingMethodQuestion 9 - CurrentMethodQuestion9	-,455	1,753	,529	-1,632	,723	-,860	10	,410
Pair 10	DrawingMethodQuestion 10 - CurrentMethodQuestion10	1,846	2,641	,732	,250	3,442	2,521	12	,027
Pair 11	DrawingMethodQuestion 11 - CurrentMethodQuestion11	,125	1,458	,515	-1,094	1,344	,243	7	,815
Pair 12	DrawingMethodQuestion 12 - CurrentMethodQuestion12	,000	1,673	,505	-1,124	1,124	,000	10	1,000
Pair 13	DrawingMethodQuestion 13 - CurrentMethodQuestion13	,200	1,135	,359	-,612	1,012	,557	9	,591
Pair 14	DrawingMethodQuestion 14 - CurrentMethodQuestion14	1,000	1,414	,471	-,087	2,087	2,121	8	,067
Pair 15	DrawingMethodQuestion 15 - CurrentMethodQuestion15	1,917	1,730	,499	,818	3,016	3,838	11	,003
Pair 16	DrawingMethodQuestion 16 - CurrentMethodQuestion16	2,000	2,198	,610	,671	3,329	3,280	12	,007
Pair 17	DrawingMethodQuestion 17 - CurrentMethodQuestion17	1,692	2,463	,683	,204	3,180	2,478	12	,029
Pair 18	DrawingMethodQuestion 18 - CurrentMethodQuestion18	-,615	1,805	,500	-1,706	,475	-1,230	12	,242
Pair 19	DrawingMethodQuestion 19 - CurrentMethodQuestion19	,923	1,706	,473	-,108	1,954	1,951	12	,075

Table 22



Correlations

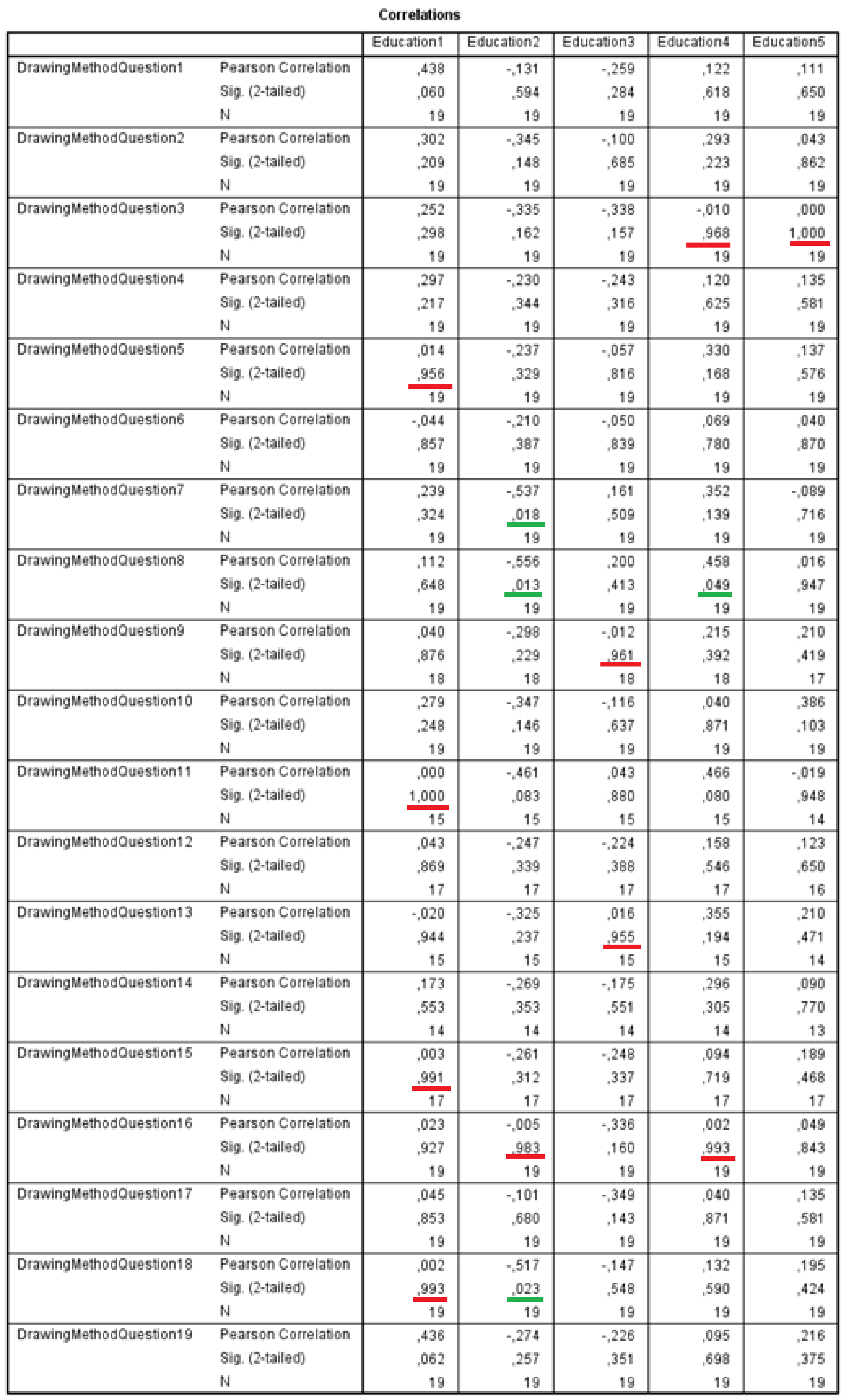
		Education1	Education2	Education3	Education4	Education5
CurrentMethodQuestion1	Pearson Correlation	-,326	,566	-,437	-,319	-,501
	Sig. (2-tailed)	,277	,044	,135	,288	,097
	N	13	13	13	13	12
CurrentMethodQuestion2	Pearson Correlation	-,185	,553	-,407	-,166	-,337
	Sig. (2-tailed)	,546	,050	,168	,588	,284
	N	13	13	13	13	12
CurrentMethodQuestion3	Pearson Correlation	-,440	,464	-,325	-,380	-,495
	Sig. (2-tailed)	,132	,111	,279	,200	,102
	N	13	13	13	13	12
CurrentMethodQuestion4	Pearson Correlation	-,014	,564	-,337	-,180	,159
	Sig. (2-tailed)	,963	,044	,260	,557	,622
	N	13	13	13	13	12
CurrentMethodQuestion5	Pearson Correlation	-,480	,294	-,242	-,225	-,446
	Sig. (2-tailed)	,097	,329	,425	,459	,147
	N	13	13	13	13	12
CurrentMethodQuestion6	Pearson Correlation	-,506	,176	-,195	-,093	-,656
	Sig. (2-tailed)	,078	,566	,524	,763	,020
	N	13	13	13	13	12
CurrentMethodQuestion7	Pearson Correlation	,025	,426	-,361	-,161	-,279
	Sig. (2-tailed)	,934	,147	,226	,600	,379
	N	13	13	13	13	12
CurrentMethodQuestion8	Pearson Correlation	-,400	-,007	-,115	-,039	-,313
	Sig. (2-tailed)	,176	,983	,707	,899	,323
	N	13	13	13	13	12
CurrentMethodQuestion9	Pearson Correlation	-,203	,435	-,256	-,212	-,488
	Sig. (2-tailed)	,527	,158	,422	,509	,127
	N	12	12	12	12	11
CurrentMethodQuestion10	Pearson Correlation	-,427	,297	-,171	-,042	-,339
	Sig. (2-tailed)	,145	,324	,576	,892	,281
	N	13	13	13	13	12
CurrentMethodQuestion11	Pearson Correlation	,242	,488	-,450	-,094	-,210
	Sig. (2-tailed)	,473	,127	,165	,784	,560
	N	11	11	11	11	10
CurrentMethodQuestion12	Pearson Correlation	,358	,310	-,261	,066	-,104
	Sig. (2-tailed)	,253	,326	,413	,839	,761
	N	12	12	12	12	11
CurrentMethodQuestion13	Pearson Correlation	,166	,147	-,128	,064	-,265
	Sig. (2-tailed)	,607	,649	,691	,844	,430
	N	12	12	12	12	11
CurrentMethodQuestion14	Pearson Correlation	,171	,463	-,361	-,039	-,375
	Sig. (2-tailed)	,595	,130	,249	,905	,256
	N	12	12	12	12	11
CurrentMethodQuestion15	Pearson Correlation	-,048	,257	-,424	-,302	-,448
	Sig. (2-tailed)	,876	,396	,149	,316	,144
	N	13	13	13	13	12
CurrentMethodQuestion16	Pearson Correlation	-,139	,230	-,416	-,356	-,729
	Sig. (2-tailed)	,652	,449	,158	,233	,007
	N	13	13	13	13	12
CurrentMethodQuestion17	Pearson Correlation	,017	,236	-,411	-,356	-,546
	Sig. (2-tailed)	,956	,437	,163	,233	,066
	N	13	13	13	13	12
CurrentMethodQuestion18	Pearson Correlation	,482	-,044	-,288	,018	-,145
	Sig. (2-tailed)	,096	,886	,339	,953	,653
	N	13	13	13	13	12
CurrentMethodQuestion19	Pearson Correlation	-,167	,453	-,293	-,098	-,195
	Sig. (2-tailed)	,586	,120	,331	,749	,543
	N	13	13	13	13	12

Table 23

Correlations

		Education1	Education2	Education3	Education4
DrawingComponent1	Pearson Correlation	,431	-,407	-,123	,131
	Sig. (2-tailed)	,065	,084	,615	,593
	N	19	19	19	19
DrawingComponent2	Pearson Correlation	,174	,025	-,382	-,147
	Sig. (2-tailed)	,476	,920	,106	,549
	N	19	19	19	19
DrawingComponent3	Pearson Correlation	,363	-,043	-,212	,064
	Sig. (2-tailed)	,127	,861	,384	,793
	N	19	19	19	19
DrawingComponent4	Pearson Correlation	-,200	-,180	-,112	,153
	Sig. (2-tailed)	,413	,460	,649	,532
	N	19	19	19	19
DrawingComponent5	Pearson Correlation	,078	-,407	,014	,319
	Sig. (2-tailed)	,751	,084	,953	,183
	N	19	19	19	19
DrawingComponent6	Pearson Correlation	,221	,079	-,387	-,059
	Sig. (2-tailed)	,364	,749	,102	,811
	N	19	19	19	19
FeedbackComponent1	Pearson Correlation	,334	-,358	-,251	,027
	Sig. (2-tailed)	,162	,132	,300	,914
	N	19	19	19	19
FeedbackComponent2	Pearson Correlation	,125	-,385	,012	,247
	Sig. (2-tailed)	,610	,104	,960	,307
	N	19	19	19	19
FeedbackComponent3	Pearson Correlation	,544	-,409	-,330	-,064
	Sig. (2-tailed)	,016	,082	,168	,793
	N	19	19	19	19
FeedbackComponent4	Pearson Correlation	-,072	-,332	-,213	-,003
	Sig. (2-tailed)	,775	,179	,395	,990
	N	18	18	18	18
FeedbackComponent5	Pearson Correlation	,090	-,503	-,181	,085
	Sig. (2-tailed)	,715	,028	,459	,728
	N	19	19	19	19
FeedbackComponent6	Pearson Correlation	,185	-,402	-,158	,103
	Sig. (2-tailed)	,449	,088	,517	,674
	N	19	19	19	19

Table 24



Correlations

		Education1	Education2	Education3	Education4	Education5
DrawingMethodQuestion1	Pearson Correlation	,438	-,131	-,259	,122	,111
	Sig. (2-tailed)	,060	,594	,284	,618	,650
	N	19	19	19	19	19
DrawingMethodQuestion2	Pearson Correlation	,302	-,345	-,100	,293	,043
	Sig. (2-tailed)	,209	,148	,685	,223	,862
	N	19	19	19	19	19
DrawingMethodQuestion3	Pearson Correlation	,252	-,335	-,338	-,010	,000
	Sig. (2-tailed)	,298	,162	,157	,968	1,000
	N	19	19	19	19	19
DrawingMethodQuestion4	Pearson Correlation	,297	-,230	-,243	,120	,135
	Sig. (2-tailed)	,217	,344	,316	,625	,581
	N	19	19	19	19	19
DrawingMethodQuestion5	Pearson Correlation	,014	-,237	-,057	,330	,137
	Sig. (2-tailed)	,956	,329	,816	,168	,576
	N	19	19	19	19	19
DrawingMethodQuestion6	Pearson Correlation	-,044	-,210	-,050	,069	,040
	Sig. (2-tailed)	,857	,387	,839	,780	,870
	N	19	19	19	19	19
DrawingMethodQuestion7	Pearson Correlation	,239	-,537	,161	,352	-,089
	Sig. (2-tailed)	,324	,018	,509	,139	,716
	N	19	19	19	19	19
DrawingMethodQuestion8	Pearson Correlation	,112	-,556	,200	,458	,016
	Sig. (2-tailed)	,648	,013	,413	,049	,947
	N	19	19	19	19	19
DrawingMethodQuestion9	Pearson Correlation	,040	-,298	-,012	,215	,210
	Sig. (2-tailed)	,876	,229	,961	,392	,419
	N	18	18	18	18	17
DrawingMethodQuestion10	Pearson Correlation	,279	-,347	-,116	,040	,386
	Sig. (2-tailed)	,248	,146	,637	,871	,103
	N	19	19	19	19	19
DrawingMethodQuestion11	Pearson Correlation	,000	-,461	,043	,466	-,019
	Sig. (2-tailed)	1,000	,083	,880	,080	,948
	N	15	15	15	15	14
DrawingMethodQuestion12	Pearson Correlation	,043	-,247	-,224	,158	,123
	Sig. (2-tailed)	,869	,339	,388	,546	,650
	N	17	17	17	17	16
DrawingMethodQuestion13	Pearson Correlation	-,020	-,325	,016	,355	,210
	Sig. (2-tailed)	,944	,237	,955	,194	,471
	N	15	15	15	15	14
DrawingMethodQuestion14	Pearson Correlation	,173	-,269	-,175	,296	,090
	Sig. (2-tailed)	,553	,353	,551	,305	,770
	N	14	14	14	14	13
DrawingMethodQuestion15	Pearson Correlation	,003	-,261	-,248	,094	,189
	Sig. (2-tailed)	,991	,312	,337	,719	,468
	N	17	17	17	17	17
DrawingMethodQuestion16	Pearson Correlation	,023	-,005	-,336	,002	,049
	Sig. (2-tailed)	,927	,983	,160	,993	,843
	N	19	19	19	19	19
DrawingMethodQuestion17	Pearson Correlation	,045	-,101	-,349	,040	,135
	Sig. (2-tailed)	,853	,680	,143	,871	,581
	N	19	19	19	19	19
DrawingMethodQuestion18	Pearson Correlation	,002	-,517	-,147	,132	,195
	Sig. (2-tailed)	,993	,023	,548	,590	,424
	N	19	19	19	19	19
DrawingMethodQuestion19	Pearson Correlation	,436	-,274	-,226	,095	,216
	Sig. (2-tailed)	,062	,257	,351	,698	,375
	N	19	19	19	19	19

Table 25

Correlations

		Education1	Education2	Education3	Education4	Education5	RouteSucces Avg	avgDrawTime	avgDrivetime
RouteSuccesAvg	Pearson Correlation	,487	-,204	,107	,282	,391	1	-,471	-,041
	Sig. (2-tailed)	,035	,402	,662	,243	,097		,036	,863
	N	19	19	19	19	19	20	20	20
avgDrawTime	Pearson Correlation	-,493	-,125	-,092	-,180	-,400	-,471	1	-,031
	Sig. (2-tailed)	,032	,609	,707	,460	,090	,036		,895
	N	19	19	19	19	19	20	20	20
avgDrivetime	Pearson Correlation	,309	,003	,231	,251	,159	-,041	-,031	1
	Sig. (2-tailed)	,198	,991	,341	,299	,515	,863	,895	
	N	19	19	19	19	19	20	20	20

Table 26

Correlations

		DrawTime1	DrawTime2	DrawTime3	DrawTime4	DrawTime5	DriveTime1	DriveTime2	DriveTime3	DriveTime4
Education1	Pearson Correlation	-,525	-,379	-,148	-,583	,079	-,119	,351	-,177	-,083
	Sig. (2-tailed)	,021	,109	,545	,011	,764	,639	,154	,469	,797
	N	19	19	19	18	17	18	18	19	12
Education2	Pearson Correlation	,085	-,034	-,307	,118	,186	,479	,119	,027	,438
	Sig. (2-tailed)	,729	,890	,201	,641	,474	,044	,638	,912	,155
	N	19	19	19	18	17	18	18	19	12
Education3	Pearson Correlation	,067	,046	-,249	,005	-,001	,561	,215	,197	,542
	Sig. (2-tailed)	,784	,851	,303	,983	,998	,016	,392	,419	,069
	N	19	19	19	18	17	18	18	19	12
Education4	Pearson Correlation	-,102	-,012	-,117	-,139	-,158	,307	,125	,195	,504
	Sig. (2-tailed)	,678	,960	,635	,582	,546	,216	,622	,425	,095
	N	19	19	19	18	17	18	18	19	12
Education5	Pearson Correlation	-,300	-,118	-,107	-,468	-,245	,030	,346	-,207	,101
	Sig. (2-tailed)	,212	,630	,661	,050	,343	,906	,146	,396	,742
	N	19	19	19	18	17	18	19	19	13

Table 27

Correlations

		DriveTime1	DriveTime2	DriveTime3	DriveTime4
DrawTime1	Pearson Correlation	-,040			
	Sig. (2-tailed)	,869			
	N	19			
DrawTime2	Pearson Correlation		-,172		
	Sig. (2-tailed)		,482		
	N		19		
DrawTime3	Pearson Correlation			,367	
	Sig. (2-tailed)			,112	
	N			20	
DrawTime4	Pearson Correlation				,563
	Sig. (2-tailed)				,045
	N				13

Table 28

Correlations

		Education1	Education2	Education3	Education4	Education5
Education1	Pearson Correlation	1	-,066	-,301	-,133	,391
	Sig. (2-tailed)		,789	,210	,586	,109
	N	19	19	19	19	18
Education2	Pearson Correlation	-,066	1	,004	-,158	,045
	Sig. (2-tailed)	,789		,988	,519	,860
	N	19	19	19	19	18
Education3	Pearson Correlation	-,301	,004	1	,803**	,107
	Sig. (2-tailed)	,210	,988		,000	,672
	N	19	19	19	19	18
Education4	Pearson Correlation	-,133	-,158	,803**	1	,232
	Sig. (2-tailed)	,586	,519	,000		,355
	N	19	19	19	19	18
Education5	Pearson Correlation	,391	,045	,107	,232	1
	Sig. (2-tailed)	,109	,860	,672	,355	
	N	18	18	18	18	19

**. Correlation is significant at the 0.01 level (2-tailed).

Table 29

Correlations

		DriveTime1	DriveTime2	DriveTime3	DriveTime4
DrawTime1	Pearson Correlation	-,040			
	Sig. (2-tailed)	,869			
	N	19			
DrawTime2	Pearson Correlation		-,172		
	Sig. (2-tailed)		,482		
	N		19		
DrawTime3	Pearson Correlation			,367	
	Sig. (2-tailed)			,112	
	N			20	
DrawTime4	Pearson Correlation				,563
	Sig. (2-tailed)				,045
	N				13

Table 30

Problems routes map 1	
1	Corner to right after second wall following

1	Last wall follow to left corner, going too far, bump on wall could not directly turn
1	Bounced on wall cubicle left, not cleaned that part
1	Corner left bottom, robot software try’s to correct but makes it worse
1	Turning to the right at cubicle
5	

Problems routes map 2	
1	Missed charger, needed local solver to get in charger
1	Getting in cubicle turn to right
7	Corner out of middle
5	Corner going in of middle
1	Drove 0.5 meter too far to get in middle
15	

Problems routes map 3	
11	Corner top left large cubicle
1	Corner top left small cubicle
8	Drinking trough bottom
2	Needed local solver to charger
1	Corner top right large cubicle
2	Corner bottom left large cubicle
3	Corner bottom left small cubicle
1	After drinking trough bottom left (bump to long on wall)
1	Corner top left small cubicle
30	

Problems routes map 4	
5	Corner out of middle
1	Getting inside of cubicle
4	Corner going in of middle
1	Missed charger needed local solver
1	corner small cubicle top left
1	Corner top left
13	

Table 31 route changes

Routes map 1

Simple changes	
3	First or last waypoint deleted
4	Wall follow to charger
1	Route part close to charger moved
2	Deleted a point
1	Shifted one point out of red area to make route green

Routes map 2

Simple changes	
2	First or last waypoint deleted
1	Deleted a point
2	Start or end in charger
1	Route part close to charger moved
1	Point in middle before going out shifted
1	Wall follow to charger
Complex changes	
1	Added point on red line to avoid wall follow bug

Routes map 3

Simple changes	
1	Turned after alley
2	First or last waypoint deleted
1	Start or end in charger
1	Fix small part drawn in red area
1	Wall follow to charger
1	Moved part bumping on charger
Complex changes	
1	Aligned line with wall
20	Corner top left
1	Fixed top left to avoid wall follow bug
1	Deleted point after corner top left
1	Deleted part where it crashed in test 1
1	Fixed part after drinking trough (bug in evaluation algorithm)
1	Fixed bug in evaluation algorithm

Routes map 4

Simple changes	
1	Moved part close to charger
1	Moved last part due to wall follow bug
Complex changes	
8	Corner top left

17.2 Answers open questions current programming method

List the most **negative** aspect(s) about the current route programming method on the Lely Discovery

4x	Route modifications are complex and takes a lot of time
3x	Cost a lot of time to make a route
3x	It’s hard to have a good insight in the barn coverage
3x	The organization of information on the system screens is not clear
2x	The organization of information on the system screens is not clear
2x	It is not easy to find the information I needed, more infor on the screen could be helpful
2x	Not clear which route options you need to use, and how they work
2x	App interface has no added value (only wireless)
2x	Difficult to perform advanced tasks
1x	Inefficient use of ui platform
1x	Slow down method is unclear
1x	The menu looks old
1x	To many clicks to perform action (not intuitive)
1x	Bugs in the software that cause the route to be discarded
1x	For good routing you need to draw the barn, draw routes and then teach routes
1x	You need good insight in how the navigation works
1x	Requires lots of knowledge about the behavior of the machine (to avoid possible problems)
1x	Can't add actions between a route
1x	Hard to find the action that made the route crash
1x	Undoing multiple actions
1x	Overseeing potential problems in an early stage

List the most **positive** aspect(s) about the current route programming method on the Lely Discovery	
6x	You see directly what you get
5x	Simple to use
3x	Reliable
2x	Basics look straight forward
1x	It works
1x	Simple interaction (few buttons)
1x	self-learning ability is quite high
1x	copy/ paste is a handy tool
1x	With just little experience, a technical interested person can easily make a reliable route
1x	I don't need to think ahead too much in details
1x	Predictable system
1x	Intuitive route programming
1x	If you know where to find it you can change a lot

List the three most occurring problems that you encountered while programming a route with the current method	
7x	No easy or/and reliable way to cancel an already programmed route action
4x	Selecting a wrong radius, difficult to set the robot back
2x	Route reliability, crashes
2x	Route lost because of a bug
1x	Information is sometimes wrong

1x	Programming with a lot of manure there is a lot of slip which cause that the distances are incorrect
1x	Lack of knowledge of possibility to use (parts) of existing route while making new route
1x	There is no clear feedback and interactive explanation towards new users
1x	Buttons not performing the action that is written on screen
1x	An action is not properly executed and after a few trials you have to start again from scratch
1x	Wrong gyro values remembered when making adjustments
1x	Small corners have big margins
1x	Adding actions after making a route

Please provide any other comments on using the current route programming method:	
1x	Route programming needs to be done by experienced technician! If customers want to do it by their own, they should have this possibility but technician should first show, demonstrate how to make the first route's to let the customer having an idea of performance and capabilities of the machine
1x	My impression is that bugs and limitations of the platform are hollowing back this programming method. In general i like the idea of programming the routes by teaching

17.3 Answers open questions route drawing method

List the most **negative** aspect(s) about the route drawing tool	
8x	Zooming and navigation
6x	Feedback is unspecific , which makes problem solving less efficient
3x	Sometimes confusing whether adding a point or moving it
2x	Distance to drinking trough, no suggestion
2x	No how to use guide/No interactive help functionality
2x	Difficult to find out what's possible/ allowed within drawing a route without test/training
1x	Switching to feedback mode to see area covered makes route making slower
1x	Colors in feedback mode are difficult to see
1x	Too few grid options
1x	Setting startpoint is difficult
1x	Needed to search for functionality of mouse and keyboard
1x	Route action with … floor no reverence
1x	selection button, in/out zoom
1x	I cannot specify the action I want to perform (turn with big radius)
1x	No feedback about accuracy of the action (maybe warnings would be nice)
1x	If there is a fault, a lot of lines become red
1x	With real machine seeing it working
1x	Error message sometimes invisible
1x	Feedback module geeft me niet direct vertrouwen of een route robust is
1x	Bij grote kaarten/routes komen er al snel te veel referentie punten, overzicht weg
1x	Snapping is not exact enough
1x	Missing some efficiency feedback like runt time and area covered double
1x	Missing undo function
1x	Soms gele lijnen die naar zijn om weg te werken

1x	Inschatten waar niet-wandvolg acties moeten koemen is vaag
1x	Overlappende lijnen zijn verwarrend
1x	Measuring distance
1x	Influence the turn or behaviour of the Lely Discovery
1x	Feedback realtime

List the most **positive** aspect(s) about the route drawing tool	
12x	Fast and efficient
7x	Interface is clear and simple, easy to use
4x	Overview of the route
3x	Easy to change the route
3x	Helps with drawing (snapping to corners etc)
1x	Learning using the route making tool is easy
1x	User friendly
1x	No more walking along with the machine for 3 hours
1x	The red marking of area where you can't drive, easy to place waypoints on the border
1x	Visual
1x	After tips easier to use
1x	Clear boundaries for robot
1x	Clear if everything is cleaned
1x	You don't need to think by yourself which actions you need
1x	Analysis of route reduce the fault risk
1x	Getting more and more experienced results in some time winning
1x	Instant feedback
1x	Relatief overzichtelijk
1x	From the "comfort" at the office

List the three most occurring problems that you encountered while programming a route with the drawing tool	
5x	Difficult to solve a problem like red lines
5x	Zooming
4x	Miss click resulting in adding points
2x	Bringing back the Lely Discovery to the charger
2x	Snapping, snapped at the wrong places
1x	Small walls
1x	Switching between route maker and feedback
1x	Estimate of distance between route-parts, sometimes you miss a small strip
1x	difficult to get straight lines
1x	Lack of warnings or tip messages to start/end a route
1x	Efficient routing
1x	Selection dots (in/ out zoom) /feedback text small
1x	How many route lines in an alley, based on width of robot
1x	Needed to add extra point to get right referent points
1x	Overlap in points not clear

1x	No real time floor coverage
1x	Moeilijk weg te werken gele lijnen, 1 rode lijn die moeilijk weg te werken was omdat de eigenlijk verkeerde lijn nog groen was
1x	Continuously questioning reliability
1x	No information of the barn

Please provide any features you missed in the route drawing module	
9x	Immediate feedback of area covered
2x	Create automatically a route
2x	Efficiency
2x	Making lines exactly horizontal or vertical
2x	Integrate drawing and feedback tool
1x	Use Lely Discovery figure instead of points
1x	Interactive grids
1x	Dump areas
1x	Action once a day
1x	If a waypoint could not be reached, add error-info which tells which part of a wall is the problem
1x	Capacity
1x	Red area constant on screen
1x	Using double click for making a point, scrolling with single click
1x	Instruction overview of key's
1x	Reference lines for parallel to wall, bumping points, end of wall (reliability of route)
1x	Drawing curves
1x	Pop-up feedback about incorrect routes
1x	Mini map. While making a route i would like to see a live feed with an indication of the coverage
1x	Runtime of the route
1x	Ctrl + z
1x	Feedback messages
1x	In de route maker een idee krijgen waar niet-wandvolg acties getekend moeten worden om aan te sluiten op eerdere acties
1x	Copy part working

Please provide any features you missed in the route feedback module	
5x	More detailed information about red lines
2x	Auto improvements suggested
2x	Simulation of the route
1x	Use a smarter way to indicate that a route is reliable
1x	Bigger feedback
1x	Warning for accuracy of actions
1x	Speed, (I don't want to wait)
1x	Clear overview of actions
1x	Information about the percentage of the covered area and length of the route (some statistics)

Please provide any other comments on using the route drawing tool	
2x	In practice you see it directly, not directly a secure feeling about the route
1x	It is a concept with lots of potential and the current implementation is good to prove the "proof of concept"
1x	With the big map I sometimes had to squint at the screen. Zoomed out I couldn’t see everything well enough and the tool was not exact enough
1x	See logs from practice directly in the tool would be nice, then you will have a good tool
1x	Small things in the barn are not visible in the map
1x	Gray area between drawing and practice

18 Appendix E

18.1 Solutions

There are a lot of possible options how a robot can navigate and finds it way. In this section some solutions are described in short for programming a route, building a map, how feedback from the robot can be used and how we could improve the current interface or which options we can use for the new interface.

18.2 Building global map

For some programming methods we need a map. In this section, some methods are described to make a map of the barn.

Handmade
A map from the barn can be made by hand. To make a map, all distances between walls needs to be measured, which is time consuming. Hand-made measurements are not very accurate which will result in a map which is not very accurate.

Augmented reality
A tablet with an camera can be used to make a 3d model of the environment (Project Tango, n.d.). This 3d model can be used as a global map. The user can see the camera feed on the tablet with an overlay or just only the 3d model while walking around. By walking around with the tablet we can make a map.

Drones
In the barn one or two drones can be launched which autonomously tries to make a complete picture from the barn. According to (Dijkshoorn, 2012) the main bottleneck for indoor drones is reliable indoor positioning. So for today this solution is not reliable enough to use.

Driving
A map can also be made by driving around with the robot on every path once. To drive the robot around costs some time but this can result in an accurate map. See building a local map for which sensors we can use.

Driving Autonomous
The robot can explore the map by driving around (Yamada, 2004). This will cost the user almost no time, but the available sensors on the robot are not accurate enough to make an accurate map. If we use temporal sensors on the robot, see building local map, we have the problem that cows can make the sensors dirty necessary for exploring the barn. The user can move the cows but then we have an safety issue, because the user doesn’t know which way the robot goes.

18.3 Building local map

For some methods to build a global map, we first need to explore local parts of the barn, where after the local parts are merged to one global map. In this section, some methods are described on which way a robot can make a local map around his position.

Ultra sound

With the ultra sound sensors, walls within 2 meters can be measured. From the data, a map can be made, but ultra sound sensors are not very precise and contains a lot of noise which cause they are not usable for making a map.

-censored-

18.4 Route programming

Before the robot can drive around automatically, the preferred route needs to be programmed. In this section, some methods to program the route for the robot in the barn are described.

Sketch the route

The route can be sketched on a tablet (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004). Before we can sketch the route, a map of the barn is needed. From the sketch and the map, landmarks and waypoints are extracted where after the robot tries to drive the sketched route. This is a fast and cheap way to program the route. The problem is that it is not accurate and not robust to errors in the sketch.

Description

1) Sketch place and direction of the robot on the map
2) Sketch route which needs to be followed
3) Derive waypoints/ description from sketch (Skubic, Blisard, Bailey, Adams, & Matsakis, 2004)
4) Robot drive route and convert description/ waypoints to actions
    a. Drive indicated direction
    b. If the description match the current observation, execute new action which follows from the description
    c. Save all actions and count wheel turns, distance to wall x.
    d. Connect every action with a description/ waypoint

5) From the sketch you can click on a part of the description and go the corresponding action

Pros

- When humans don't have a detailed map, they make also a sketch of how to drive which don't need to be very precise
- A sketch is a fast way of describing the barn and the route which can result in a fast installation
- A map is more intuitive than a sequence of actions
- It is more intuitive to show preferences on a map than with a menu

Cons

- Cows can block the route
- Sketch can be wrong
- Difficult to follow the description/waypoints with the current sensors
- Different corners are difficult to sketch

Sketch the route with straight lines

The same idea but there can be sketched with straight lines. This is less user-friendly because the user has less freedom to draw lines. But because of the straight lines, distances between the (straight) walls are more accurate which result in a more robust behavior of the robot.

Autonomous path finding
With autonomous path finding the robot can find its own route (Silver, Bagnell, & Stentz, 2010). To make this possible we need an accurate map and the robot needs to get a better understanding of its environment which will result in a high investment necessary to make this possible. If the farmer has some route preferences, it should still be possible to add those preferences.

Pros

- User does not need to program the route

Cons

- Difficult to add preferences if there are preferences
- Very accurate map needed

Select and change automatic generated paths
The robot can generate automatically paths if a map is available (Park, Bell, Kaparias, & Bogenberger, 2007). From the generated paths the user can select the preferred path. To generate robust and accurate paths, an accurate map is necessary.

Pros

- User does not need to program the route

Cons

- Very accurate map needed
- Need to select a path
- Difficult to update a path with preferences if there are preferences

18.5 Interface

In this section some options are described which can be used for the interface with the current route programming method.

Action suggestions
During the route programming, the system can give suggestion for the next action, based on actions already done, other routes, sensor data and the map if available. Two or three action suggestions are displayed, which the user directly can select. Those suggestions can be calculated with a cost function (Silver, Bagnell, & Stentz, 2010) which for example depends on previous chosen actions and wall/map knowledge from ultrasound sensors. This can save the user a lot of time, because he don't need to search through the menu for the right action. The problem can be that other actions which are better than the suggested actions, like specific curves, are not considered by the user.

Sketch commands
Instead of searching for an action in a menu, actions can be found by making a small sketch as command (Shah, Schneider, & Campbell, 2010, October). From this sketch one action can be derived or more actions

can be shown on the suggestion menu. Sketching and accept or select an action from the suggestion menu if faster than searching for the right action in the menu.

Speech recognition
The user can ask with speech for an action, the system selects the most relevant actions which are displayed in the suggestion grid. This can be very fast if the speech is directly recognized and correctly interpreted.

18.6 Feedback

To make routes more robust and accurate, we can let the robot give feedback to the user of the programmed route. This will help the user to evaluate and to improve the programmed routes.

Feedback during programming
During the programming of the routes, the robot could evaluate the actions and give some feedback about the accuracy of the actions. On this way the user can learn from the robot too increase the reliability of the routes.

Ask questions via smartphone
If the robot gets lost or if he wants to know more about the environment to make the route more robust, he can ask questions (Fong, Thorpe, & Baur, 2003). If the robot gets lost, he can make a snapshot of the environment and sends the previous actions done. The user can give feedback to the robot with the smartphone. Also during the route programming, the robot could ask for example some questions about the correctness of some actions.

Description
- Send latest actions done, expected location and sensor information
- Turn 360 degrees for more detailed information about the walls
- User can give instructions where after the robot can try to get back on track
- Or user knows that there is a problem and can go to the barn
- If user can identify problem, user can update action sequence

Pros
- Problems with robot are identified fast
- Help robot on distance
- Increase reliable of the route when asked questions during the rout programming

Cons
- Difficult to give correct instructions
- User can get annoyed when he gets the same questions about the route many times

18.7 Solution comparison

In Table 32 the solutions are compared on the following points:

Accuracy
Accuracy is a measure on how close the result or a repeated measurement is, compared with the true-value or other measurements.

Robustness
Robustness is defined as "the ability of a system to resist change without adapting its initial stable configuration” (Wieland & Wallenburg, 2012).

Installation cost
The price is a measure of the cost for every installation, which contains time and extra cost for products needed for every installation/ route update.

Investment
Investment is defined by the total cost for developing new techniques or products and the procurement costs for the necessary items.

User-friendly
User-friendly is the ease of use and learnability of a human-made object. This can for example be how easy it is to use a specific programming method or item, or how easy it is to read a specific map.

Feasibility
Feasibility is an evaluation method on how great the change is that a new method or technique will be in production in a specific time-frame, and that the knowledge is available to develop the new method. In this case for the time-frame, one year is available to develop the new method or technique. This time frame is for projects for the short-term, for this reason project for the long-term have a more negative feasibility.

In Figure 56 a decision tree is displayed where you can see the decisions that needs to be made. First we need to decide which route programming technique we will use or explore. When we have chosen a programming technique and if the technique requires a map we need to choose how we can make a map. If we make a global map by driving the route, the robot needs to make a local map of the environment around him. The local map of every place will then be merged to a global map, and after this we can make an decision on which device we need. This device is needed for the user to program the route or to get feedback from the robot. After this we can make a decision on what new functionality we need in the interface, which navigations sensors we need for the robot to navigate the programmed routes, and if we can use a sort of feedback technique.

Table 32 Solution comparison

	Accuracy	Robustness	Price	Investment	User-friendly	Feasibility	Comments
Building global map							
Handmade	--	++	-	+	+	++	Everything needs to be measured by hand
Driving	+	+	+/-	+	+	+	Need first to drive around
Virtual reality	+/-	-	+	-	+	-	Dynamic objects in environment (cows)
Building local map							
-Censored-							
-Censored-							
Route programming							
Sketch the route	-	-	+	-	++	+/-	
Sketch with straight lines	+/-	+/-	+	-	+	+/-	Cows can make metal stripes dirty
Autonomous path finding	+/-	-	++	--	++	--	
Path generator	+/-	+/-	+	-	++	-	
Interface improvements							
Action suggestions	+/-	+/-	+	-	+	++	
Sketch commands	+	+	+	-	+/-	+	
Speech recognition	+/-	+/-	+	--	+	+/-	
Feedback							
Feedback during programming	+	+	++	+	+/-	+	
Ask questions via smartphone	+	+	++	-	+	+	

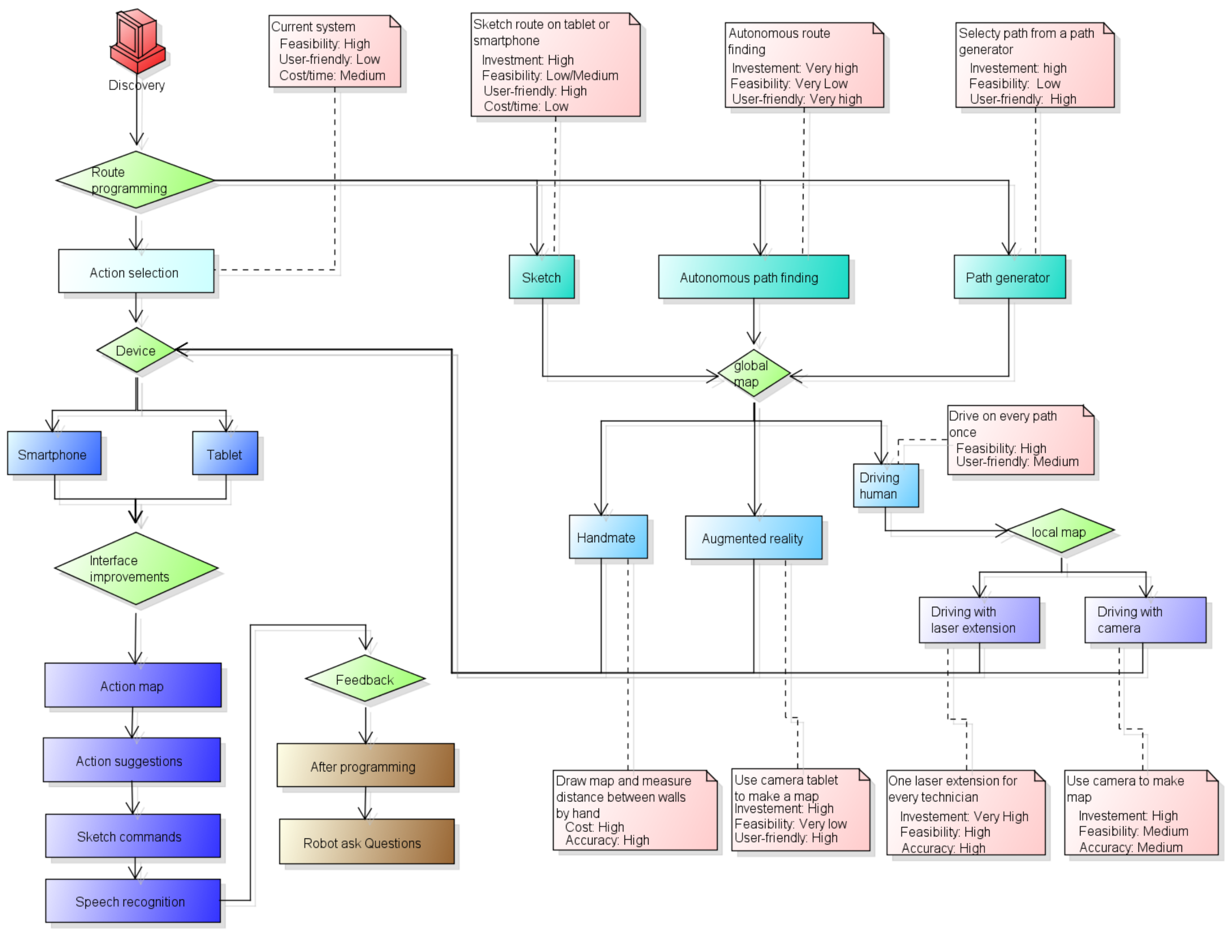


Figure 56 Decision tree with possible solutions

18.8 Conclusion and decision

In this section the possible solutions are discussed which can give an answer to the research question "How can we improve the user friendliness and effectiveness of the route teaching for the Lely Discovery platform?". From the possible solutions and the field research done, a decision is made.

Route programming

From Table 32 we can see that the current route teaching method where you must select an action from a menu has the highest feasibility and we don't need to invest a lot of money to improve this method. But the current method has a high price for every installation and is not very user-friendly. The two reason to choose for a method with a map is the low installation cost because of the time-saving during route-programming and that the user-friendliness will be much higher. So a solution which use a map will be a good option. There are different options on how we can make a map where from which we need to choose one option if we will use a map. From Table 32 we find that driving autonomous the route, to make a map and to make route options will be the best solution, because of the very low installation price and the high user friendliness. Driving autonomous is only not reachable on this moment, because of the high investment and the time needed to develop this new technique, so we can't use this technique on this moment. Virtual reality could be a nice alternative but the current technique is not that well developed to make accurate and robust maps so we also can't use this technique on this moment. Drones are still to futuristic and should be further developed before they are reliable enough to use in barns, so this is also not usable on this moment. -Censored- Sketching the route and the map is not accurate enough which will result in unreliable routes, so we can't use this technique directly. Drawing the map with a "Computer Aided Design" (CAD) program and also drawing the routes with a Computer Aided Route Drawing (CARD) program can be a good alternative. The technique to draw maps with a CAD program is available and the installation price is better than the current method.

Current method

From the field research we can conclude that the current route programming method works well for the technician. He needs to work with it every week so he can program the routes in an acceptable time. The user-friendliness of the current method is not very high, but this is not a problem for the technician because he needs to work with it every week. Some Lely-centrums let the farmer program the routes so that they can sell the robots for a cheaper price. But this can give the farmer a lot of problems, because it is too difficult for him to make good routes. Those bad programmed routes result that the robot gets stuck where after the farmer needs to bring the robot back to the charger by hand. This will decrease the trust in the robot by the farmer and the cost can be high if the technician needs to come back to reprogram the routes made by the farmer. So another route programming technique can be valuable to the farmer, and can decrease the price of the robot if the farmer is able to program the routes by himself.

Decision

We need to make a decision if we will improve the current method or that we choose for a programming technique which makes use of a map. If we improve the current method we can increase the user-friendliness for both the farmer and the technician. For the technician this will not result in a lower installation cost. For the farmer it can be possible that a more user-friendly design of the current method will help him to understand how to program the routes like the technician does. This can result in more reliable routes by the farmer. If we choose for a route programming technique with a map, a CARD program will be the best option for now. It is only the question off the installation cost will be lower when the technician needs to program the routes. This because all the distances between the walls needs to be measured to make a map from the barn. But the new robot has the option to add a map to the robot to make the robot more reliable. So if a map is added, the installation cost can be lower when the routes are programmed with a CARD program. If the farmer is able to program good routes, the installation cost can be much lower. Also the user-friendliness will be much higher if the routes can be programmed with a CAD like program on the laptop.

Computer Aided Route Drawing (CARD)
Based on the solutions found and the field research and the fact that the new robot has the option to add a map, we will research if we can improve the user-friendliness and the installation-cost when the user program the routes with a CARD program. For this we need to build a prototype where the user is able to make a map and draw the routes on the laptop. With this program the user can export the result to an xml-file which can be exported to the robot. To make an xml-file with the right route actions and a route that is reliable and robust, we need to develop an algorithm that is able to translate the map and the drawn route to the right actions. When the prototype is build and if we are able to transform a map and a drawn route to reliable route actions, we can do a user study on the available test-farms. From the user study we need to find an answer to the question whether the user is able to make reliable routes in an acceptable time and on a satisfying way with this new method.

18.9 Research computer aided route design method

In this section is described how the new route programming method should work and which research is necessary.

18.10 Route programming with CARD tool

With the new route programming method, the user should be able to make a map and to draw the desired routes on the map. When the routes are drawn on the map, the program should be able to export the map and for every route an action sequence to the robot. When the user has measured all the distances between the walls in the barn, the user can make the map and the routes on his computer outside the barn.

18.11 Algorithm to build the action sequence

From the map and the drawn routes an action sequence should be automatically made. This can be done with an algorithm which calculate the most accurate and robust route. The algorithm should try to follow the drawn lines and to find the most robust actions available on a specific place in every situation. The selected action can for example be wall following if there is a long wall next to the robot. To make the route more robust, the algorithm needs to find as much as possible bump points. The algorithm should also take into account errors in the map. The algorithm should be tested on the following points:

- Is the robot still able to follow the automatically created actions when there are one or more errors of 5 cm in the map? This should be tested by drawing several maps with various errors for every barn.
- Are the automatically created routes robust enough? Routes should be drawn by different users and be evaluated by some field testers. The robots should drive the created routes for several days with not more than one mistake related to the route in two days.
- Are the created routes in line with the drawn routes? The routes should clean the same space in the same direction of the drawn routes with a minimal coverage of 90%.

18.12 User study

The user should be able to drawn the map and the routes in a reasonable time-frame. The user should do this in a satisfying way. The program should be tested on the following points:

- Is the user able to draw a map with 1cm accuracy? This should be tested for different farms by different users.
- Is the user able to draw the routes to his wishes? Are the routes drawn on the paths he wanted to drive the robot the route. This can be tested by first letting the user explain in the barn how the route should be done. Then the explanation can be compared with the drawn route.
- Is the user able to draw the map and the route in not more than a half hour?
- Can the user program the routes in the same time or faster than the current programming method? This can be tested by programming the routes with both methods, in different order.
- Is the user convinced enough that his drawn routes are correctly driven by the robot?
- Is the user satisfied about the program?
- Is the user able to make good routes with the program?

18.13 Map making and route programming by drawing

Before we make a program by our self, we need to do some research on existing programs if they are usable. If this is the case, this can save a lot of time on developing and support.

DraftSight

This program is a copy of AutoCAD and contains almost the same functionality.

Pros

- All the functionality you want is in it

Cons

- Too complex, you need some hours to get handy with it
- Too much options, the normal user like the farmer or the technician does not know what and how to use
- Before the program can correctly be used, a lot of settings first need to be set
- Export function to the right format need to be developed. We can also make the export to svg usable for importing
- Need to pay if we want to write plugins or to use it for an application
- Not possible to define different objects, like a wall and a charger
- We can’t add functionality to give directly feedback off the route to the user

Other simple CAD programs

- We mostly miss some functionality that we just want or need
- Same cons as DraftSight

Other simple drawing programs

- Mostly you can only print the drawing and you can’t export the drawn objects to another file

Edraw Max, floorplan

- Predefined objects
- Walls seen visual nice
- Walls have some thickness which cannot be set
- Usable for home decoration but not for making a map which can be used for programming the routes

Draw in web-browser

- Miss too much functionality

Conclusion

There are a lot of different programs available on the internet, but it is difficult to find an application that we can use for our prototype. Mostly they are too complex and if they are simple, they miss too much functionality. Besides that we also couldn’t find a program that can draw a map and define routes in another way, so we still need to build that functionality. For this reason we choose to build the prototype from scratch.