A shared control approach for obstacle avoidance in robot supported USAR

Master Thesis
Master of Science (MSc) in Engineering (Robotics)

Karolina Dudzińska
Adrian Herskind

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg
Spring 2021
Group 1064

Abstract

The development of more advanced robotic technology gives a rise in possible applications. A field with great potential is the area of shared control. Here the human operator and the robot are both involved in solving the task at hand. This can happen at various degrees of autonomy, from where the robot has almost no decision power to fully autonomous solutions. Often in shared control tasks, the responsibilities are split according to the individual abilities. The need to share control can also arise, if neither of the involved parts have access to all the information required to solve the task. One such task is Urban-Search-and-Rescue with a mobile robot, where the operator only has access to a high level map of the building, while the robot can sense obstacles locally. These obstacles pose a threat to the robot and need to be avoided. In this experiment we investigate whether using different degrees of autonomy during obstacle avoidance has an effect on task performance. Participants control a TurtleBot3 in a Gazebo simulation of a house. Obstacles are invisible to the operator and need to be avoided to successfully complete the task. We design and test three different conditions with increasing degree of robot autonomy, from teleoperation, over shared control to full autonomy. The change in autonomy is through robot initiative, but operators have the authority to reclaim control. In the (semi-)autonomous conditions, we use a potential field method for reliable and efficient obstacle avoidance. Audio feedback about changes in autonomy are provided to the participants. We measure how fast the participants complete the task and find no significant differences between the conditions. For subjective measurements we use the QUEAD and NASA-TLX questionnaires and find only one significant partial result, favoring teleoperation. However, we observe a general trend, both in objective as well as subjective measurements, that favor the use of the teleoperation condition. We conclude that participants might feel more comfortable having full control over the robot, but that further research is needed to investigate additional scenarios, autonomy modes, and ways to transfer control.

Contents

Chapter 1

Introduction

With the continuous development of more intelligent and flexible robots, scenarios have shifted from robots working in separated cells on their individual tasks to a wide range of applications. These include collaborative tasks where humans interact with the robot in the same workspace, multirobot systems sharing knowledge amongst themselves only being supervised by a human, and a wide range of tasks where mobile robots break their confines and work autonomously or teleoperated.

An example of such application for mobile robots are Urban Search and Rescue (USAR) scenarios. Robots and drones provide numerous benefits for disaster response. They can often fit into confined spaces where humans cannot be and operate in environments, which are too dangerous for human rescue workers. This is needed in disaster areas, like after earthquakes or nuclear accidents. Additionally, in a USAR context, time is a critical parameter, as the chance of survival of victims decreases rapidly. It is, therefore, crucial to deploy all the USAR assets as quickly as possible. Another benefit of using robots or drones instead of human rescue workers is that one can very quickly obtain an overview of the situation and the dangers in the crisis area. Located in very disaster-prone areas, countries like Japan or Korea are investing many resources into developing novel USAR robotic systems, which could help in case of such events. Those robotics systems have been used after the 2011 Great Eastern Japan Earthquake in Tohoku, Japan, where ground robots helped with monitoring and surveillance operations, whereas marine robots assisted with clearing the harbors [27]. As proven by past successes and impressive research efforts around the world, robotic tools have a great promise to increase the effectiveness of USAR operations.

Operations like these not only require more advanced technology but also increased flexibility. This flexibility does not only apply to the robot in the physical sense, meaning that they can move around instead of being fixed to a work station, but also to approaching and solving a task. This kind of flexibility is necessary both when interact-

ing with humans, who by nature are not entirely predictable, and when being subjected to a task, which might have uncertain or multiple possible outcomes and a possibly infinite number of paths to get there. Flexibility can be expressed through differing degrees to which the robot acts autonomously, ranging from robots which are completely teleoperated by the human to completely autonomous robots. In-between those extremes lie different degrees of automation, describing how much and which parts of the task are automated through the robot and which remain under human control. There is a variety of different scales trying to categorize these nuances. These scales are generally called levels of automation (LOA). Whenever the robot is not controlled entirely by teleoperation or by full autonomy, it can be classified as shared control.

One reason to share control in a USAR task is that the human and the robot possess different qualities, which make them better for certain parts of the task, e.g. the robot has very precise sensors and the human can interpret the data better. In this way, shared control would help utilise the best skills of each actor to improve the outcome of the task. Another reason to share control is, if each of the actors only has access to partial information, so shared control is necessary to be able to complete the task. This could be, for example, if only the robot has access to sensor data, but only the human has knowledge of the task, or if the human is teleoperating the robot from a distance but only has a high level map and the robot can gain knowledge of the more immediate low level surroundings. How situations with partial information are handled depends on the circumstances and the task, the robot could for example only sense the surroundings and relay this information to the human or it could completely analyse and interpret the data and send its evaluation to the human.

A specific example of how partial information can be utilised is for obstacle avoidance in USAR tasks. While controlling the robot, the operator might only have a blueprint of the house, seeing walls and doors. However, there will be other obstacles, which the operator will not be aware of, like furniture and, depending on the degree of catastrophe, the building might have partially collapsed, adding debris as possible obstacles. This means that the operator has to deal with unexpected invisible obstacles, which they, naturally, need to avoid driving into. To ensure a successful USAR mission, it is important to avoid these obstacles in a fast and safe manner, imposing a small amount of workload on the operator, so they can concentrate on their actual task of identifying victims. Here the robot can use sensors to collect data about possible obstacles and either relay them to the operator or autonomously take efforts to avoid the obstacles.

LOA are not necessarily strict constructs selected once when initiating the task, but are often dynamically switched between during the task execution, depending on the requirements of the current situation. This of course prompts the question who has the authority to change between the LOA; the human, the robot, or both equally, and if we consider the latter, how much authority should which actor have. This authority of change is not necessarily an all or nothing question, so the robot might have authority

to change the LOA in certain situations or for certain parts of the task while the human has the authority otherwise. One of the actors could also have the authority to change the LOA and the other actor could have the power to override this decision. The possibilities are manifold and highly depend on the task requirements and the capabilities of the robot and the human. This concept is called mixed initiative.

In the case that the robot has full or partial authority to change the LOA, one needs to consider how and if this change is communicated to the human. This type of feedback can help the human to understand where the control of the robot lies and can help to ease transitions between LOA, so uncertainties are eliminated. Depending on the number of LOA, which are available, different feedback methods can be applied. In simple binary cases, a signal can notify the operator of the change of LOA, while a larger number of possible LOA would require a type of display or voice feedback, informing the operator of the current state. Being kept in-the-loop can lead to higher acceptance of the robot's authority, better anticipation of upcoming changes in LOA, and the possibility for oversight and intervention. One also needs to consider how fast the LOA are changed, whether it is instantaneous or if it should be a sliding transition, which opens up for alternative feedback methods. However, again, the optimal feedback is dependent on the task, robot hardware, and operator skill.

Initial Problem Formulation

A fast developing and exciting application field for mobile robots is USAR where small, agile robots are deployed to help search for survivors of natural catastrophes, like earthquakes, nuclear fallout, or fires. However, the distance between the robot operator and the robot itself provides some difficulties. Exchange of data might be unreliable and the operator might not have access to the same information as the robot, meaning that, for example, unexpected obstacles would not be visible to the operator. This necessitates a certain level of automation from the robot, which could go from collection of sensor data, over evaluation of data and planning avoidance behaviour all the way to fully autonomous obstacle avoidance. Luckily, systems are not limited to work at a fixed LOA but can be changed flexibly if the situation requires it, for example could the operator have full control over the robot, but when an obstacle is detected the system allocates control to the robot who then avoids the obstacle at a given LOA and afterwards hands back control to the operator.

With these considerations in mind, we pose this initial problem formulation:

How do different LOA affect obstacle avoidance in USAR when the human operator cannot see these obstacles?

Chapter 2

Related Works

Shared control is a broad field with many aspects, which have to be taken into consideration. In this chapter, we look at the existing research in shared control and with the initial problem formulation in mind we try to find a base for our project.

Models of automation

When looking at automation in robotics, we first need to define a model of automation, which we base our efforts on. There are two concepts of automation, which display the possible effect it has on the human operator. On the one hand, a higher degree of automation could reduce the number of interactions required to be taken by the operator [1] [21]. On the other hand, the frequency of interaction could remain the same but the type of interaction could change in method or complexity [35] [17]. Still, a combination of both is also thinkable, less frequent interaction but more challenging decisions could be present.

We take a base in the definition of automation by [28], which states that automation is a "*device or systems that accomplishes (partially or fully) a function that was previously, or conceivably could be, carried out (partially or fully) by a human operator*". The addition of **partially or fully** implies that automation is not an all-or-nothing option, but can be found in various degrees. One of the earliest attempts to build a scale of automation was done by Sheridan and Verplank [32] in 1978 and concerned the *"levels of automation in man-computer decision making"*. In their 10-level scale, shown in Table 2.1 they describe what responsibility in a task is executed by the human and which is relegated to the computer. Since this scale is focused on a computer, the substeps of each decision making process only include: getting options, selecting options, approving selections, starting action, and informing about decisions. It is always the computer executing the task in the end and it is always a third party which provides the possible options, meaning there is no sensing involved. On the two extremes, on level 1,

the lowest LOA, the human operator does the entire job and lets the computer execute the selected action (similar to teleoperation of a robot, with the important notice that execution of an action in the robot would only relate to the hardware side, i.e. translating the desired velocity into respective voltage output for the actuator) and on level 10, the highest LOA, the computer does the entire procedure of getting possible options, selecting an action, starting the action, and informing the user, if it decides to do so. This full autonomy level has the interesting notion that the computer has the decision power whether even to inform the human about the action it is implementing, meaning that the system is not only having authority on action taking but also on keeping the human in-the-loop. Additionally, the computer also has the power to decide whether the action should even be implemented at all. Sheridan's LOA scale is still widely used and referred to in research.

LOA	Description
1	Human makes all decisions, up to the point of turning the job over to the computer to implement
2	The computer provides the options, but the human makes all decisions and starts actions
3	The computer determines all options and suggests one, which the human does not have to follow.
4	The computer determines all options and selects one, which the human may or may not send to implementation
5	The computer selects an option and implements it if the human approves
6	The computer selects an action and implements it, but gives human time to stop it.
7	The computer does the whole job and informs the human what it did.
8	The computer does the whole job and informs the human, if the human explicitly asks.
9	The computer does the whole job and informs the human, if it decides to do so.
10	The computer does the whole job, if it decides that it should be done and informs the human, if it decides to do so.

Table 2.1: Sheridan and Verplank's model of LOA

The lack of input in Sheridan's LOA was addressed by Ensley and Kaber in 1999 [15] who devised an updated model for LOA, similarly including 10 levels and also addressing how information is acquired and options generated. The model is shown in Table 2.2.

LOA	Description
Manual Control	The human monitors, generates options, selects options (makes decisions), and physically carries out options.
Action Support	The automation assists the human with execution of selected action. The human does perform some control actions.
Batch Processing	The human generates and selects options; then they are turned over to automation to be carried out (e.g., cruise control in automobiles).
Shared Control	Both the human and the automation generate possible decision options. The human has control of selecting which options to implement; however, carrying out the options is a shared task.
Decision Support	The automation generates decision options that the human can select. Once an option is selected, the automation implements it.
Blended Decision Making	The automation generates an option, selects it, and executes it if the human consents. The human may approve of the option selected by the automation, select another, or generate another option.
Rigid System	The automation provides a set of options and the human has to select one of them. Once selected, the automation carries out the function.
Automated Decision Making	The automation selects and carries out an option. The human can have input in the alternatives generated by the automation.
Supervisory Control	The automation generates options, selects, and carries out a desired option. The human monitors the system and intervenes if needed (in which case the level of automation becomes Decision Support).
Full Automation	The system carries out all actions.

Table 2.2: Ensley and Kaber's model of LOA

However, a model for automation is not enough when one considers that despite the similarities they share, automation and robotics have some fundamental differences, which makes it necessary to devise a model of LOA, that takes these into account. This is called levels of robot automation (LORA). These differences are both visible in the

functionality of the robot as well as its physical appearance and are including but not limited to social interaction, manipulating the environment, and mobility. In 2014, Beer et al. [7] proposed a model for LORA aimed at including these specific aspects of robots. For each level, the authors split the task into the substeps of: 1) Sense, 2) Plan, 3) Act and specify whether it is the robot (R), the human (H), or both (H/R) who execute this part and give a detailed description of this level. However, the authors do specify that they see autonomy as a continuum and the LORA they propose should merely be seen as a suggestion for waypoints along this continuum, enabling implementations where the autonomy is more blurred, lying in-between these steps. The model can be seen in Table 2.3.

LORA	Sense	Plan	Act	**Description**
Manual	H	H	H	The human monitors, generates options, selects options (makes decisions), and physically carries out options.
Tele-operation	H/R	H	H/R	The robot assists the human with action implementation. However, sensing and planning is allocated to the human. E.g., a human may teleoperate a robot, but the human may choose to prompt the robot to assist with some aspects of a task (e.g., gripping objects).
Assisted Teleopera-tion	H/R	H	H/R	The human assists with all aspects of the task. However, the robot senses the environment and chooses to intervene with task. E.g., if the user navigates the robot too close to an obstacle, the robot will automatically steer to avoid collision.
Batch Processing	H/R	H	R	Both the human and robot monitor and sense the environment. The human, however, determines the goals and plans of the task. The robot then implements the task.
Decision Support	H/R	H/R	R	Both the human and robot sense the environment and generate a task plan. However, the human chooses the task plan and commands the robot to implement actions.
Shared Control W/ Human Initiative	H/R	H/R	R	The robot autonomously senses the environment, develops plans and goals, and implements actions. However, the human monitors the robot's progress and may intervene and influence the robot with new goals and plans if the robot is having difficulty.
Shared Control W/ Robot Initiative	H/R	H/R	R	The robot performs all aspects of the task (sense, plan, act). If the robot encounters difficulty, it can prompt the human for assistance in setting new goals and plans.
Executive Control	R	H/R	R	The human may give an abstract high-level goal (e.g., navigate in environment to a specified location). The robot autonomously senses environment, sets the plan, and implements action.
Supervisory Control	H/R	R	R	The robot performs all aspects of task, but the human continuously monitors the robot, environment, and task. The human has override capability and may set a new goal and plan. In this case, the autonomy would shift to executive control, shared control, or decision support.
Full Autonomy	R	R	R	The robot performs all aspects of a task autonomously without human intervention with sensing, planning, or implementing action.

Table 2.3: Beer et al.'s model of LORA

Although a classification with 10 levels might be technically necessary to describe all logical combinations of task sharing, most applications require only a few of those levels to be used. Bruemmer et al. [10] consider the use of a robot for USAR missions and select four different autonomy modes, which are defined as following:

1. **Teleoperation:**
   The operator has full control over the robot on a low level.

2. **Safe Mode:**
   While the operator still has full control over the robot, the system continuously assesses its status in the surrounding and tries to protect itself by placing the minimum amount of limits on the human, e.g. by stopping before a collision.

3. **Shared Control:**
   The robot autonomously makes decisions about settings its path and responding to the environment accordingly. However, this control is mainly reacting to the direct surroundings and through obstacle avoidance. This is supplemented by the operator through general directions, often given at the request of the robot.

4. **Full Autonomy:**
   The robot can fully autonomous plan and execute a high level task given by the operator.

Other studies [5] [14] have picked up these four modes suggested in the paper. Bruemmer et al. [9] showed in another study that those modes were usable by participants to successfully find objects in a search task.

Mixed Initiative

LORA are not necessarily fixed during the execution of a task, but can shift variably based on what the current situation requires. This change in LORA can be called mixed initiative, given that it can be initiated by both the robot and the human. If the robot system has exclusive control over the change of LORA, it is called adaptive autonomy (or robot initiative), and conversely, if the human has sole authority it is called adjustable autonomy (or human initiative).

Wang and Lewis [37] used a USAR scenario where a multi robot system (MRS) explored a simulated building environment to find victims in the aftermath of an earthquake. While the robots work in autonomy mode, they sense their environment and find optimal paths by communicating with each other to avoid overlapping paths and thereby duplicated effort. The operator of the robots has the authority to take control over any robot at any time and either teleoperate it or give it new way points, during which it is not communicating with the other robots. After the robot has executed the commands of the operator, it goes back to autonomy mode. The authors saw it as important to integrate autonomous behaviour in a MRS, since a single operator would have difficulty

planning and executing the behaviour of several robots. The mixed initiative, however, would still allow the human operator to exert a supervisory function and intervene if necessary. They also mention that their current system does not allow for automatic victim recognition and while it could probably be solved with a more advanced system it might be favorable that such a crucial part of the USAR task should be done by the operator. Wang and Lewis found that performance (i.e. percentage of region explored, number of victims found) was slightly higher in the mixed initiative mode compared to full autonomy mode and significantly higher than in the manual control mode. They look at other studies, which found high autonomy to be more optimal, and conclude that in more complex tasks like USAR, splitting it into subtasks of navigation and perception, could play to the strengths of the human and the robot and thereby increase performance.

Another study, which takes the scenario of USAR, is by Baker and Yanco [5]. The authors' motivation was their observation that operators for USAR tasks often either do not use the available autonomy modes efficiently or even chose modes, which were doing more harm than good. Reasons for this are that operators are unfamiliar with autonomy modes or see the robot merely as a tool, and even though operators received introduction exercises beforehand they rarely switched between autonomy modes. To counter this, the authors devised a system which would suggest autonomy modes to the operators. They defined four autonomy modes (Teleoperation, Safe, Shared, Autonomous), based on [10], as described above, and two autonomous behaviours (Escape, Pursuit), which get suggested to the operator on the interface based on the current situation of the robot. The system will learn and adapt to the preferences of the operators and show fewer suggestions. However, the authors have not tested their system.

Brookshire et al. [8] look at the scenario of a robot team assembling a structure, comparing performance for teleoperated system, fully autonomous system, and system with sliding autonomy. In the sliding autonomy mode, the system usually runs on full autonomy, unless: a) human initiated - the operator takes over control if they see a need to intervene b) scripted - a subtask is designated beforehand to be executed by a human c) robot initiated - the system continuously estimates its own confidence in executing the task and can at low confidence request the human to take over. The authors found that sliding autonomy and teleoperated modes had higher success rates than the fully autonomous mode and that the sliding autonomy had a reduced completion time compared to the teleoperated mode. This suggests that a mixed initiative approach with sliding autonomy both increases performance and efficiency in a task with multiple robots.

Hardin and Goodrich [18] took a Wilderness Search and Rescue scenario and compared the performance of a multi robot system in three different modes: Mixed Initiative, Adaptive Autonomy, and Adjustable Autonomy. Participants would use a simulator to supervise/control a group of 200 robotic searchers with the goal of finding missing people and scattered objects. The authors hypothesized that mixed initiative would be

superior to the other two, since complex tasks benefit from complementary abilities of the human and the robot, meaning that a human might be superior at task-specific knowledge, while the robot team could perform better at systematic routines. The authors concluded that mixed initiative performed better than the other two modes. For this they identified three possible reasons. As the first reason, they confirmed the suspected complementary abilities, since they could show that the mixed initiative and adjustable autonomy perform equally strong in metrics related to primary and secondary performance metrics, which made use of task-specific knowledge, and conversely mixed initiative and adaptive autonomy performed equally strong with management and workload related metrics, such as idle time and supervisor awareness, things related to systematic routines. The concept of complementary abilities is similar to what Wang and Lewis [37] and Kortenkamp et al. [23] reason with in their studies as a benefit of mixed initiative. As the second reason, they mentioned that mixed initiative solves the issue of micro management, as both informed and uninformed operators perform equally well. This would suggest that the level of expertise might not be relevant when using mixed initiative since this mode would cancel out such differences. As the third reason, they listed that in multi robot systems with a sufficiently large number of agents, there is never any idleness since there are always enough robots to contribute to the task even while some wait idly for operator input. These results suggest that mixed initiative outperforms other modes and might not be highly dependent on operator expertise. However, the study was based on large multi agent robot teams and many results are most likely only transferable to such same systems.

In mixed initiative, it is necessary to determine a measure for when the robot should switch the LORA. In [12], the authors devised an Expert guided Mixed-Initiative control switcher, which based on previous experience and guidelines determines whether the human operator currently performs the task at the same level as an expert user would. If this is not the case the system will switch the LORA and allocate control to the robot. The human, on the other hand, had authority to switch the LORA if they saw fit. In a USAR scenario the authors found that their expert guided system outperformed human initiative (i.e. adjustable autonomy) control both in the objective task performance as well as in the subjective perceived operator workload.

In conclusion, several studies indicate that mixed initiative control modes improve task performance and operator load, but that this effect is mainly present when the task consists of subtasks, which can advantageously be split between the robot and the operator, utilizing their individual strengths. Furthermore, it must be noted that many mixed initiative studies are either, based on robot teams or swarms, which are too large to be controlled by a single operator, or used for distal tasks like USAR where due to the nature of the task the human operator and the robot each only have access to partial information. This could be overall task knowledge and maps for the human operator and local scanning and obstacle avoidance for the robot.

Two major reasons for mixed initiative stand out. The first reason is difference in

skills. In tasks, which consist of more than a single process, there is always the possibility that one part of the task is better suited to be solved by a human, while another part of the task falls under the expertise of a robot system. The second reason is that not all information and data might equally be available to both. While the human might know the goal of the operation or the whole map of the surroundings, the robot might be unaware of it, but instead be equipped with a scanner with which it can scan the local surroundings. Which type of mixed initiative approach one should take, depends on the task and the circumstances.

Feedback

Feedback is an essential part of all communication, not only is it used to signal continued attention, but also to signal perception, understanding, or danger. Since feedback is an essential component in the interaction between humans and robots, feedback methods have been researched to find a suitable method for different scenarios and tasks. Since there is no one universally applicable method of feedback it resulted in the development of many feedback methods, which each come with their advantages and disadvantages.

In [38], haptic feedback was used to increase the embodiment of a certain robotic device and the feeling of immersion in human-robot interaction. Haptic feedback is a technology that creates an experience of touch by applying forces, vibrations, or motions to the user. As presented by the authors, tactile feedback is commonly used in robotic surgical systems. It gained recognition when it was designed and integrated with the Da Vinci Surgical System [26]. The system presented in [38] was using sensors placed at the tip of the robot's grasper and converting them to pressure at the surgeon's fingertips, causing the deformation of a finger pad and creating the experience of touch. The system was tested on surgeons and novices. The task was to pass porcine bowel from one grasper to the other. For each participant, the test was run in the following three sequential conditions: haptic feedback off, haptic feedback on, and haptic feedback off. Results showed that with the addition of haptic feedback, less force was used, which correlated with less tissue damage to the bowel.

Haptic feedback was also used in [30] where the authors present a system in which a human operator controls a robot remotely through some control device while receiving haptic force feedback. Saeidi et al. focus their work on the role of mixed initiative in human robot interaction. In mixed initiative control scheme, users' trust in the system can highly influence the interaction. Therefore, the authors propose computational models of two-way trust, i.e. human-to-robot trust and robot-to-human trust, as metrics to dynamically adjust the control. Additionally, they decided to scale the force feedback with variables to which level of trust were assigned. This provides various levels of assistance to the operator. The authors tested their system and results showed that operators, who controlled the system without receiving haptic feedback performed worse with increased workload compared to operators, which received haptic feedback.

Another feedback method is visual feedback. Visual feedback is a visual output from a system that allows the user to interact better with it. The use of vision feedback with robots has a long history and nowadays vision systems are commonly available and highly integrated with robot programming systems.

In [13], the authors attempted to improve the performance on trajectory planning and parking tasks by using visual feedback. The authors mounted a black surface with three LED's on top of the robot in order to reconstruct the robots posture. The solution was then tested and compared to other feedback methods. The performance improved when using the visual feedback.

Another study that uses visual feedback is [6]. The authors compared the effect on motor output in subjects during robotic-assisted gait training when they used visual feedback and when they followed verbal instructions of a physical therapist. The authors tested their system on 12 subjects with neurological gait disorders. Results did not show significant difference in motor output in either condition. However, subjects reported that they were more motivated and concentrated on their movements when using computerized visual feedback compared with instructions from the therapist.

A different type of feedback was investigated in [34], where the authors used audio feedback to create a reverse car parking assistant. For their scenario, they have used Arduino Uno-based microcontroller, which was equipped with two distance sensors and an MP3 Shield.

The MP3 shield was used to give audio feedback to the user to reverse, drive slowly, or to stop. The authors tested their algorithm and the results showed that the system worked correctly, based on the distance to barriers. However, they did not provide any data for the conducted tests.

Another use of audio feedback was presented in [22]. In the article, high school students, instead of only receiving written feedback of their writing assignments, were additionally receiving audio feedback. The authors, based on previous research, show some benefits of using audio feedback such as: it can be easier to understand than other types of feedback and can be easily personalised. In the experiment 126 high school students participated. The results supported previous findings from [29] that "students liked or were motivated better by audio feedback". In addition many of them pointed that they perceived audio feedback as more detailed. Results showed that the majority (66.7 %) of students surveyed preferred the voice comment feedback over written comments.

Much state-of-the-art research on feedback in HRI is dedicated to investigate single feedback modalities instead of multimodality. However, more researchers are exploring those possibilities; an example of that is using a combination of audio-visual feedback as presented in [36]. In the experiment, subjects were asked to steer the robot and perform a simple task such as picking a bottle. The authors combined visual feedback with human footstep sounds as audio feedback. The scenario was tested on 28 subjects and the results showed that the combination of two feedback methods increased the

users' control over the robot.

With the fast development in robotics more custom feedback methods are being developed. An example of those is presented in [33], where the authors developed a feedback method based on brain signals. This type of feedback can be very useful in applications related to neurorehabilitation or neuroprosthetics.

The methods presented above do not include all types of feedback, because the possibilities are very extensive. However, they all show the importance of feedback in communication and task performance.

Autonomous Obstacle Avoidance

When mobile robots drive (semi-)autonomously through an unstructured unknown environment, they need to be able to identify and avoid obstacles. In simple applications, bug algorithms might suffice to avoid obstacles, but these are not always complete, possibly inefficient, and might rely on knowing the goal, which is not always given, especially in open-ended tasks, like USAR.

One method for autonomous obstacle avoidance is potential field methods, which is the one we will focus on in this project. Potential fields are gradient vector fields, assigning a vector to each point in space, representing a gradient towards the minimum of a given function. A way to reach this minimum is called gradient descent. Potential field methods or potential functions define this vector field by assigning attractive or repulsive potentials. Goals are assigned attractive potentials and obstacles are assigned repulsive potentials. The attractive potential gradually increases with distance from the goal, while the repulsive potential gradually decreases with distance from an obstacle. The closer the robot gets to an obstacle the more it gets repulsed, thereby avoiding collision. The gradient of this potential function then ideally gives the optimal path towards the goal. Potential field methods have generally been successfully used in obstacle avoidance as we will outline below.

Horiguchi and Sawaragi [20] use a potential field method to calculate repulsive forces from each of the robot's 7 sensors if they detect obstacles. A weighted sum gives a total velocity and steering value for the robot in order to avoid obstacles autonomously.

Aigner and McCarragher [2] also use a potential field method to generate a potential field with obstacles acting as repulsive forces and the goal acting as an attractive force. However, unlike [20] they also integrate input from a human operator by combining the determined velocity of the autonomous system with the velocity desired by the human, making it a shared control task with a medium LORA. In experiments they showed that the human operator could influence the direction and velocity of the robot but was not able to steer into an obstacle or direct the robot too far away from the goal, since in those cases the potential field method got too strong and intervened. Therefore, the authors showed that potential fields are a viable option in shared control scenarios.

Gerdes and Rossetter [16] successfully used potential fields to develop driving assistance systems, which help with lane keeping.

Chapter 3

Methods

In this chapter we consider the existing research, presented previously, and apply some of the concepts to our initial problem formulation in order to design an experiment.

This project takes inspiration from Bruemmer et al. [10], adopting the idea of using a limited number of LORA in USAR missions. The presented research has confirmed that USAR missions are particularly interesting for shared control, since the mission consists of both tedious low level tasks, such as navigation and obstacle avoidance, as well as complex high level tasks, like detecting victims and covering wide areas. Here, complementary abilities could boost performance as suggested by [18] [37] [23]. Furthermore, the robot is executing a task at a different location than the human, meaning that sensory information might not be equally available to both actors. This involuntarily necessitates a shared control approach.

While most of the presented mixed initiative studies are based on the presumption that it is the robot acting autonomously, requesting the human to take control or the human claiming control, we consider the findings by Baker and Yanco [5], that humans tend to not make use of different LORA even if it benefit them and that the system should, therefore, suggest LORA to the operator. We take this a step further by giving the system authority to autonomously change the LORA, when it sees fit, in this case when it detects an obstacle, also called adaptive autonomy. This might also make the transition smoother, compared to systems where the robot sends a request to the operator to take over, which first needs to get accepted/denied.

Exploratory Research Question

Research generally shows that using different LORA has a beneficial effect on task performance, but often the methods are mainly applicable for the specific task. While

many USAR research projects are focused on improving victim localisation and operator support, we have not found any studies looking into obstacle avoidance when the operator cannot see the obstacle. Given the novelty of this study, we dare not make any predictions to whether a higher autonomy level is superior to a lower autonomy level.

With these considerations in mind we try to design an experiment, which provides data to answer the following question:

Does the LORA in a mixed initiative USAR scenario influence the task performance in obstacle avoidance when the operator cannot see the obstacles?

3.1 Experimental Design

The scenario in which we test our research question is obstacle avoidance in USAR. In the experiment, the test subject takes the role of the robot operator. They see a high level map of a building only showing walls and the location of the robot. They are asked to control the robot to reach a number of target areas indicated on the map. They are made aware that there will be obstacles they cannot see but which the robot can sense.

To clarify, we will throughout this report use the term *obstacle* in reference to the unexpected obstacles, which are not shown in the map and invisible to the operator, even though walls are technically obstacles as well.

Our main focus is on the avoidance of these obstacles, however, as it is a subtask and difficult to recreate separately, the experiment consists of a complete USAR robot navigation task. Normally, the main task of the operator would be to locate possible survivors, which is often done through thermal camera imaging from the robot. This would require the focus of the operator to be split between robot navigation, to cover and explore the target areas, and victim localisation. However, we have decided to neglect the latter parts and only focus on the robot navigation task.

Test Conditions

To test the effect of different LORA on obstacle avoidance we need to isolate this specific part of the task and define how many and which autonomy modes we will implement as test conditions. For further considerations we can split the task into two distinct states:

1. Robot navigation with no obstacle nearby (now **Free Driving**).
2. Robot navigation with obstacles nearby (now **Obstacle Avoidance**).

Since the focus of the project is on obstacle avoidance, the control mechanisms and LORA during free driving should be identical in each test condition. On the other

hand, when the robot is located within a certain predefined range towards an obstacle, the system should trigger a different LORA depending on the test condition.

Obstacle Avoidance

While Bruemmer et al. [10] make a distinction between four different autonomy modes in their LORA (i.e. Teleoperation, Safe Mode, Shared Control, Full Autonomy), we are content with three modes. To define which part of the task is done by whom, we use the principle established by Beer et al. [7], splitting the task into *Sense*, *Plan*, and *Act*.

We define our autonomy modes for obstacle avoidance as following:

1. **Feedback Assisted Teleoperation:**
   At one end of the spectrum we want to allocate as much control as possible to the human operator and, therefore, define the first mode as teleoperation, meaning that both the planning and acting parts are done by the operator. However, full teleoperation is not possible in this case since the obstacles that need to be avoided are not visible to the operator. Thus, the sensing is done by the robot and the operator. While the robot is doing the sensing related to obstacles, the operator is still doing general sensing like evaluating the position of the robot in the overall map and the position of walls. The information the robot gathers about obstacles, needs to be somehow conveyed to the operator. A number of the presented studies successfully used audio feedback and we decided to use this as well, since we expected a number of advantages from this type of feedback, such as:

   - The operator is already occupied with a lot of visual feedback, i.e. steering the robot through a map, locating victims in a separate thermal camera feed. By displaying obstacles through visual feedback, the operator might get distracted by and overloaded with information and performance could decrease. However, auditory senses are still unused.
   - With audio feedback about the presence of obstacles more information can be conveyed, like the location of the obstacle (shifting direction of beeping in stereo, e.g. obstacle on the left gives audio feedback on the left ear) and the proximity to the obstacle (change in frequency, e.g. the closer the robot is to an obstacle the faster it beeps).
   - Familiarity with audio feedback from everyday applications, e.g. car parking assistance, which shows the same direction and proximity features as described in the previous bullet point. Using a familiar system might make it easier to use.

2. **Shared Control:**
   While being called the same as the mode in [10], our version is differing slightly.

When getting in range of the obstacle, the robot senses the obstacle and by using a potential field method (explained in detail in Section 4.4) the system determines in which direction it should turn to avoid the obstacle. Then the system takes control over the angular velocity of the robot and turns it in that direction. The operator is still left in control over the linear velocity, meaning the robot will only turn but stand still unless the operator gives input to drive forward or backward. This is continuously repeated while the robot is in the proximity of the obstacle, hence the robot will sense the obstacle, plan which direction to turn to avoid the obstacle, and act by implementing this turn. The operator still senses the general map, plans which way to go to reach their overall goal and acts by driving the robot forwards or backwards. Therefore, the control is shared in all three parts of the task.

3. **Full Autonomy:**
   At the other end of the spectrum, full autonomy gives the robot complete control over all aspects of the obstacle avoidance. The robot senses the obstacle, plans which direction to turn to avoid it and executes this plan by both setting the angular and linear velocity. The operator still senses the general map and can plan a way to reach the overall goal, but has no authority to enact this plan.

The autonomy modes for obstacle avoidance are listed in Table 3.1.

LORA	Sense	Plan	Act	**Description**
Feedback Assisted Teleoperation	H/R	H	H	The robot senses obstacles and provides feedback, the operator plans a path and acts by fully controlling the robot.
Shared Control	H/R	H/R	H/R	The robot senses the obstacle, plans a way to avoid the obstacle and sets the angular velocity. The operator senses the general map, plans a way to the overall goal and sets the linear velocity.
Full Autonomy	H/R	H/R	R	The robot senses the obstacle, plans the best way to avoid it and fully controls the robot. The operator senses the general map and plans a way to the overall goal but has no authority to enact it.

Table 3.1: Our model of LORA for obstacle avoidance

In the two (semi-)autonomous modes it is additionally necessary to somehow inform the operator of the change in LORA. When simply changing the LORA in the middle of driving, without any indication, the operator might get confused or irritated, or might

assume that there is an error in the system. By providing feedback to the operator and keeping them in-the-loop, these problems might be prevented and acceptance of the LORA could be increased.

These three autonomy modes will be used as the different test conditions in the experiment.

Free Driving

While the autonomy modes differ for each test condition, the general behaviour for the rest of the test should be identical. The autonomy mode used for the free driving situations is taken directly from Bruemmer et al. [10]. The goal is to let the operator drive as freely as possible, but of course to ascertain that the robot is not hitting any walls. Therefore, the autonomy mode for free driving is as following:

- **Safe Mode:**
  The operator is controlling the robot in teleoperation mode. The robot is sensing for walls and if it is getting too close to a wall it first slows down and then ultimately backs away from the wall to avoid collision.

To avoid collisions with obstacles, the principle of the **Safe Mode** should not only be used in free driving but also in the obstacle avoidance. Thus, while the obstacle avoidance is characterised by the three autonomy modes mentioned previously, each mode should also include an underlying emergency break and backing when the robot gets too close to the obstacle.

Mixed Initiative

In this experiment we use adaptive autonomy for switching from free driving state to obstacle avoidance. This switch will be triggered by the system when the robot gets in a predefined proximity to an obstacle. However, the system is not aware of the overall goal of the operator, i.e. victim localizing. During the obstacle avoidance in **shared control** or **full autonomy** mode it might, therefore, happen that the robot steers into a certain direction - in its best efforts to avoid the obstacle - which completely contradicts with achieving the overall goal. Due to this, it is necessary to give the operator the possibility to claim back control from the system, overwriting the current input to the robot. For the rest of the report we refer to this situation as *breaking out* or *breakout*, since the operator is breaking out from the control of the robot. Naturally, this only applies for the two autonomy modes mentioned above, where the robot holds some action authority, and not for the **feedback assisted teleoperation** mode, where the operator has full control the whole time anyways.

Thus, the change in LORA in this experiment cannot be classified as adaptive autonomy (or robot initiative) but fulfills the requirements for mixed initiative.

To sum up, the change in LORA can happen at three points:

1. The robot gets into close proximity to an obstacle. The system automatically changes to one of the obstacle avoidance autonomy modes (depending on which condition is currently tested).
2. The robot is no longer in close proximity to an obstacle. The system automatically changes to **Free Driving** mode.
3. The operator determines that the robot is no longer on a desired path and decides to claim back control to try to steer back towards the overall goal. The system is changed back to **Free Driving** mode.

Measurements

To measure the effect of different LORA on obstacle avoidance, we collect various kind of data. To get objective data, we need to look at performance based measurements. Previously, we only mentioned that we wanted to look at whether task performance was influenced, thus it is necessary to determine how task performance can be quantified.

In a USAR task it is necessary to reach the victims as fast as possible since it could be deciding about life and death of the victims. Therefore, one of the performance based measurements we will take is **Completion Time**. The easier it is for the operator to avoid obstacles, the faster they will reach the goals.

Previously we described the concept of breakout from partial or full autonomy modes, where the operator can reclaim control if they see it necessary for reaching the overall goal. By counting the number of times that the operator is making use of this breakout function we hope to get data about how willing the operator is to give control to the robot. We will be looking at **Number of Breakouts** per minute, so that the value will not be influenced by the duration of the experiment. We assume that the smaller the value - the more the operator is willing to let the robot be in control.

Task performance is also something the robot operator perceives subjectively. Even if a system works really efficiently, an operator might disprefer it, if it causes too much mental strain.

Therefore, we collect subjective data with two types of questionnaires:

- **NASA-TLX:**
  The NASA Task Load Index is a widely used assessment tool for perceived workload. In the context of this project, perceived workload is a critical aspect for the operator, because of the importance of the task. The lower the workload for obstacle avoidance is, the more excess attention they can spend on the important task of locating victims. We are using the raw NASA-TLX questionnaire, which consist of only the rating scale, without the pairwise comparisons. Some research support that the shorter version of the questionnaire may increase experimental validity [11]. The raw NASA-TLX questionnaire consists of six items rated on a 7-point Likert scale and is shown in Figure A.3.

- **Questionnaire for Evaluation of Physical Assertive Devices (QUEAD):** - The QUEAD scale was developed to measure usability and acceptance in physical Human-Robot-Interaction. The authors in [31] developed a shorter version of the scale which included 16 items, rated on a 7-point Likert scale. The authors claim that the shorter version of the questionnaire does not lead to a loss of information and even increased its reliability. Additionally, by being shorter and faster to complete, it can reduce the possibility of the participant getting bored or frustrated. Elements of the scale can be categorized into five categories: Perceived Usefulness (PU), Perceived Ease of Use (PEU), Comfort (C), Attitude (A), and Emotions (E). Figure 3.1 represents the perceived outcome applying the QUEAD with the five scales PU, PEU, E, A, and C. Figures A.1 and A.2 show the QUEAD questionnaire.

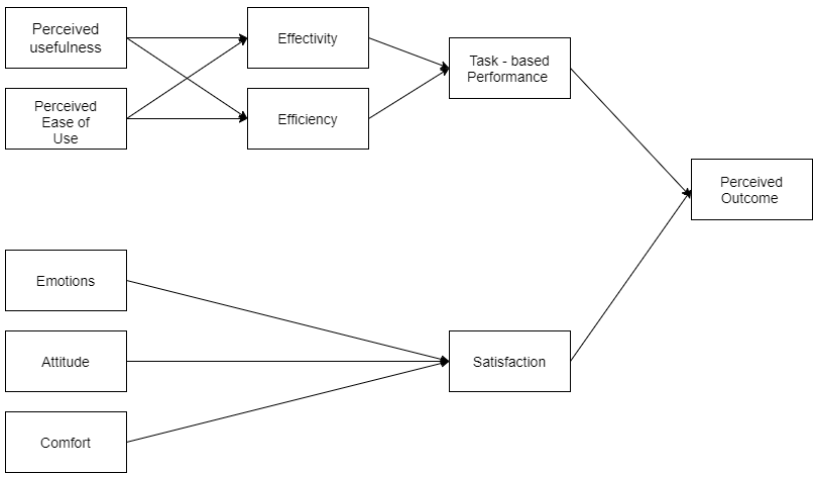


Fig. 3.1: Perceived outcome represented by QUEAD

Furthermore, as a part of the data collection we collect comments from the participants over the course of the experiment. Those subjective comments can be later used to evaluate our experimental design and scenario.

Data Validation

To make sure that the collected data is useful and reliable, different perspectives and aspects of the experiment have to be considered.

Reliability

To make sure that the collected data is reliable, we need to take the measurements in an accurate and consistent way. We choose the QUEAD questionnaire, which is used to measure usability and acceptance in physical Human-Robot-Interaction. In [31] the reliability of both versions of the questionnaire (longer and shorter) were established. The second questionnaire that will be used, is the NASA-TLX questionnaire, which is a well established method for measuring the perceived workload. The reliability of the survey was measured in [19].

Validity

Collected data can be considered valid if the test measures what it is supposed to measure, which is the perceived workload and performance while controlling the robot. The validity of the results can be easily threatened if other variables and factors are influencing the results. Threats to validity can be grouped into two categories - threats to internal validity and external validity.

Threats to internal validity

Most of the factors that may reduce internal validity can be avoided by suitable experimental design. In this section we present some of the most common threats to internal validity and ways in which they can be minimized.

- **Group threats:**
  If the participants assigned to the experimental and control groups are different, the collected data might measure the differences between them, not the differences caused by different conditions. This threat can be minimized by avoiding selection differences and randomly assigning participants to the conditions.

- **History:**
  The participant's history and previous experience can affect the behaviour and performance. To reduce the effect of participant's previous experience we can assess their history with a short background questionnaire before participating in the experiment. Additionally, we can provide all participants with test runs with the robot, to try to level the participants' skills and experience.

- **Time threats:**
  Over time, events may occur, which can change the participant's behaviour. Those events may be caused by some external factors (e.g distractions), or by some inner experiences (e.g boredom, tiredness). We can try to minimize the effect of those by making sure that the experiment is conducted in an environment with reduced external distractions. Furthermore, it should not last too long which may tire the participant.

- **Reactivity and Experimenter Effects:**
  Measuring people's behaviour may change participants' actual behaviour. If participants become aware what the subject of the experiment is, they can start behaving differently. To minimize this effect is to use a well-designed questionnaire and measures, which can hide the true focus of the study. Another element, which can influence the participant's behaviour is the experimenter themselves. The experimenter's age, race, sex, and other characteristics may affect the results they obtain [4]. Ideally, to minimize this effect would be using the 'double blind' technique, which means that both the experimenter and the participant are unaware of the focus of the experiment and testing conditions. Unfortunately, we are unable to apply this technique. In order to minimize this effect we limit the experimenters interaction with the participant to a minimum. All instructions and questionnaires are handed to the participant in paper form, so that the experimenter will interact with the participant as little as possible.

Threats to external validity

External validity of the experiment can be described as to what extent the result of the study can be generalized to the entire population or other situations. According to studies [4] it is more difficult to deal with the effects of threats to external validity.

- **Over-use of special participant groups:**
  In general, participants who take part in an experiment are volunteers. Volunteers generally have a higher interest in and knowledge about the subject of the study, which does not necessarily apply to the entire population.

- **Restricted numbers of participants:**
  It is impossible to use the entire population as participants of the experiment. Using a part of the population means that the results will have to be generalized to the population as a whole.

Within-Subject and Between-Group Design

Within-Subject design consists of using the same subject to go through all the experimental conditions. This method generally results in less noise, as some of the threats to the internal validity are minimized by the fact that the same person is used and, therefore, there are no differences stemming from the personality or personal skills. However, if a person is involved in several conditions, they may become bored, tired, and fed up by the time they come to the second condition.

Between-group design uses different groups of people to test different conditions, so that each person is only exposed to one condition of the independent variable. The biggest disadvantage of the between-group design is the fact that differences between participants in the groups may affect results, for example: different age, gender, or

social background. Although it would appear that using Within-Subject design would a better and more reliable choice for the experiment design, it is also important to take into account other conditions, which are related to the specific experiment.

When one participant takes part in the experiment more than once, they will become more familiar with the test environment, which will significantly influence the performance. Additionally the participant's skill in controlling the robot may increase over time, which can also highly influence the participant's behaviour and performance. As a last point we can mention that the participant would learn the locations of the invisible obstacles in the first condition and then have a clear advantage in the other conditions. Of course we could make different obstacle maps for each condition but then this is another disturbing factor.

Therefore, we decided to use the Between-Group Design for the experiment.

3.2 Hardware

To implement this experiment we use a TurtleBot3 Burger mobile robot and the scenario is implemented in a Gazebo Simulation.

TurtleBot3 Burger

The TurtleBot3 Burger is a low weight (1 kg) mobile robot with a maximum translational velocity of 0.22 m/s and a maximum angular velocity of 2.84 rad/s. It is equipped with a 360° Laser Distance Sensor LDS-01. The LDS-01 can scan in 360° with a minimum scan distance of 120 mm and a maximum scan distance of 3500 mm.

Gazebo Simulation

We use Gazebo version 9 and program the system in ROS Melodic.

In the Gazebo simulation we can create a world defined through an SDF file.

3.3 Implementation Outline

Given the experiment described above we have identified the following parts that need to be implemented.

1. **World and Map Creation:**
   The experiment takes place in a simulated world, a building, in Gazebo. We need to create the world and then turn it into a map from which we can extract information, e.g. like the position of walls.

2. **Control of the Robot:**
   The participant needs to have some way to control the robot in the simulation. We need to connect an external controller to the system and send the input signals to the robot.

3. **Obstacle Identification:**
   The system needs to be able to differentiate between obstacles and walls, because we want to implement different behaviours depending on what the robot is facing. Thus, the system constantly evaluates the sensor data of the scanner and determines whether there is something in close proximity and if yes, what type of object it is; a wall, which is on the map and that the operator should be aware of or an obstacle, which the operator cannot possibly know of.

4. **Supervisor System:**
   At the heart of the program, we will implement a supervisor system, which constantly evaluates the position of the robot, interprets the data it gets from the obstacle identification, and changes the LORA as necessary. Furthermore, the supervisor system allocates control of the robot to either the operator or to itself depending on the LORA. Specifically, the following parts have to be implemented:

   - **Free Driving / Safe Mode:**
     The operator drives the robot freely. When it comes close to a wall, the speed gets reduced and ultimately the robot is stopping and backing up.
   - **Potential Field Method:**
     The two (semi-)autonomous obstacle avoidance modes rely on getting information about the relative location of the obstacles. This is done through potential field methods.
   - **Feedback Assisted Teleoperation:**
     The operator controls the robot freely. The system gives audio feedback regarding the location of and proximity to an obstacle.
   - **Shared Control:**
     The system calculates and enforces the optimal direction to avoid the obstacle. The operator still controls linear movements.
   - **Full Autonomy:**
     The system calculates and enforces the optimal direction to avoid the obstacle and also controls linear movements.
   - **Feedback:**
     In the (semi-)autonomous modes the system indicates to the operator that it has (partial) control.
   - **Breakout:**
     The operator needs to be able to breakout from the (semi-)autonomous robot behaviour.

Chapter 4

Implementation

For implementing the system we split up the program into several parts as explained previously. Each part of the system adds part of the functionality.

Figure 4.15 and figures throughout the chapter show a flowchart of the entire system with red marked, which part of the system is described currently.

4.1 World and Map Creation

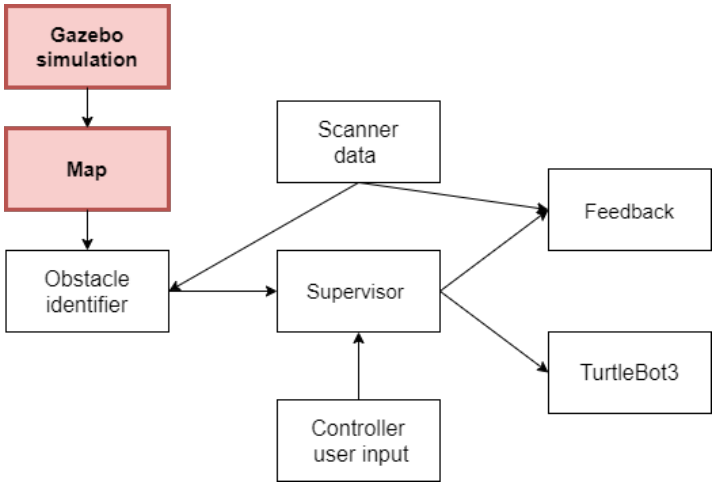


Fig. 4.1: System flowchart

World

The experiment takes place in a simulated world in Gazebo. The world is supposed to represent a floor plan of a building, only showing walls.

The world we have created is $20m$ x $20m$ big, with the coordinate $(0, 0)$ being in the center. The world is in an SDF file format. In SDF, objects are defined through both, a collision tag and a visual tag. Collision tags define the behaviour of the object when getting in contact with another object, or when being hit by a sensor. Visual tags define how the object appears in the simulation. Code 4.1 shows an example of one wall in the world. The wall is made by taking a box and elongating it. In line 2 and 12 we define the pose of the box, which is identical for the collision and the visual tag. Line 5 and 15 define the size of the box, but here the third parameter, the height of the wall, is 2.5 for the collision tag, meaning that the sensor will catch the wall, but in the visual tag the height is only defined as 0.01. This is done to give it a more two-dimensional look in the simulation, to resemble a floor plan view as much as possible. Line 18-23 select the looks of the wall from a predefined selection of materials.

Listing 4.1: SDF code for a wall

```
1  <collision name='OuterWall_1_Collision'>
2      <pose>0.0 10.0 0 0 -0 0</pose>
3      <geometry>
4          <box>
5              <size>20.0 0.05 2.5</size>
6          </box>
7      </geometry>
8      [...]
9  </collision>
10
11 <visual name='OuterWall_1_Visual'>
12     <pose>0.0 10.0 0 0 -0 0</pose>
13     <geometry>
14         <box>
15             <size>20.0 0.05 0.01</size>
16         </box>
17     </geometry>
18     <material>
19         <script>
20             <name>Gazebo/Bricks</name>
21             <uri>file://media/materials/scripts/gazebo.material</uri>
22         </script>
23     </material>
24 </visual>
```

Identically, we build the rest of the walls and end up with a world, shown in Figure 4.2. The four coloured circles are used later for the experiment and are defined by only using a visual tag and not a collision tag. The robot is in the middle of the green square.

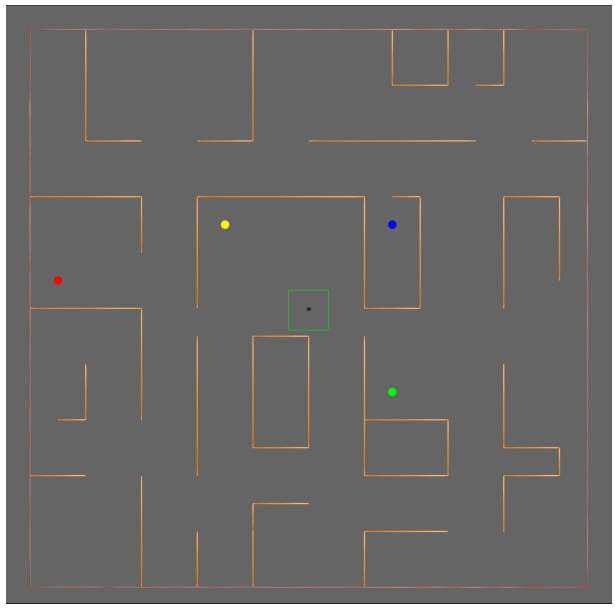


Fig. 4.2: Entire World

Map

After creating the Gazebo world, we use the Simultaneous Localization and Mapping (SLAM) algorithm to create a map from the existing environment. A map is used for extracting information about the position of walls.

The SLAM algorithm is creating a map of the unknown environment while simultaneously keeping track of an agent's location within it. For the purpose of this experiment we have used a preexisting SLAM algorithm, which was implemented and described on Link to SLAM.

The obtained map is shown in Figure 4.3. It does contain imperfections, which are caused by the imprecision of the position estimation of the robot. The larger the map, the bigger the imperfections. Black pixels represent walls, white pixels represent free space and grey pixels represent unknown areas, which have not been properly mapped.

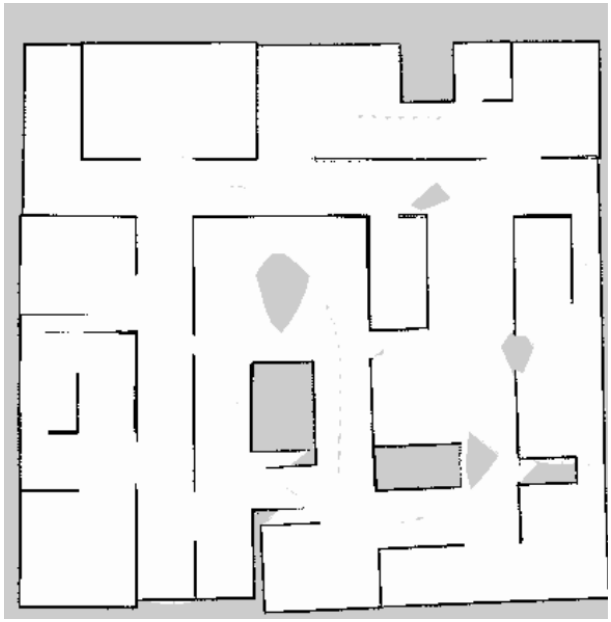


Fig. 4.3: Map of the environment

Obstacle Course

First we build an obstacle course for the test round, where the obstacles are visible, shown in Figure 4.4. For the actual experiment we needed to build a map with invisible obstacles, which the participants need to navigate through. We did this by adding boxes in the world but only with the collision tag and without the visual tag. This way, the scanner still senses the obstacles and the robot could drive into them, but they are invisible to the participant. The world with the obstacles visible is shown in Figure 4.5a and how the participant will see it with invisible obstacles is shown in Figure 4.5b.

4.2 Control of the robot

The operator needs to be able to control the robot in the simulation. Therefore, we need to connect an external controller to the system and send the input signals to the robot. One way that the robot could be controlled is with the keyboard. There is already an existing package making this possible, using keys to increase and decrease the linear and angular velocities. This code is here. However, we found that this way of controlling was not optimal, since it does not allow for instant changes in speed/direction and the

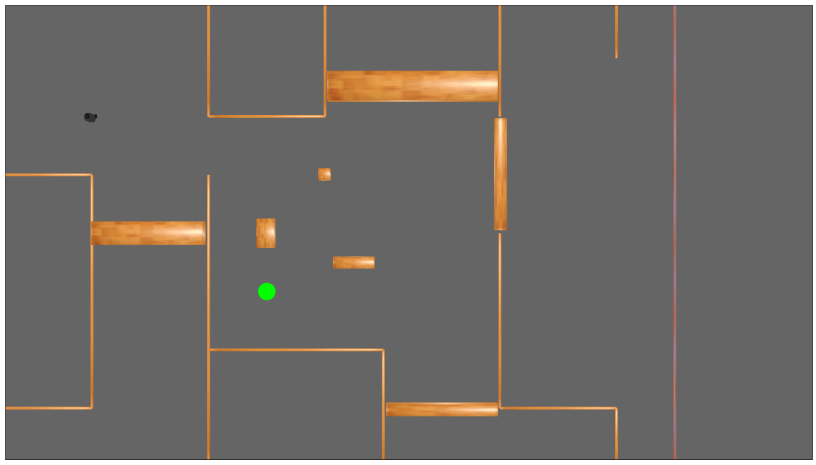
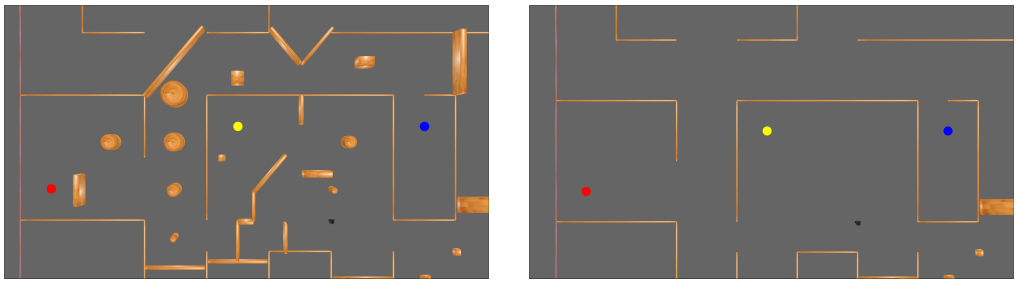


Fig. 4.4: Test round with visible obstacles



(a) World with visible obstacles

(b) World with invisible obstacles

Fig. 4.5

velocity is not relative to the strength of the input. The last point means that input on keys are binary, either it is pressed or not pressed, but it is not possible to press it harder to send higher velocities, just as the robot does not stop once you let go of the key. Changing velocities incrementally can lead to accidental overshoot, since corrections are longer to take effect.

Therefore, we decided to use a PlayStation 3 (PS3) controller for the operator input. The PS3 controller offers the advantages that the keyboard could not. The input to the robot is relative to the strength that the control knob is pushed in the correlating direction and the change in input is immediately transferred to the desired velocity, e.g. the robot stops when the knob is released. There is a preexisting package for giving input commands with a PS3 controller. This package is found here. We made small

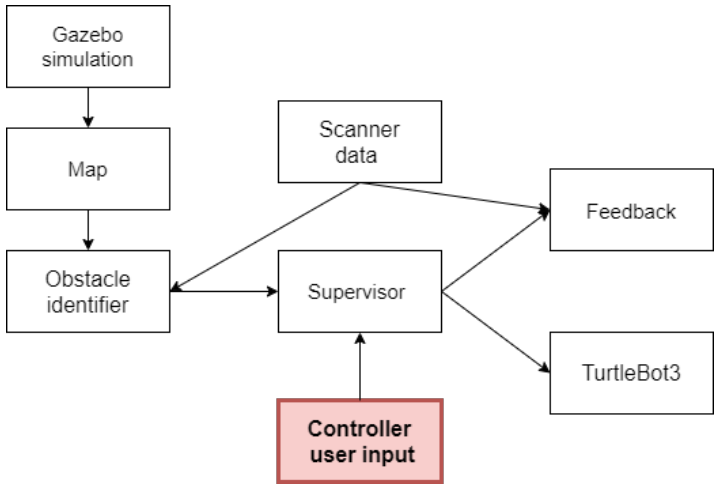


Fig. 4.6: System flowchart

modifications to this package.

- Instead of publishing the controller input directly to the robot, we publish to a differently named topic. The supervisor system will subscribe to this node and process the input. This is necessary to do, since it is not always the operator who controls the robot, but at times the system itself. Therefore, the input needs to go through the supervisor who then relays it to the TurtleBot3 if needed.
- We define a subscriber, which subscribes to a topic to which the supervisor publishes a message. This message contains the current linear and angular velocity limits. If the input to the controller exceeds those limits, we automatically cap the published velocities at the limits.

To sum up, the PS3 controller node reads the input from the operator, checks whether the desired velocities are within the limits, and publishes the input, which is later processed by the supervisor node.

4.3 Obstacle Identification

As mentioned previously, the robot can be in different stages depending on the presence of obstacles:

1. The robot can drive freely, there are no obstacles
2. The robot faces a wall, needs to make sure it does not crash

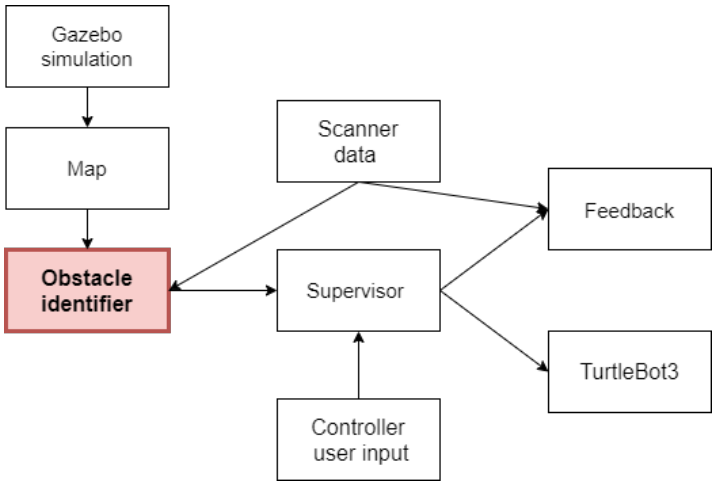


Fig. 4.7: System flowchart

3. The robot faces an obstacle, needs to drive around it

To determine which action the robot should take and which LORA is applicable, meaning who is in control of the robot, it is necessary to identify which type of object the robot is seeing in its scanner, a wall or an obstacle. In this section we describe how the obstacle identification algorithm works.

Robot localisation

After creating the map, the next step is to create an algorithm that will be able to locate the robot in it. This means that when the operator moves the robot in the Gazebo simulation, we will be able to determine its current location on the map. The robot's current location is published in the odometry topic, which contains an estimate of the position and velocity of the robot in free space. An example of the odometry message is shown in Figure 4.8. To access those information we subscribe to the odometry topic and store information regarding the x and y position in variables so they can be used throughout the program. Once the current x and y position are known, they need to be translated to the map's coordinate system. To do that, the following equations are used:

$$newX = \frac{sizeX}{2} - positionX * \frac{sizeXmax - sizeXmin}{sizeXworldmax - sizeXworldmin}$$

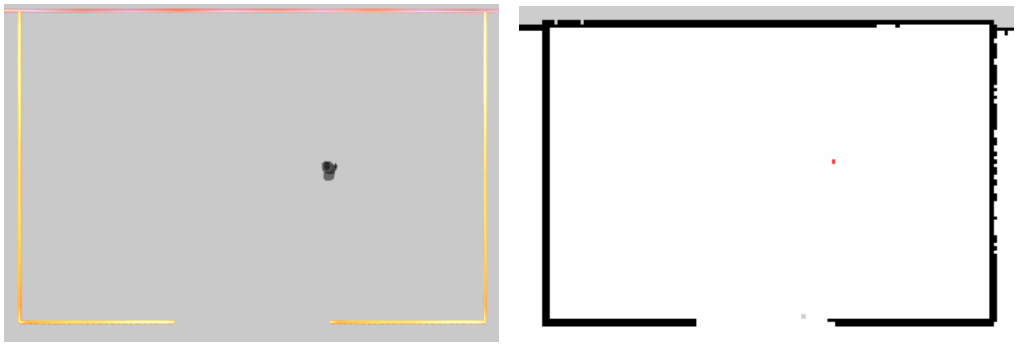
$$newY = \frac{sizeY}{2} - positionY * \frac{sizeYmax - sizeYmin}{sizeYworldmax - sizeYworldmin}$$

```
pose:
  pose:
    position:
      x: 0.0056596175349
      y: -0.0112042226439
      z: -0.00100236770782
    orientation:
      x: 0.00210091403072
      y: 0.00323875320765
      z: -0.545932470151
      w: 0.837820287814
```

Fig. 4.8: Odometry information output in the terminal

Where *newX* is the robot's newly calculated x coordinate on the map, *sizeX* is the size of the map in x dimension, *positionX* is the current robot position in the simulation, obtained from the odometry topic. *SizeXmax* is the maximum value of x in x dimension on the map, whereas *sizeXmin* is the minimum value. In the divisor, *sizeXworldmax* relates to the maximum value of x in the x dimension in the simulation, whereas *sizeX-worldmin* is the minimum value. In the second equation, analogous variables were used for the Y axis.

After calculating the new position of the robot, it can now be correctly localised on the map, which is shown in Figure 4.9.



(a) Location of the robot in the simulation

(b) Position of the robot on the map (represented by red pixel)

Fig. 4.9: Position of the robot in the simulation and on the map.

Region of interest

Once the position of the robot is known, its region of interest needs to be defined. By region of interest we define everything that is within the range of the scanner. The scanner on the TurtleBot3 has an angular range of 360° and a distal range of 3500 mm.

We define a function whose task it is to cut out a fragment of a map in the shape and size of the scanner range.

We do that by superimposing a mask over the image; the function takes four parameters as input: *shape*, *center*, *radius*, and *angular range*. *Center* is defined as the robot's position, which was calculated in Section 4.3. *Radius* is the range of the scanner scaled down to the scale of the map, according to equation

$$radius = \frac{mapSize}{simulationSize} * scannerRange$$

where *mapSize* and *simulationSize* are the size of the map and simulation and *scannerRange* is the distal range of the scanner in the simulation.

This means that for our simulation and map sizes, the scanner range will be represented on the map as a circle with a 720 *pixel* radius. In this case the scanner has an angular range of 360°, but the algorithm can work for other angles as well.

The final output of superimposing the mask over the map for an angular range of 360° is shown in Figure 4.10.

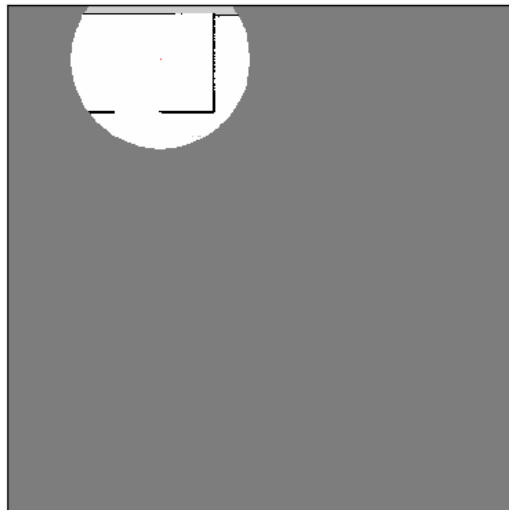


Fig. 4.10: Scanner range on the map

Obstacle identification

The goal of the obstacle identifier is to differentiate what type of objects are found in the scanner range, walls or obstacles. We can divide the possible findings of the scanner into three groups:

1. Robot detects something in the simulation, which also exists on the map - we consider that to be a wall.
2. Robot detects something in the simulation that does not exist on the map - we consider that to be an obstacle
3. Robot does not detect any objects in its range

To access information from the robot's scanner readings, we subscribe to the LaserScan topic. The LaserScan message consists of an array called *ranges*, which contains the measured distances to the closest object at each angle. The whole array consist of 360 elements, for the 360° angular range. Elements in the array can reach values from 0.12 m to 3.5 m or be infinite, in case there is no object in range.

Once all the distances to the objects are known, we can try to superimpose them on the map to find their corresponding coordinates in the map coordinate system. We do this, so we can later compare whether we can expect an object at those coordinates according to the map.

Since the angle, distance, and the origin (i.e. the robot coordinates) are known, the object coordinates can be found through the following trigonometric equations:

$$x = robotX + (radius * cos(angle))$$

$$y = robotY + (radius * sin(angle))$$

Where x is the x-coordinate of the object in the map, *robotX* is the position of the robot in x dimension. *Radius* is the distance to the object, obtained from the LaserScan message and *angle* is the angle at which the object is located. For the second equation, to calculate the y-coordinate of the object in the map, we use corresponding variables.

However, these equations only apply if the robot is not rotated in any direction. If the operator starts rotating the robot, it becomes necessary to take this angle into the equation as well. The robot rotation is shown in Figure 4.11. Information about the robot's rotation is stored in the odometry message. However, those values are represented in Quaternions and need to be transformed into Euler angles.

Once the Euler angles are calculated, it can be used to find the angle of the object while accounting for the rotation of the robot. This is done with the following equation

$$angle_{New} = (rad(360 - LaserScanAngle - EulerAngle))$$

(a) Robot not rotated **(b)** Robot rotated by 45°

Fig. 4.11: Robot rotation

where $angle_{New}$ is the angle at which the object is located, *LaserScanAngle* is the angle obtained from the LaserScan message and *EulerAngle* is the newly calculated rotation angle of the robot.

After solving those equations, we can find the actual x and y coordinates of the object by once more using the following equations with the newly found angle.

$$x_{New} = robotX + (radius * cos(angle_{New}))$$

$$y_{New} = robotY + (radius * sin(angle_{New}))$$

This calculation is repeated for each element from the 360 element array.

Two examples of calculating an object coordinate in the map, are shown below. In Figure 4.12, we see the robot in the middle of the map. In front of the robot is an obstacle, the cardboard box, behind the robot and on its sides are walls. Figure 4.13 shows the calculated object position at an angle of 0°, represented by the green dot, and the robot as the red dot below it. Black pixels are walls in the map and grey pixels are unknown areas, which have not been properly mapped. The figure shows that the robot detects the cardboard box in front of itself at 0°. Figure 4.14 shows the case of scanning at 180°. Here the robot detects the wall behind itself, represented as the green dot, just above the black pixel of the wall.

Once the object coordinates for all 360° are known, we need to compare this information to the map data, to see whether there actually is an object registered on the map at those coordinates.

First, we need to extract information about the walls from the map. We store the position of the walls, which are in the region of interest, in two arrays. One for the x coordinates and the second one for the y coordinates.

We do that by simply saving the coordinates of every black pixel, representing walls, which are located in the region of interest on the map.

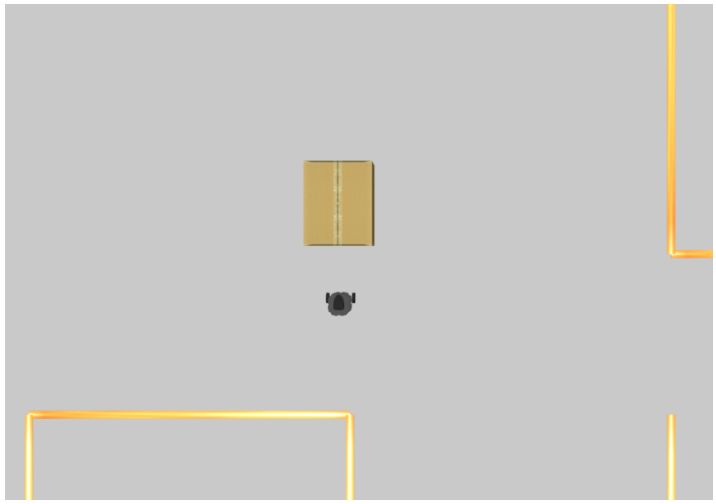


Fig. 4.12: Obstacle placed in front of the robot

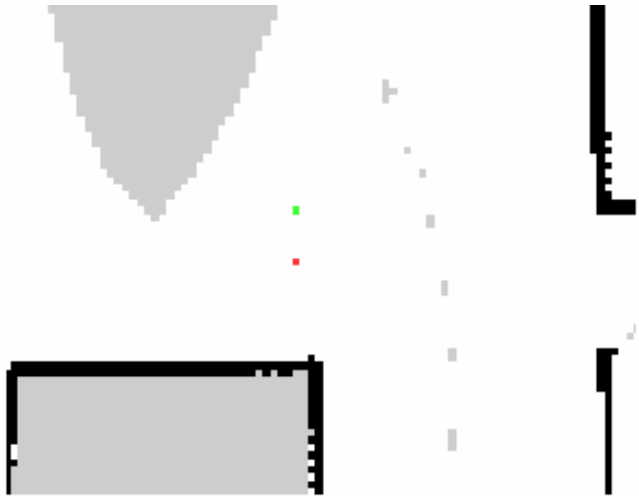


Fig. 4.13: Estimated position of the obstacle - represented by green dot - 0°

Secondly, we need to compare the data we gathered from the scan with the information from the map. To classify the type of object, we assign integer values to each angle.

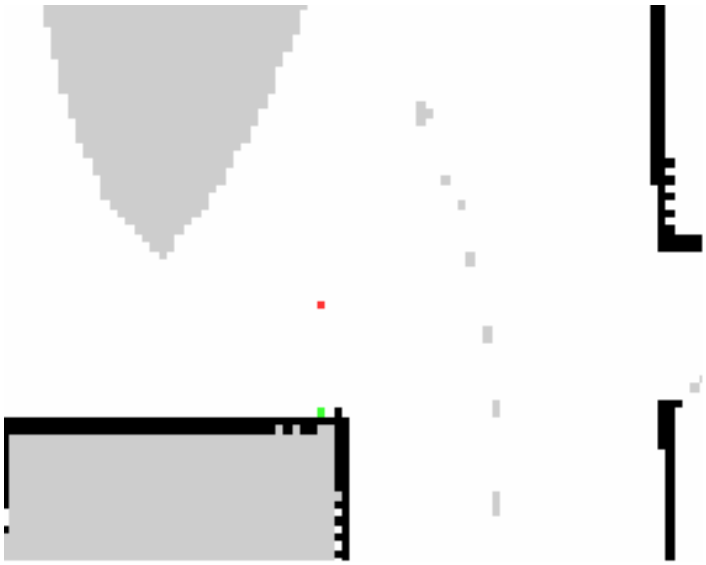


Fig. 4.14: Estimated position of the obstacle - represented by green dot - 180°

- Object is an obstacle → 0
- Object is a wall → 1
- There is no object in the range → 2

To make this classification, we check the following for each of the 360 scan angles.

1. Is the distance to the object infinite? → We assign value **2** to this angle to indicate that there is no object in range.
2. The distance to the object is finite. We compare whether for the calculated x-coordinate of the object there also is a wall, which has the same x-coordinate. Due to imprecision in the scanner, we accept all values within a 3% range of the target coordinate.
3. If we find a match for the x-coordinate, we do the same for the y-coordinate, again with the 3% range.
4. We find a corresponding wall, matching with both the x- and y-coordinate → We assign value **1** to this angle to indicate that the object is both in the map and the scan, meaning that it is a wall.

5. We do not find a corresponding wall → We assign value **0** to this angle to indicate that it is only in the scan, meaning that it is an obstacle.

After the algorithm is repeated for every value from the 360° scan, the obstacle identifier produces an output of three arrays each containing 360 elements.

- Array containing the scan angles
- Array containing distances to the scanned object
- Array containing classified object type

These arrays are published in a topic, which is subscribed to by the supervisor.

4.4 Supervisor

The central piece of the program is a node, which is basically acting as the brain of the whole program. It collects information from all the other nodes, decides whom to allocate control to and how the robot should behave. This node we call the *Supervisor*.

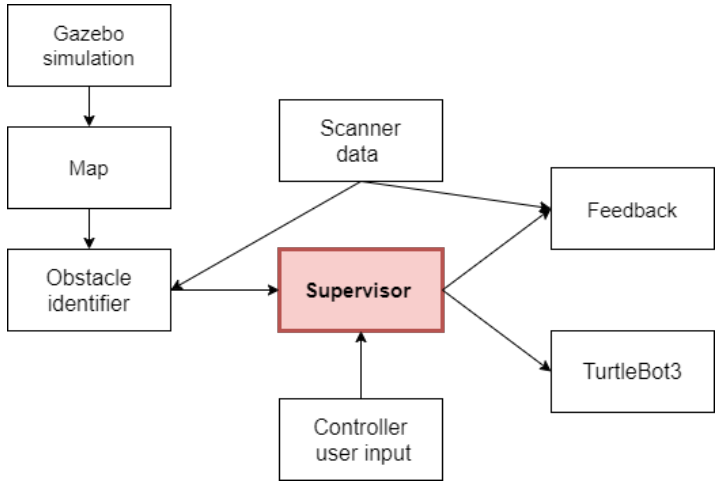


Fig. 4.15: System flowchart

As mentioned previously, the robot can be in two main states, **Free Driving** and **Obstacle Avoidance**. In each of these states there are substates, such as being close to an obstacle/wall, having to avoid crashing into the obstacle/wall, and different obstacle avoidance behaviours, given by the three conditions. Since the robot can be in a clearly defined number of (sub-)states, we have decided to implement the supervisor with a

Finite State Machine (FSM). A FSM can describe a system in a finite number of states, which have clearly defined transitions from one state to another. Each transition between states has a condition attached to it, which needs to be fulfilled to transition to the other state.

Before we get to describe how the robot behaves in each state, we first explain what the different states are and how it is decided which state should be active at what point.

Finite State Machine

The concept behind choosing the appropriate behaviour for the robot at any point in time, is based on driving freely, avoiding visible walls, avoiding invisible obstacles, and making sure that the robot crashes into neither of them. Therefore, it seems most appropriate to look at the distances towards walls/obstacles, to decide which state the FSM should change to. The sensor provides us with distances to objects and, as shown above, the system can determine whether these objects are walls or obstacles.

One of the main states that the robot can be in is **Free Driving**. The free driving state should be active when the robot is neither close to a wall nor to an obstacle.

The next state is the case that the robot is getting closer to an object, independent of whether it is a wall or an obstacle. In this case we want to gradually reduce the speed of the robot, the closer it gets to the object. Therefore, we call this state **Buffer Zone**.

The other main state is that the robot has to perform **Obstacle Avoidance**. This state needs to be in action if the robot gets into a certain distance of the obstacle. Due to the test conditions, we have three different states for this, depending on which avoidance behaviour we want to induce. These three states are **Feedback Assisted Teleoperation**, **Shared Control**, and **Full Autonomy**. Of course each participant in the experiment can only get the system into one of these states.

When the robot gets too close to a wall or an obstacle we need to perform an emergency break and push the robot away from the wall/obstacle, so the operator can again drive freely. This state we call **Safe Zone**.

This gives in total a number of six states, of which three are parallel.

- Free Driving
- Buffer Zone
- Safe Zone
- (Obstacle Avoidance)
  - Feedback Assisted Teleoperation
  - Shared Control
  - Full Autonomy

Something that makes the system more manageable is the fact that in our FSM, the transition conditions (mostly) do not depend on the origin state. This means that, for example, the transition condition to go from state **Free Driving** to **Obstacle Avoidance** is the same as from **Buffer Zone** to **Obstacle Avoidance**. However, there is a slight variation in some of the transition conditions when moving away from the different obstacle avoidance states. This was necessitated by the behaviour of the robot in each condition, i.e. the need for breakouts, and will be explained later.

Distance to nearest object

Initially in the development of the program, we were not yet concerned with differentiating between walls and obstacles and had the first goal of letting the robot drive freely, reduce speed, and stop before hitting an object, meaning implementing the three states **Free Driving**, **Buffer Zone**, and **Safe Zone**. For this, we need to identify the object, which is currently posing the most danger to the robot, so we look at the sensor input and try to find the closest object to the robot. However, since we are only worried about objects, which the robot could realistically hit, we only look at the sensor data in front and in the back of the robot, so different driving directions can be considered. We are not concerned with what is on the robot's sides, since the robot should be able to drive closely parallel to an object. Through trial-and-error we found that a scan angle of 70°, each in front and back, gave good results. Larger angles caught objects, which were not a threat, and smaller angles missed objects. Figure 4.16 shows the scan angles of the robot for detecting the nearest object.

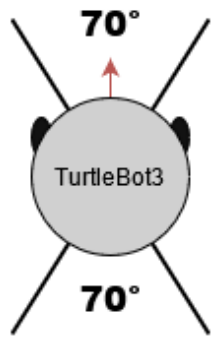


Fig. 4.16: 70° scan angles in front and back

To determine at which distance and angle the closest object is, we individually look at the 70° angles scan from the front and the back. For each data point we have, we compare whether it is smaller than the previously detected minimum. If that is the case we save it as a new minimum.

This way the system determines the distance and angle to the closest object both in front and in the back at an angular range of 70°.

In the next step, we need to determine values for how far from an object the robot was allowed to be before it should reduce the speed and how close it could be to an object before performing an emergency break. Through trial-and-error we settled with the values 0.8 m for the **Buffer Zone** state and 0.2 m for the **Safe Zone** state. This means that

1. the robot can drive freely if the distance to the closest object is larger than 0.8 m $\rightarrow$ **Free Driving**
2. the robot reduces its speed if the distance to the closest object is smaller than 0.8 m and larger than 0.2 m $\rightarrow$ **Buffer Zone**
3. the robot performs an emergency break and pushes itself back into the buffer zone if the distance to the closest object is smaller than 0.2 m $\rightarrow$ **Safe Zone**

Figure 4.17 shows a drawing of the different zones related to the distance to an object. However, one needs to remember that the distances only account for objects in front and in the back of the robot. Thus the robot could very well be in the buffer zone but parallel to the object and, therefore, in the **Free Driving** state. This makes it very difficult to visualize and the figure is mainly thought to give an approximate idea of how the system works. Now that we have established the different states and the transitions,

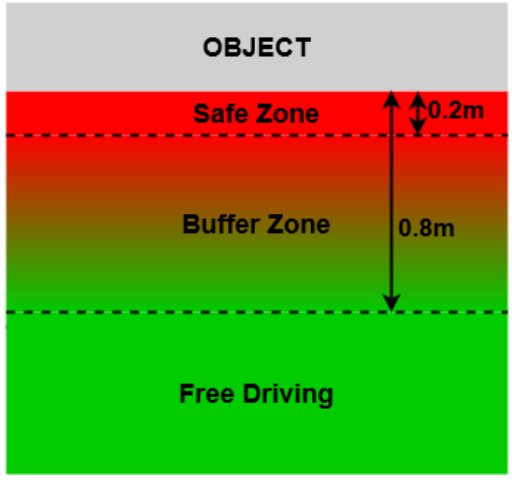


Fig. 4.17: Overview of different zones

and before we consider the presence of obstacles, we will first explain what happens in the individual states.

Free Driving State

In this state there is no object in near proximity of the robot. The robot is in the free driving zone. Here the operator has full control over the robot, within the predefined linear and angular velocity limits. The supervisor node relays the input from the PS3 controller directly to the robot.

Buffer Zone State

When the robot is closer than 0.8 m and further than 0.2 m from an object, it enters the buffer zone. Here the operator is still in full control over the robot, but the maximum allowed linear velocity gets gradually reduced the closer the robot gets.

To calculate the current speed limit, we first check whether the closest distance to the object is in the back or the front

$$minRange_{total} = min(minRange_{back}, minRange_{front})$$

Furthermore, we use the variables *maxSpeed* and *slowSpeed*, which give the maximum allowed speed in free driving zone and the slowest possible speed at the closest distance to obstacle. We also take the 0.8 and 0.2, which represent the range of the buffer zone in relation to the object. We use the following formula the calculate the current speed limit

$$speedLimit = ((minRange_{total} - 0.2)/(0.8 - 0.2)) * (maxSpeed - slowSpeed) + slowSpeed$$

This new speed limit we continuously publish in a topic that the PS3 controller node subscribes to.

Safe Zone State

When the robot gets within 0.2 m of an object it needs to perform an emergency break and push the robot away from the object so it once again can drive freely.

When implementing this state we realised that we actually profit from splitting this state into three substates, depending on the location of the object. The new states are defined as following

- Object is within 0.2 m in front of the robot → **Safe Zone Front**
- Object is within 0.2 m behind the robot → **Safe Zone Back**
- Objects are within 0.2 m both in front and behind the robot → **Safe Zone Front and Back**

The benefit of splitting it into substates is that each of the cases require different robot behaviour, but the common goal is to move the robot away from this state of safe zone violation.

The behaviour of the system in the newly defined states is as following.

1. **Safe Zone Front:**
   The system allocates control of the linear velocity to the robot and drives slowly backwards. However, the operator has the possibility to increase the velocity by pushing the joystick in the corresponding direction. The system always selects the larger value. The operator keeps control over the angular velocity. This way the operator can help with getting out of the safe zone faster.

2. **Safe Zone Back:**
   The system does the same thing as in the previous state with the difference that it drives forwards.

3. **Safe Zone Front and Back:**
   The system takes full control over the robot. Linear velocity is set to 0.0 $\frac{m}{s}$ and the angular velocity is set to 0.5 $\frac{rad}{s}$ to turn the robot. As soon as one of the objects no longer appears in the front or back scan, the system will automatically move to a different state.

After implementing these states, we test the system to check whether it all functions as planned. We find that the robot can be driven in the simulation with the PS3 controller, that it reduces the speed as planned and that it performs emergency breaks reliably.

This way all the states are covered, given that obstacles are not considered independently. In the next step, we need to take into account how the system should switch between the states if obstacles are treated differently than walls.

Potential Field Method

When the robot senses and identifies an obstacle, one of the three different **Obstacle Avoidance** substates needs to be activated, depending on the test condition. Our initial approach was to use the same distance checking method as explained before, but we realised that this would only help the system move into the obstacle avoidance state, but not help with the actual obstacle avoidance itself. Of course, we could implement a separate system, which would take care of obstacle avoidance (or a degree hereof, depending on the autonomy mode), but we thought that it was beneficial if it was combined in one system. Therefore, we developed a system inspired by potential field methods.

Potential field methods are used in robot navigation to calculate optimal goal paths

for robots. The basic idea is to use attractive and repulsive forces, which always steer the robot in the best direction towards the goal. A goal would be assigned a high attractive force and obstacles a high repulsive force. This way, the robot would be attracted to move towards the goal and be repulsed by the obstacles, thereby avoiding collisions. The closer the robot would get to the obstacle, the stronger the repulsive force would be. The whole concept resembles magnets with their polarities. Goals can be represented as plus poles and obstacles as minus poles. The robot itself, if seen as one unit, would also be a minus pole.

In our scenario, there is no given goal that the system is aware of and there is no existing map of obstacles, only of the walls but they should not induce any obstacle avoidance behaviour. This raises of course the question whether potential fields are at all applicable in this case. We found that we could take the concept of polarity out of the global path planning scheme and use it for local obstacle avoidance. Thus, the robot will not look at the entire potential field and calculate a path towards a goal, but only react locally to obstacles. To do this, we take the distance towards the obstacle and calculate an attraction/repulsion value in relation to the robot.

We changed and improved the algorithm over several iterations.

Iteration 1

First we had to decide at which distance to an obstacle the obstacle avoidance state should be induced. The scanner has a range of 3.5 m but it does not make sense to already react to obstacles, which are so far away. Through trial-and-error we found that an acceptable solution was to look at all obstacles, which are closer than 0.4 m and further away than 0.2 m. This way we add another zone within the buffer zone, described above, which started at 0.8 m.

This way, when the robot gets close to an object that it identified as an obstacle, it will first enter the buffer zone and reduce the speed, then enter the obstacle avoidance zone and execute the avoidance behaviour according to the autonomy mode and then possibly enter the safe zone.

Figure 4.18 shows the updated top view with the different zones and distances including the obstacle avoidance zone. In the following iterations we put more focus on obstacles in the front and back of the robot, therefore the figure is, as mentioned with the buffer zone, only a visual thinking aid.

Iteration 2

In the previous iteration we defined the distance in which our scan values need to be to have an effect on the robot. As the next step, we need to process these values to create a repulsive force from the obstacle, which gets stronger the closer the robot gets to the obstacle. For each scan value x within the distance, i.e. $distToObst_x < 0.4$, we

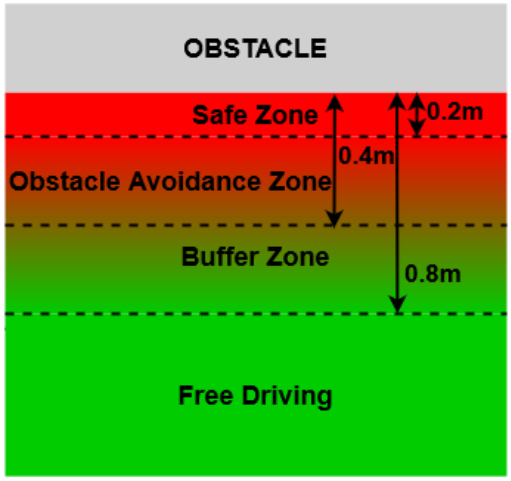


Fig. 4.18: Overview of different zones

calculate a value with the following equation.

$$x_i = e^{(e^{(0.4 - distToObst_x)})}$$

By taking a double exponential function we put more weights on smaller distances, meaning if the robot is very close to an obstacle it gets even more repulsed than with a single exponential function.

Afterwards, we sum all values into a collected repulsive sum r, given by equation:

$$r = \sum_{i=0}^{n} x_i$$

n represents the number of scan values which detect an obstacle in the range of 0.4 m, with $360 \geq n \geq 0$.

This function gives us an increasing repulsion value with decreasing distance to the obstacle, just as intended. However, as mentioned previously, we are not interested in repulsing the robot if it drives parallel to an obstacle, or if it drives forward and the obstacle is behind it.

Iteration 3

To differentiate between the direction in which the obstacle is relative to the robot, we could look at each of the 360 values individually and evaluate whether that angle should be repulsive and with what strength. This is not feasible though, so we decided to separate the 360° angular scan range into eight chunks of 45° each. This is shown in

Figure 4.19. The numbers in the chunks are the indices by which we refer to the chunks and the red arrow points into the driving direction of the robot.

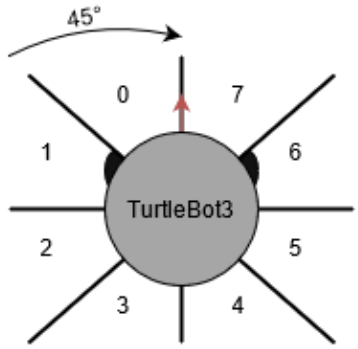


Fig. 4.19: Scan angles split into 8 chunks of 45° each

Then we calculate eight individual distance sums for each chunk, which gives us:

$$r_{0-7} = \sum_{i=0}^{m} x_i$$

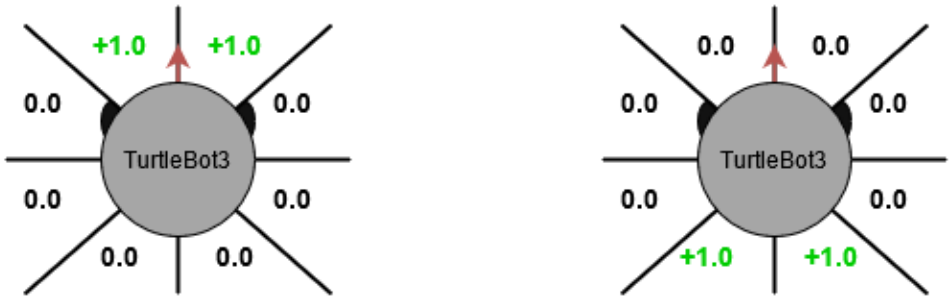
where x_i are distances at all angles in the chunk and $45 \geq m \geq 0$. Next we assign weights $w_0 : w_7$ to each of the eight chunks, to change how much we should take that individual direction into consideration.

To choose the individual weights for each chunk we first only consider driving direction and disregard all other directions. This means that for driving forwards we set the weights for the two front facing chunks w_0 and w_7 to 1.0 and the rest to 0.0, as shown in Figure 4.20a.

For driving backwards we do it inversely and set the weights for back facing chunks w_3 and w_4 to 1.0 and the rest to 0.0, as shown in Figure 4.20b. Positive weights, i.e. repulsive forces, we mark with green numbers, negative weights, i.e. attractive forces, we mark with red numbers.

We determine the driving direction by looking at the current input from the PS3 controller. If the linear input velocity is positive, we set the driving direction to forwards, if it is negative we set it to backwards.

Currently, the system is only determining a value for the potential field but it is not used for anything yet. The overall goal is to avoid obstacles and to avoid obstacles it is best to not be facing an obstacle. Thus, it is good to turn away from the obstacle when the robot is facing it. We concretize this "facing an obstacle" in mathematical terms by



(a) Front chunks are weighted **(b)** Back chunks are weighted

Fig. 4.20: Chunks are weighted depending on driving direction

calculating the sum of all individual chunk values, which we call *collectedChunk*. This value is the sum of each of the eight weighted chunks,

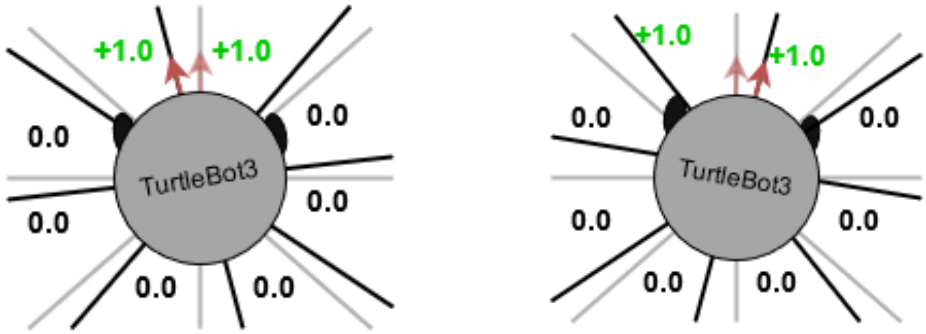
$$collectedChunk = \sum_{i=0}^{7} (r_i * w_i)$$

Given that we currently only assigned non-zero chunk weights to the two chunks facing in the driving direction (forward/backward), we assume that *collectedChunk* gives an adequate representation of how much the robot "faces the obstacle". When the robot gets closer to the obstacle, the value increases, if the robot turns away from the obstacle, the value decreases.

Thus, to avoid the obstacle we would simply have to try to decrease the *collectedChunk* value and possibly bring it to 0.0.

This could of course be done by simply driving the robot in the opposite direction until the obstacle is out of reach, but this is not an optimal solution since obstacles also need to be circumnavigated and not only avoided. Therefore, the *collectedChunk* value needs to be decreased by turning the robot. Ideally, the robot will turn until the *collectedChunk* value is 0.0 and it can again drive freely.

The next question that arises is which direction to turn to. Since the map of obstacles is not known, it is impossible to say whether it is more optimal to turn left or right in a global perspective, but locally we can judge whether there is more obstacle on the left or right side of the robot. We do this by slightly rotating the entire set of chunks 10° both to the right and the left and for each direction recalculate the *collectedChunk* value. This way we simulate what the robot would face if it actually had been turned 10° to the left or right. This is visualised in Figure 4.21. The newly acquired values we can call *collectedChunkTurnedLeft* and *collectedChunkTurnedRight*. We then compare the three values and by turning the robot to the direction where the value is smallest, we turn the robot away from the obstacle as much as possible.



(a) Robot slightly rotated to the left

(b) Robot slightly rotated to the right

Fig. 4.21: Rotation of robot by 10°

This process is repeated until the current *collectedChunk* value has reached 0.0 or is below a threshold, which seems optimal.

Iteration 4

To test the effectiveness of the current state of the potential field we let the robot drive autonomously in the simulation. To let the robot drive autonomously, we simply overwrite any input from the PS3 controller and give a constant input for linear velocity. The system sees this input as coming from the operator, so it is also respecting the velocity limits. When the robot encounters an obstacle, the system takes control over the angular velocity and turns the robot away from the obstacle, as described above.

Besides small adjustments in the individual velocities, we found that this system worked well to turn the robot away from an obstacle and to shortly drive along the wall of the obstacle. Still the robot often turned too far away from the obstacle, back into the direction where it came from.

However, a simple obstacle avoidance by turning was not good enough for us at this point. We thought that in a USAR mission it would be even more favorably if the more autonomous modes actually made it possible to completely drive around the obstacle, similar to wall following algorithms.

Iteration 5

To improve the obstacle avoidance algorithm and enabling the robot to also go around an obstacle, we considered whether we could use the sideways facing chunks. Thinking back to the concept of magnets, we came up with the idea to let the sideways facing chunks actually be attracted to the obstacle, thereby ideally keeping the robot parallel to the obstacle, while still turning when the obstacle becomes too prominent in the

driving direction of the robot.

In a first attempt we assigned negative weights of −1.0 to the sideways facing chunks of the robot, as shown in Figure 4.22.

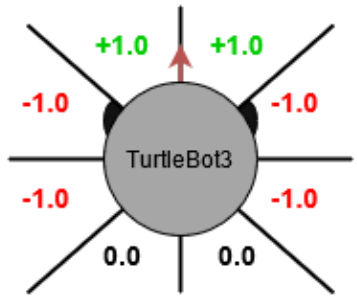


Fig. 4.22: Assigning negative weights to the sides

By introducing negative weights, the *collectedChunk* value can now also be negative itself, meaning that there is a stronger overall attraction towards the obstacle than overall repulsion. The strategy with rotating the chunks slightly still works, with the difference that negative values try to be increased towards 0.0.

This means that when the *collectedChunk* value is positive the system turns the robot to reduce the value, if it is negative the system turns the robot to increase the value. Both sides try to bring the value into an equilibrium at 0.0 where the attraction and repulsion towards the obstacle are balanced. This way the robot should theoretically exhibit a wall following like behaviour.

During tests we found out that the attractive chunk weights of the sideways facing chunks were too large and that the robot would drive too close to the obstacle, repeatedly violating the safe zone.

Through trial-and-error we found that using chunk weights of −0.1 for the sideways facing chunks was keeping the robot in a good balance. This is shown in Figure 4.23.

Iteration 6

While testing the reliability of the wall following obstacle avoidance algorithm, we noticed the tendency of the robot to turn away from the obstacle stronger than we would like. We were not aware of why this happened and changed the chunk weights again, experimenting with making the sides partially repulsive and attractive. Through a mistake in entering the numbers, we accidentally set the chunk weights as shown in Figure 4.24. This actually gave really good results in wall following, with the robot often continuously circling the obstacle before randomly getting too far away from the obstacle, thereby giving control back to the operator.

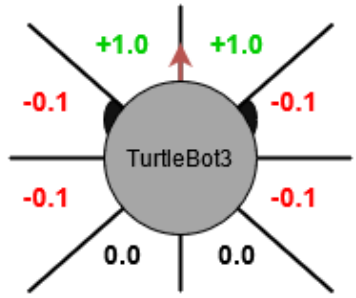


Fig. 4.23: Assigning smaller negative weights to the sides

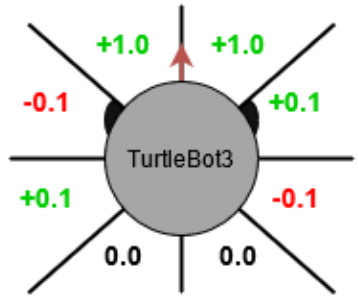


Fig. 4.24: Mixed weights on the sides

However, these chunk weights worked best when driving around the obstacle counter-clockwise. We then flipped the numbers, as shown in Figure 4.25. As expected, this

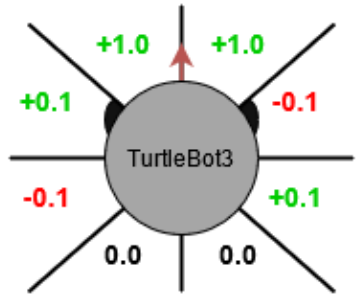


Fig. 4.25: Mixed weights on the sides - Mirrored

configuration favored driving around the obstacle clockwise.

In an attempt to combine both of the well working parts, we set up the system to continuously oscillate between the two configurations, as shown in Figure 4.26. This oscillation would happen every time the control loop gets repeated, which happens around eight times a second.

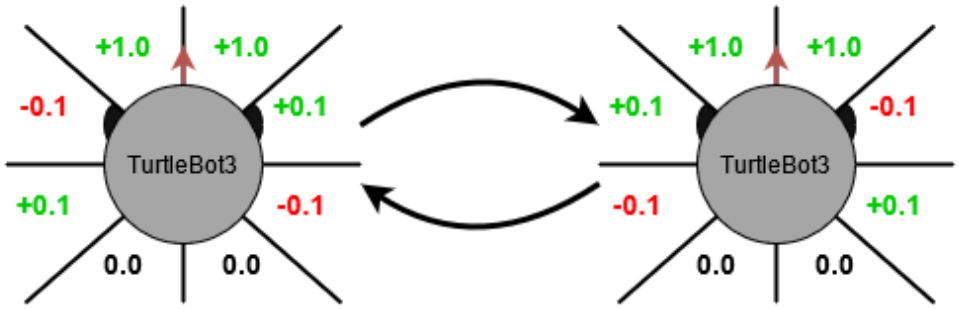


Fig. 4.26: Oscillating between two weight configurations

In this ever changing configuration of attraction and repulsion, we achieved the best results and the robot could follow the walls optimally and drive around obstacles in both directions.

Iteration 7

After implementing the obstacle avoidance algorithm, we needed to define the FSM transition conditions. This means we need to determine at which *collectedChunk* value the FSM switches into an obstacle avoidance state. Through trial-and-error we set the transition condition to $collectedChunk \geq 0.1 \;||\; collectedChunk \leq -0.1$ and based on the *testCondition* we switch to either of the three obstacle avoidance states.

Conversely, the transition condition to the **free driving** or **buffer zone** state check that the *collectedChunk* value is just around 0.0.

These transition values were tested and worked well for the purpose of this project.

Iteration 8

As mentioned previously, while most of the transition conditions in the FSM are identical, there are different transition conditions due to differences in the autonomy modes. However, to explain these properly it is necessary to first understand what happens in each condition.

In the previous Chapter 3, we already defined the behaviours of the system in each of the obstacle avoidance conditions. Hence, there is no big implementation surprises. In the following, we sum up the behaviour of the system in each of the test conditions and add the individual transition conditions for each test condition.

- **Feedback Assisted Teleoperation:**
  The operator is in full control of the robot and can drive it freely at a low velocity. Additionally, the system gives feedback about the position of the robot based on the location of the nearest obstacle. How this feedback is calculated and transmitted to the operator is shown later in Section 4.5.

  In this condition there is no special transition condition to leave the state. When the robot gets too far from the obstacle, i.e. *collectedChunk* value is around 0, the FSM automatically transitions into a different state.

- **Shared Control:**
  The system takes control over the angular velocity and turns the robot away from the obstacle to get the *collectedChunk* value as close to 0.0 as possible. The operator still has full control over linear velocity. The angular velocity is set to 0.34 $\frac{rad}{s}$.

  Here we have to consider the operator's need to use a breakout, as we mentioned earlier. Since the system does not know the overall goal, the robot might just mindlessly drive around the obstacle again and again. Therefore, the operator needs the possibility to break out from the state. This would be a special transition condition for this state.

  To implement this, we continuously check whether the operator pushes the joystick knob to the left or right direction, representing an attempt to change the angular direction of the robot. If this attempt is detected, the system reallocates control to the operator by putting the FSM into the **buffer zone** state.

- **Full Autonomy:**
  In full autonomy the system takes control over both angular and linear velocity and turns the robot away from the obstacle to get the *collectedChunk* value as close to 0.0 as possible, while continuously driving forward with a set low linear velocity of 0.1 $\frac{m}{s}$ and angular velocity of 0.34 $\frac{rad}{s}$.

  Similarly to the previous mode, the operator needs the possibility to break out from this state. Unlike the previous mode, the operator only needs to press the joystick in any direction they like. This is seen as an attempt to break out from the state and the FSM is transitioned to the **buffer zone** state.

Iteration 9

The newly defined transition conditions for breakouts generally worked, however, we encountered a significant problem. As soon as the operator broke out from one of the (semi)-autonomous states, the FSM found that the transition conditions to move directly back into the just exited state were fulfilled. This was due to the fact that the operator did not have time to move the robot out of the reach of the potential function before the robot was "caught" again.

To avoid this, we implemented timers. After the operator performs a breakout, a timer of 5 s is started and until the timer runs out, the system cannot move back into the state that the system has just broken out of. This means that the operator has five seconds to move the robot away from the obstacle and only if the robot is still in reach of the obstacle when the timer runs out, the system moves back into the (semi-)autonomous state.

This principle is also relevant for the other direction. When the operator gets close to an obstacle and the system tries to take over, the operator will most likely not have let go of the PS3 control knob yet, which would mean that the system registers a breakout attempt and gives back control to the operator immediately. To avoid this, an identical timer of 5 s is started when the system takes (partial) control over the robot. Only when this timer has run out, is it possible for the operator to perform a breakout. This should give the operator enough time to realise that the robot has taken over control and release the PS3 control knob.

Of course control can be returned faster if the robot leaves the **obstacle avoidance** zone for "natural" reasons, i.e. it drives away out of the reach of the obstacle by itself. In this case the timers are neglected.

State Feedback

To support the operator in realising that the system has taken over (partial) control, we implemented audio feedback. While the FSM is in either the **Shared Control** or **Full Autonomy** state, the operator will hear a regular constant beeping sound. This is done by playing a system sound at an interval of 0.5 s while one of those states is present and stopping the timer otherwise.

The state of **Feedback Assisted Teleoperation** does not have a state indication feedback, since it has its own audio feedback indicating the obstacle position, which at the same time indicates to the operator that they are in this obstacle avoidance state.

4.5 Feedback

In the **Feedback Assisted Teleoperation** state the operator, when facing a obstacle, will not experience that the system takes over any control. However, the operator still needs to avoid those obstacles to safely reach the goal. In order to do that, the system will give the operator audio feedback regarding the location and proximity to the obstacle. In this section we describe how the audio feedback system works.

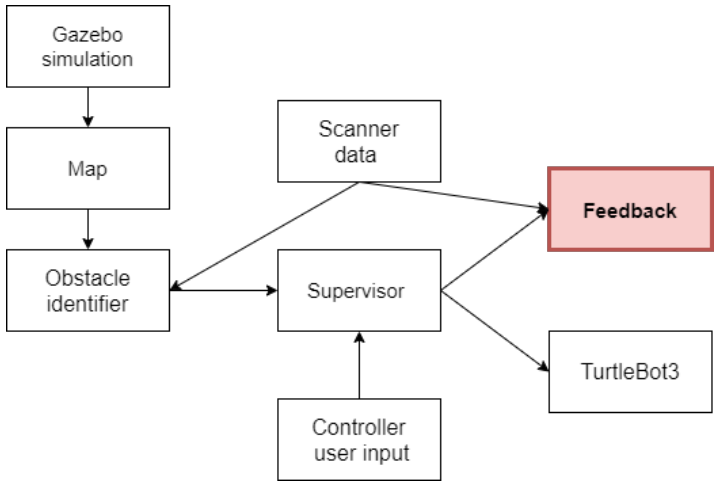


Fig. 4.27: System flowchart

Frequency of audio feedback

We base our algorithm idea on parking assistant systems in cars. Meaning that if the robot gets closer to the obstacle the frequency of the sound will increase, and vice versa - if they drive further away, the frequency will decrease. Therefore, the operator is able to estimate at what distance the obstacle is.

To calculate the beeping frequency, we are using only the distance to the closest obstacle, since this is the most critical one for the robot.

If the robot is driving forward, we are looking at obstacles in the front 90° angles. If the robot is driving backwards, we look at a 90° angle in the back.

Distances to obstacles from those angular ranges are assigned to an array. And among those value we look for the smallest one.

Once the closest distance is known, the frequency of the beeping can be calculated. If the distance is bigger than 0.8 m, the obstacle can be neglected and the system will not make a beeping sound. If the value is smaller than 0.8 m, the obstacle is in close range of the robot and the operator needs to be informed about its presence. While the two (semi-)autonomous modes only get triggered at 0.4 m, we let the feedback assistance in the teleoperation mode already start at 0.8 m. The reason for this is that we felt that the operator needed to get help earlier, since they are in full control all the way and might take longer to react and avoid the obstacle than in the modes where the robot takes over quickly and automatically turns the robot away from the obstacle.

Since we want to make the frequency of the sound dependent on the distance to the obstacle, we have to transform it into a time value corresponding to the distance between the beeps. Smaller distance to the obstacle should indicate a higher beeping frequency, so it would give the operator the information that the robot is getting closer to the obstacle. Thus, we want to increase the frequency with the distance. We use the following formula:

$$FinalFrequency = \frac{weight * e^{(minDistance)}}{100}$$

were *minDistance* is the smallest distance to the obstacle and *weight* is a parameter, which has been set up by trial-and-error method to increase the exponential value. The result is divided by 100 because we want the result to be in hundredths of a second. Once the frequency value is calculated, we can use it to set the time between the beeps.

To do that we use a sleep function, which suspends the execution of the current program for a given time. The function takes the newly calculated *FinalFrequency* as an argument, so it will play a beep sound, wait a certain amount of miliseconds and play the sound again. The function will be executed until the robot exits the Teleoperation mode. The algorithm works in a similar way as a parking assistance, and gives the operator an idea how far away the obstacle might be.

Direction of the sound

In order for the operator to more efficiently avoid the obstacles, they should receive information not only about the distance to the obstacle but also about the direction where the obstacle is placed. During the course of the experiment, the participant is wearing headphones, therefore, we are able to indicate the direction of the obstacle by playing the sound in either the left or right headphone speaker. Figure 4.28 shows the relation between the position of the obstacle and the sound source.

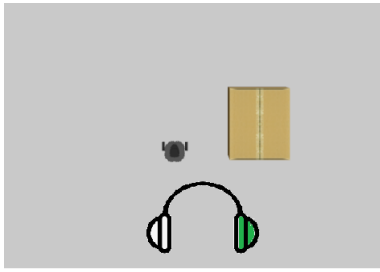
We are only considering obstacles which are placed in 90° angle in front when the robot is driving forward and in a 90° angle in the back, when the robot is driving backwards.

We can divide the forward range into three zones:

1. from 315° to 345° - the obstacle is on the right side
2. from 345° to 15° - the obstacle is located in the middle
3. from 15° to 45° - the obstacle is on the left side

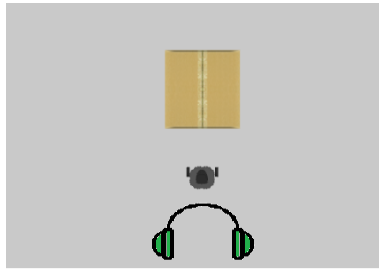
Once the angle of the closest obstacle is known, it can be assigned to one of those zones. Based on the zone, the algorithm will play the sound louder in either the left or right headphone.

To modulate the volume of the sound the pyaudio library has been used. Included in the library is a function, *setVolume*, which can set the volume for both headphone



(a) Obstacle on the right - sound from right speaker

(b) Obstacle on the left - sound from left speaker



(c) Obstacle in the middle - sound from both speakers

Fig. 4.28: Relation between robot's position and sound source

channels independently. The value of the volume ranges from 0 to 1, for both channels. So depending in which zone and what value the angle has, the volume for both channels is set.

Using both of those algorithms to indicate the distance to and the direction of the obstacle, the operator in the teleoperation mode is able to understand where the obstacle lies and to hopefully avoid it.

Chapter 5

Evaluation

In this experiment we want to investigate whether different LORA have an influence on obstacle avoidance in a USAR scenario where the robot operator cannot see the obstacles on the map.

We designed three different autonomy modes as our test conditions.

LORA	Sense	Plan	Act	**Description**
Feedback Assisted Teleoperation	H/R	H	H	The robot senses obstacles and provides feedback, the operator plans a path and acts by fully controlling the robot.
Shared Control	H/R	H/R	H/R	The robot senses the obstacle, plans a way to avoid the obstacle and sets the angular velocity. The operator senses the general map, plans a way to the overall goal and sets the linear velocity.
Full Autonomy	H/R	H/R	R	The robot senses the obstacle, plans the best way to avoid it and fully controls the robot. The operator senses the general map and plans a way to the overall goal but has no authority to enact it.

Table 5.1: Our model of LORA for obstacle avoidance

In the experiment, each participant controls a TurtleBot3 through a simulated world in Gazebo with a PS3 controller. The obstacles are invisible and we collect both objective measurements, like completion time, and subjective measurements, like workload.

Additionally to the measurements, the participants also have to fill out a background survey. In the background survey we ask for the following items

- Age
- Gender
- Subjective Estimation of the participant's skills with a PS3 controller - on a 7-point Likert scale
- Whether they have previously controlled a robot
  - and if yes, how many times approximately

Age and gender are generally included in a background survey to give an idea of what segment of the population the participants come from. The reason we ask for PS3 controller skills and previous experience with controlling a robot, is to see whether the preexisting abilities of the participant influence their ability in succeeding with the experiment.

In this chapter, we start with describing the experimental procedure, the participants, and the way we collected the data. Afterwards, we present our results and discuss what these suggest in relation to our exploratory research question.

5.1 Experimental Procedure

Before the participant enters the room the experimenter makes sure that the experiment is correctly set up and ready to be used.

Then the experimenter welcomes the participant to the experiment and informs them that all safety precautions regarding the coronavirus have been taken. The participant randomly picks one of three pieces of paper, which have the numbers "1", "2", or "3" written on them, to determine the test condition. Afterwards, the participant gets to read the following short introduction to the experiment:

Hello and thank you for participating in this experiment.

If you have any questions or encounter problems before or during the experiment please feel free to ask the experimenter. Be aware that you at any point during the experiment are allowed to quit if you have changed your mind. All personal data will be handled anonymously.

In this experiment you will control a robot in a simulation with a PlayStation3 controller. Before we get started please fill out the short background survey that your experimenter has provided you with.

Then the participant is filling out the following background survey.

Please start by giving us some background information about yourself

Age:

- 18 – 29
- 30 – 39
- 40 – 49
- 50 – 59
- 60 +

Gender:

- Male
- Female
- Other

How good are you at using a PS3 controller ?

1	2	3	4	5	6	7
Very bad						Very good

Have you ever controlled a robot ?

- Yes

  How many times ?
  - 1 – 4
  - 5 – 9
  - 10 – 20
  - More than 20

- No

Afterwards, the participant gets to read the next part of the introduction, which tells about the scenario they will be testing:

Imagine the following scenario:

*"You are a robot operator for search and rescue missions. Your task is to explore certain areas of the partially collapsed house where the victims could be. You see the map showing the walls and the position of the robot. However due to the damage on the house there might be unexpected **invisible obstacles** eg. collapsed walls etc. which you cannot see on the map. Luckily the robot is equipped with a Laser sensor that detects those obstacles."*

Generally, you will experience the robot behave in the following way:

- *If the robot is far away from an obstacle or a wall, you can freely control the robot.*
- *If the robot gets near an obstacle/wall, you will feel the robot reduce its speed automatically.*
- *If the robot gets too close to an obstacle/wall, it will perform an emergency break and drive you a bit backwards.*

*When the robot gets close to one of the **invisible obstacles**, it needs to avoid the obstacle. In this experiment we want to test different ways that the robot can avoid those **invisible obstacles**. The way the robot behaves to avoid it, is what we call **Control Mode**.*

*The **Control Mode** you will experience will make the robot behave in the following way around invisible obstacle.*

Then depending on which condition the participant will be tested on, they read one of the following texts describing the behaviour of the robot in their **Control Mode**, i.e. the behaviour of the robot for obstacle avoidance.

- **Feedback assisted teleoperation:**
  When you get close to the obstacle, you will hear a beeping sound. The beeping sound will show the distance to the obstacle (faster beeping = robot is closer to the obstacle) and the direction the obstacle is in (beeping on the left = obstacle is on the left side of the robot, and vice versa). You can imagine it being similar to the parking assistance sensor in a car.
- **Shared Control:**
  The robot will help you by deciding which way it should drive and automatically turn that way. However, it is still your job to drive forwards/backwards. While

the robot is assisting, you will hear a regular beeping sound. When the beeping stops, it is again you who has to decide the robot's driving direction. If it feels necessary, you can always get back full control by pushing the joystick to either side (left/right).

- **Full Autonomy:**
  When you get close to the obstacle the robot will take over complete control and try to drive around the obstacle for you. While the robot is in full control, you will hear a regular beeping sound. When the beeping stops, it is again you who has to control the robot. If it feels necessary, you can always claim back full control by pushing the joystick in any direction.

Afterwards, every participant gets to read the same text, which continues as:

Be aware: The robot does not know where you want to go, it will only help you with avoiding obstacles.

First you have the opportunity to get familiar with the system by driving through a test scene. Here you can learn how to control the robot and you can experience how the robot reacts around obstacles for your particular control mode. In the test drive, the obstacles are visible on the map, so you can directly aim for them to test the robot's behaviour.

When you are done with the test round and feel familiar with the system, please drive the robot to the green circle.

The experimenter will then prepare the actual experiment scenario for you.

The participant will see the screen shown in Figure 5.1 with the robot starting in the top left corner and the green circle indicating where they should drive the robot when they are done with testing. In this case, the obstacles are visible, so the participant can actively try to drive towards them to experience the way the robot will react. The participant will also put on headphones to be able to hear the audio feedback.

After the participant has controlled the robot to the green circle, the experimenter takes control over the computer again to prepare the actual experiment. At the same time, the participant gets to read the next instructions, which are:

Welcome back. Now you will control the robot in the actual experiment. The robot will behave the same way that it did just before, meaning it will help you avoid obstacles just as you just experienced in the test round. The only difference is, that the obstacles are now invisible.

Your goal is to reach the three areas of interest (marked through coloured circles) as

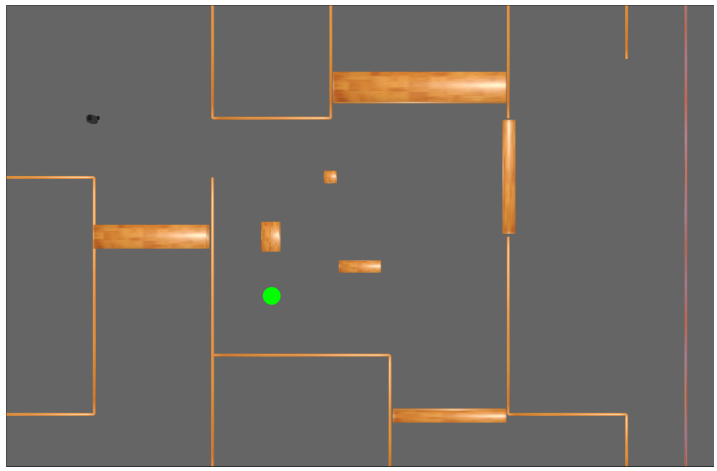


Fig. 5.1: Test Round

fast as possible in the order: YELLOW, RED, BLUE

Good luck :D

Figure 5.2 shows what the participant sees in the experiment. The robot starts in

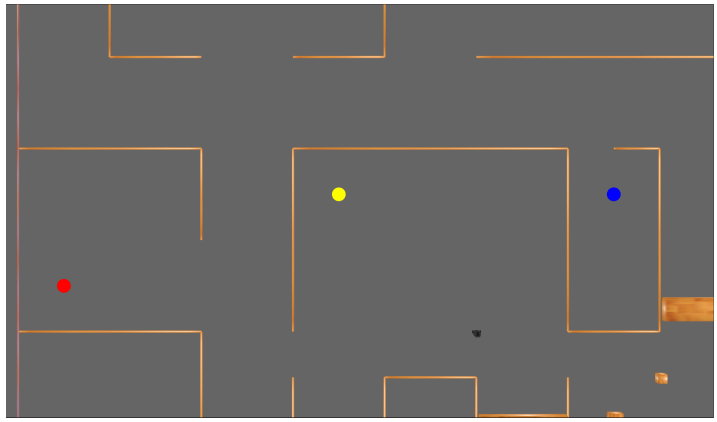


Fig. 5.2: Experiment Round

the bottom right corner and three coloured circles indicate the subgoals. The experimenter starts the stopwatch to measure how fast the participant completes the obstacle

course. Here, the obstacles are invisible, only the walls are shown.

When the participant has reached the last coloured circle, the experiment is done. The participant gets to read the last text, which is:

Good job! We hope you enjoyed the experience!

Please fill out the following questionnaire that your experimenter has provided for you.

Remember there is no wrong or right, just answer how you feel.

Thank you!

If the participant has made any interesting comments about the experience, the experimenter notes them down for future evaluation and inspiration. After the participant has left, the experimenter disinfects the used equipment and prepares the experiment for the next participant. The experimental setup is shown in Figure 5.3

Fig. 5.3: Experimental setup

5.2 Participants

A total of 39 participants (25 male and 14 female) took part in the experiment.

Incomplete data sets due to technical errors, which occurred four times, were completely ignored. Three data sets needed to be ignored due to a technical error in the Teleoperation condition, which sometimes caused the system to crash. The error was

later fixed. One data set was ignored due to the robot getting stuck in the environment to a degree where the participant could not control it anymore.

Additionally, we decided to exclude three data sets, due to the fact that the participants had big problems controlling the robot. The participants failed to comprehend the control mechanism even on the most basic level, therefore, their results would not be comparable with others. After rejecting those data we ended up with 32 data sets. 10 participants were assigned to the Teleoperation condition, and 11 each for the Shared Control and Full Autonomy conditions.

The final set contained data from 32 participants (23 male and 9 female), with 90% in the age group $18 - 29$. The participants had an average subjective PS3 controller skill of 4 and 66% have previously controlled a robot. Of the ones that have controlled a robot, 48% have done it 1-4 times, 14% 5-9 times, 14% 10-20 times, and 24% more than 20 times.

Data Collection

All data was collected from 10.05.2021 to 24.05.2021 in Aalborg, Denmark. Participants were found in the social circle of the authors and from students of Aalborg University.

All data was treated anonymously and participants did not receive any monetary compensation.

5.3 Results

In this section we present the results of the conducted tests.

The collected data can be divided into two categories:

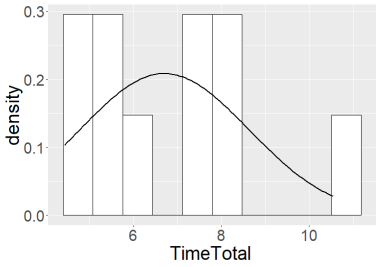
1. Data collected from the experiment
2. Data collected from the questionnaires

Throughout the section the following numbers will be assigned for the testing condition:

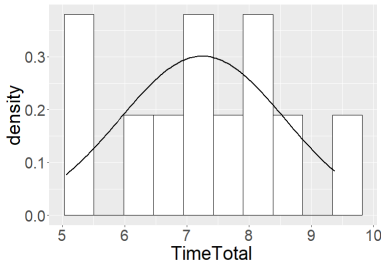
- 1 - Feedback Assisted Teleoperation
- 2 - Shared Control
- 3 - Full Autonomy

Time to complete the task

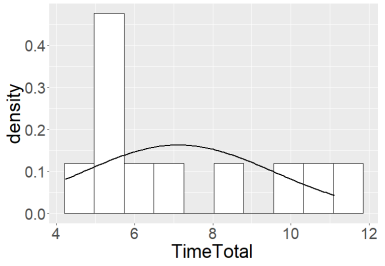
For each participant, we are looking at the time to complete the whole task and its subgoals - the times between each of the coloured circles (start → yellow, yellow → red, red → blue). Time measurements can be treated as parametric, however, they still need to fulfill certain criteria, namely the data has to be normally distributed. Histograms for all three conditions are presented in Figure 5.4.



(a) Teleoperation



(b) Shared Control



(c) Full Autonomy

Fig. 5.4: Total times for all conditions

In order to test if the data is normally distributed, we run the Shapiro test. If the result is non-significant ($p > .05$) it tells us that the distribution of the sample is not significantly different from a normal distribution, therefore, we can assume normality. For condition 1: $W = 0.9153, p = 0.3194$, for condition 2: $W = 0.97845, p = 0.9569$, for condition 3: $W = 0.8867, p = 0.1264$. All of the p values fulfill the condition, therefore, we can assume that our data is normally distributed. Additionally, we run the Levene test, to test the homogeneity of variance; the results $F(2, 29) = 1.8876, p = 0.1696$ show that the data meets all the requirements and can, therefore, be treated as parametric, and parametric tests can be run. The data is visualised in Figure 5.5. The median for condition 1: $(Mdn = 6.58min)$, for condition 2: $(Mdn = 7.27min)$, for condition 3: $(Mdn = 6.4min)$.

Since it is a three-condition experiment, with one independent variable, the one-

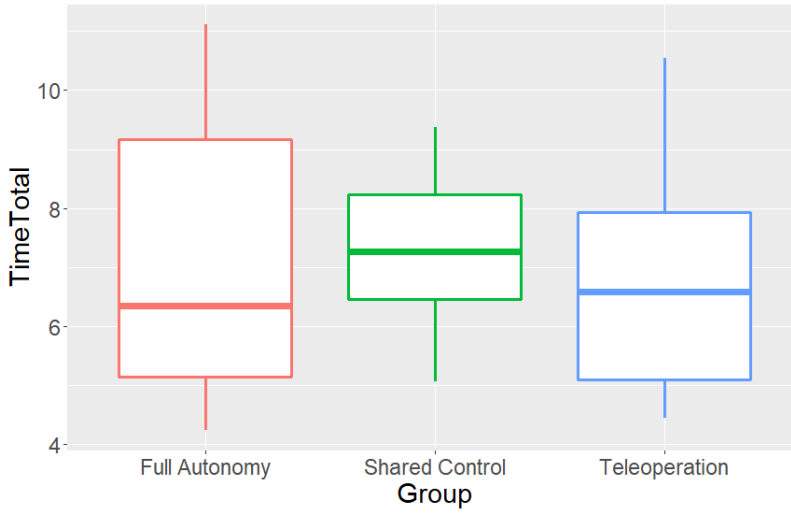


Fig. 5.5: Total time for all three conditions

way ANOVA test is used to analyse the data. The results from the one-way ANOVA $F(2, 29) = 110, p = 0.805$, show that there are no significant differences between the conditions in total time to complete the task.

Time to reach the yellow point

Time to reach the yellow point was measured for each participant, the data is visualised in Figure 5.6. The median for condition 1: $(Mdn = 1.16min)$, for condition 2: $(Mdn = 1.4min)$, for condition 3: $(Mdn = 1.18min)$. As mentioned in the previous section, in order to run parametric tests on our data, it needs to fulfill certain criteria. The histograms for all three conditions are presented in Figure 5.7.

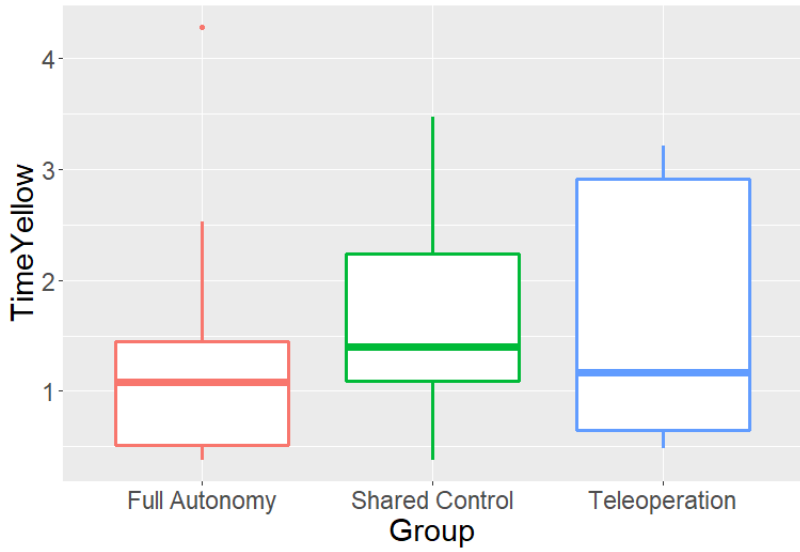
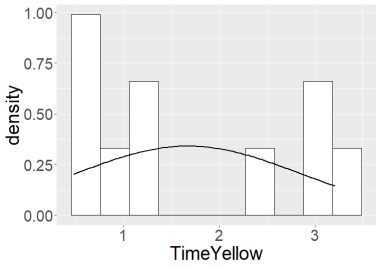
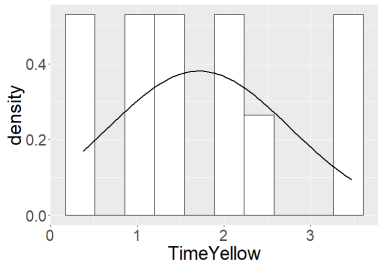


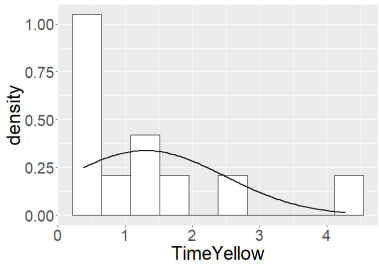
Fig. 5.6: Time to reach the yellow point for all three conditions



(a) Teleoperation



(b) Shared Control



(c) Full Autonomy

Fig. 5.7: Times to reach the yellow point for all conditions

After running the Shapiro test, the results for condition 1: $W = 0.82089, p = 0.02598$, for condition 2: $W = 0.92796, p = 0.3906$, and for condition 3: $W = 0.77122, p = 0.003938$ show that not all data meets the conditions for being normally distributed. To deal with the problem of our data not being normally distributed, we can try to transform the data, so it could meet the criteria. The transformation that is used, is the log-transformation, which replaces each variable x with a log(x). The results from the Shapiro test on the transformed data are, for condition 1: $W = 0.85609, p = 0.0686$, for condition 2: $W = 0.90916, p = 0.2384$ and for condition 3: $W = 0.93037, p = 0.4146$. The p values in all conditions are bigger than 0.05 which means we can assume normality.

The results from the Levene test $F(2, 29) = 0.1461, p = 0.8647$ show that the data fulfills all criteria and can be treated as parametric. Since it is a three-condition experiment, with one independent variable, the one-way ANOVA test is used to analyse the data. The results $F(2, 29) = 17.354, p = 0.523$ show that there are no significant differences between the conditions.

Time to reach the red point

Time to reach the red point was measured for each participant, the data is visualised in Figure 5.8. The median for condition 1: $(Mdn = 1.33min)$, for condition 2: $(Mdn = 1.47min)$, for condition 3: $(Mdn = 1.47min)$. We check if data is normally distributed,

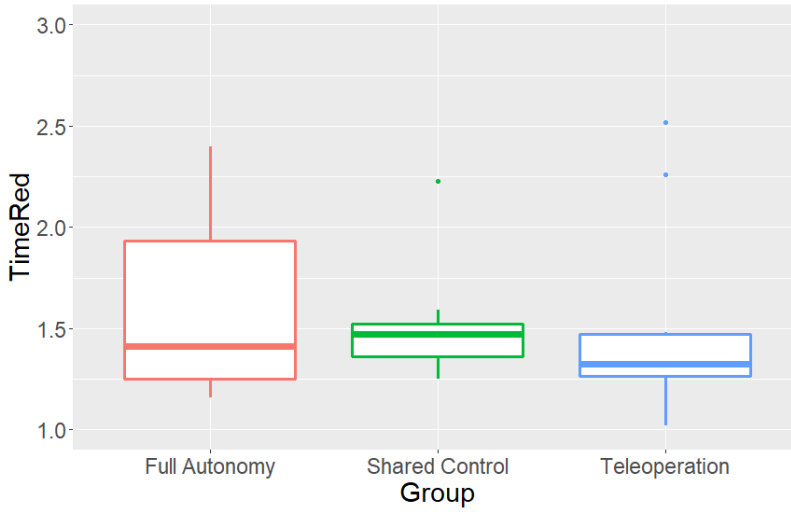
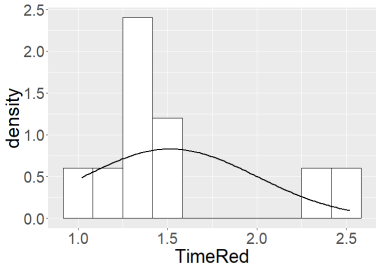
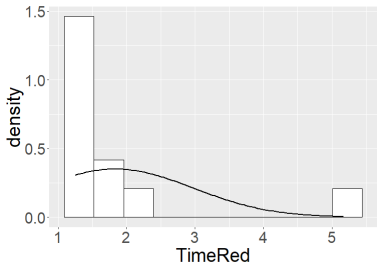


Fig. 5.8: Time to reach the red point for all three conditions

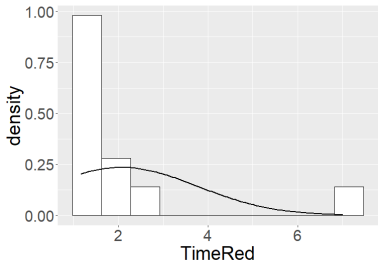
the histograms for all conditions are shown in Figure 5.9.



(a) Teleoperation



(b) Shared Control



(c) Full Autonomy

Fig. 5.9: Times to reach the red point for all conditions

Results from the Shapiro test for condition 1: $W = 0.766, p = 0.00559$, 2: $W = 0.51892, p = 2.568e-06$, 3: $W = 0.5634, p = 8.968e-06$, confirm that the data is not normally distributed.

The data is transformed with the log transformation, and the Shapiro test is run on the transformed data. The results for condition 1:$W = 0.83448, p = 0.03788$, 2: $W = 0.64521, p = 9.374e-05$, 3: $W = 0.75688, p = 0.002556$, show that data is still not normally distributed.

Another transformation we can use to transform the data is the square root, which replaces original data with their square roots transformation. The results from the Shapiro test on the transformed data for condition 1: $W = 0.80039, p = 0.01466$, 2: $W = 0.57818, p = 1.364e-05$, 3: $W = 0.65789, p = 0.0001356$ show that the transformed data is still not normally distributed.

The last transformation that is used, is the reciprocal transformation, which replaces the original data units with their reciprocals. The results from the Shapiro test on the transformed data for condition 1: $W = 0.89219, p = 0.01795$, 2: $W = 0.78056, p = 0.005222$, 3: $W = 0.89377, p = 0.155$, also shows that the transformed data is not normally dis-

tributed.

Since the conditions for parametric data are not met, data is treated as non-parametric.

For it is a three-condition experiment, with one independent variable, the Kruskal-Wallis test is used to analyse the data.

Results from the Kruskal-Wallis test $H(2) = 1.4238, p = 0.4907$, show that there is no significant difference between the conditions.

Time to reach the blue point

Times to reach the blue point was measured for each participant, the data is visualised in Figure 5.10. The median for condition 1: $(Mdn = 2.84min)$, for condition 2: $(Mdn = 3.13min)$, for condition 3: $(Mdn = 3.02min)$. We check if data is nor-

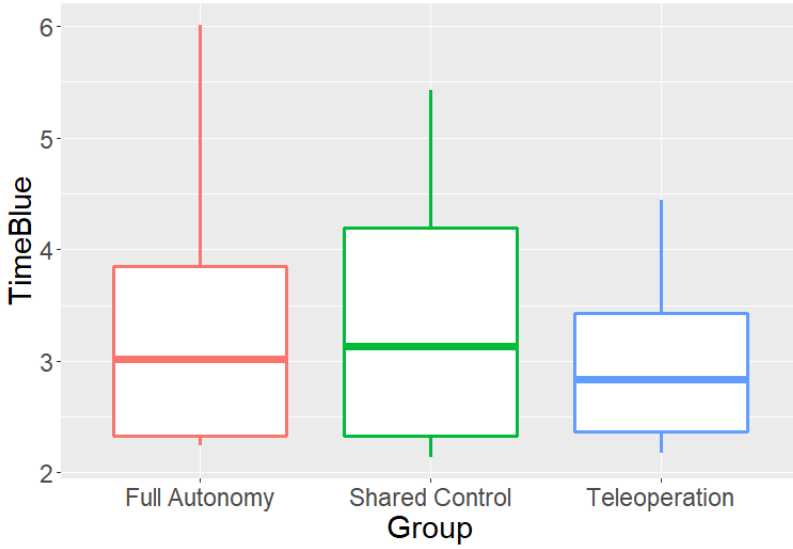
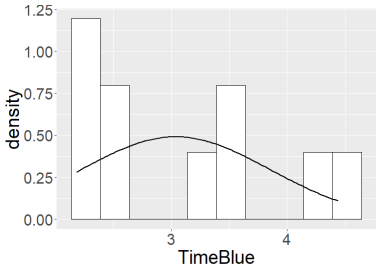


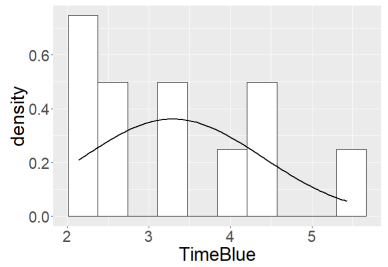
Fig. 5.10: Times to reach the blue point for all conditions

mally distributed, the histograms for all conditions are shown in Figure 5.11. The results from the Shapiro test for condition 1: $W = 0.87941, p = 0.1285$, condition 2: $W = 0.88965, p = 0.1377$, and for condition 3: $W = 0.83381, p = 0.0262$, confirm that the data is not normally distributed. We repeat the process of transforming the data presented in the section above.

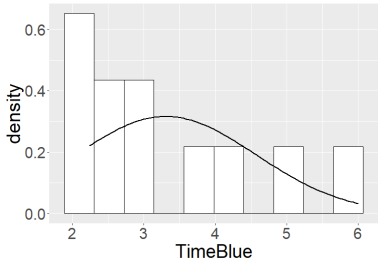
The results from the Shapiro test on the data transformed with log transformation, for condition 1: $W = 0.89386, p = 0.1873$, 2: $W = 0.90623, p = 0.2199$, 3: $W =$



(a) Teleoperation



(b) Shared Control



(c) Full Autonomy

Fig. 5.11: Times to reach the blue point for all conditions

$0.87433, p = 0.08809$, show that the transformed data is normally distributed. Results from the Levene test $F(2, 29) = 0.3258, p = 0.7246$, show that we can assume homogeneity of variance and, therefore, can treat the data as parametric.

Since it is a three-condition experiment, with one independent variable, the one-way ANOVA test is used to analyse the data. The results from the one-way ANOVA test $F(2, 29) = 2.8523, p = 0.878$ show that there is no significant difference between the groups.

QUEAD

Data from the QUEAD questionnaire is treated as non-parametric data, since neither the items or the scale are objective. We run tests not only on the mean score of all 16 items per participant, but also on each of the five subcategories.

Since the data is non-parametric and it is a three-condition experiment, with one independent variable, the Kruskal-Wallis test is used. The data for all 16 items is visualised in Figure 5.12. The median for condition 1: ($Mdn = 5.94$), for condition 2: ($Mdn = 4.38$), for condition 3: ($Mdn = 5.13$). The Kruskal-Wallis test result for all 16 items is $H(2) = 4.3258, p = 0.115$. The result shows that there is no significance

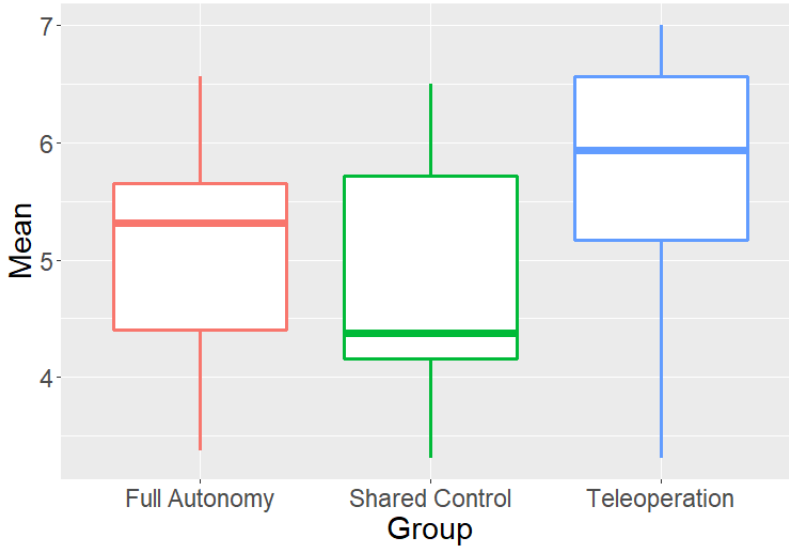


Fig. 5.12: Mean QUEAD scores for all three conditions

between the conditions. To calculate the effect size, the Pairwise Wilcoxon Rank Sum Tests is used. The p-score for all three pairs is $p > 0.05$, meaning the effect is small sized.

Perceived Usefulness

The means for items related to perceived usefulness are calculated. The data is visualised in Figure 5.13. The median for condition 1: ($Mdn = 5.87$), for condition 2: ($Mdn = 4$), for condition 3: ($Mdn = 5$). The results show that there is a significant difference between groups, $H(2) = 6.4467, p = 0.03982$. The effect size between conditions 1-2 $p = 0.035$, 1-3 $p = 0.168$ and 2-3 $p = 0.487$, shows that between conditions 1 and 2 the effect size is big with condition 1 scoring higher.

Perceived Ease of Use

The means for items related to the perceived ease of use are calculated, and the data is visualised in Figure 5.14. The median for condition 1: ($Mdn = 6.2$), for condition 2: ($Mdn = 5.6$), for condition 3: ($Mdn = 4.8$). The results from the Kruskal-Wallis test show that there is no significant difference between the groups, $H(2) = 3.9777, p = 0.1369$. The effect size between neither of the groups was bigger than 0.05, therefore, the effect size is small (1-2 $p = 0.19$, 1-3 $p = 0.19$ 2-3 $p = 0.77$).

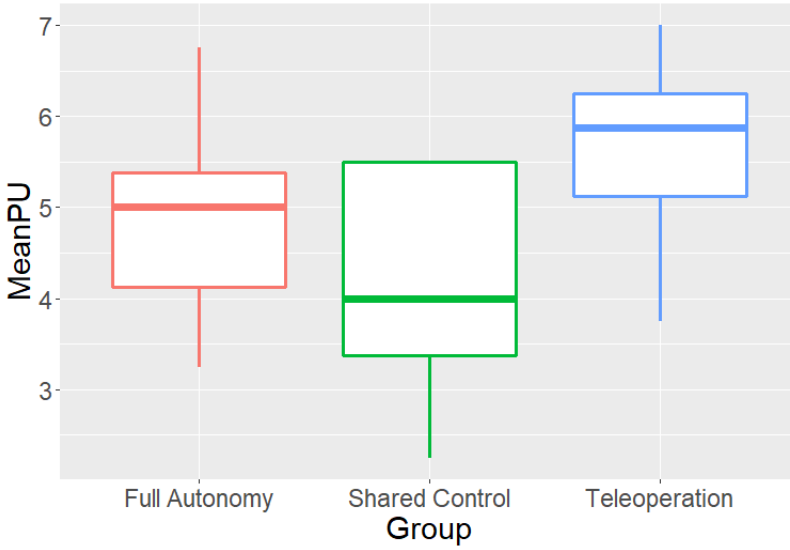


Fig. 5.13: Mean Perceived Usefulness scores for all three conditions

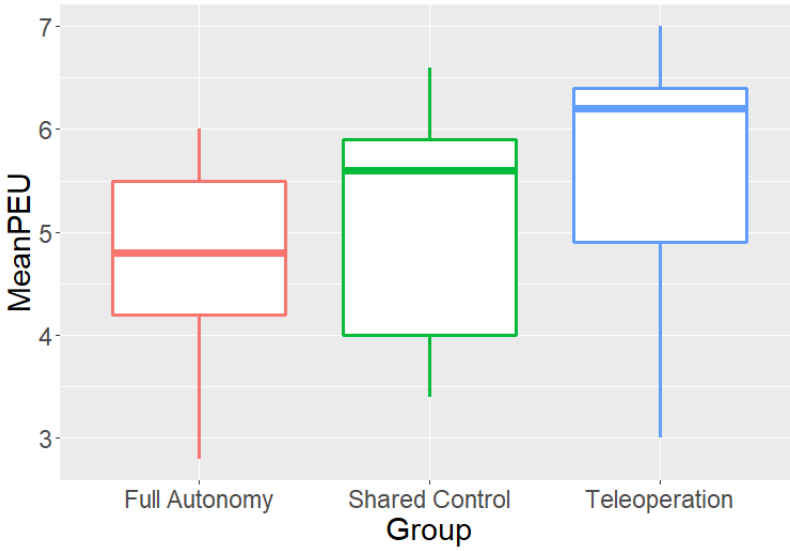


Fig. 5.14: Mean Perceived Ease of Use scores for all three conditions

Emotions

The means of the items related to Emotions are calculated and the data is visualised in Figure 5.15.

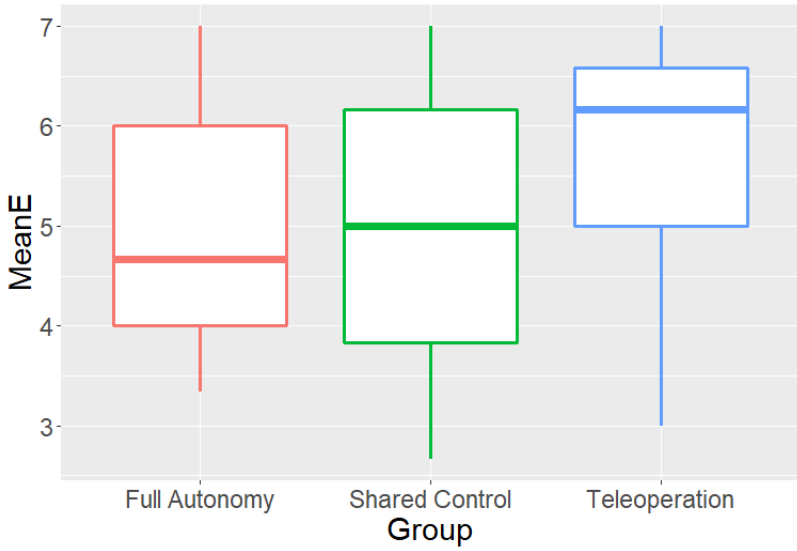


Fig. 5.15: Mean Emotions scores for all three conditions

The median for condition 1: ($Mdn = 6.17$), for condition 2: ($Mdn = 5$), for condition 3: ($Mdn = 4.67$). The results from the Kruskal-Wallis test show that there is no significant difference between the conditions, $H(2) = 2.6198, p = 0.2698$. The effect size between all conditions is bigger than 0.05, therefore, the effect size is small.

Attitude

The means of the items related to Attitude are calculated and the data is visualised in Figure 5.16.

The median for condition 1: ($Mdn = 6.25$), for condition 2: ($Mdn = 5$), for condition 3: ($Mdn = 6$). The results from the Kruskal-Wallis test show that there is no significant difference between the groups $H(2) = 2.1536, p = 0.3407$. The effect size between all groups is bigger than 0.05, therefore, the effect size is small.

Comfort

The means of the items related to Comfort is calculated and the data is visualised in Figure 5.17. The median for condition 1: ($Mdn = 6.25$), for condition 2: ($Mdn = 5$),

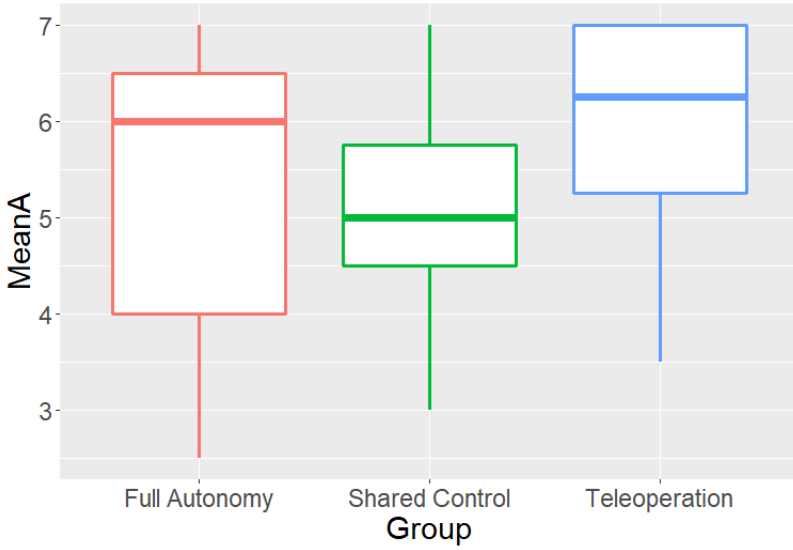


Fig. 5.16: Mean Attitude scores for all three conditions

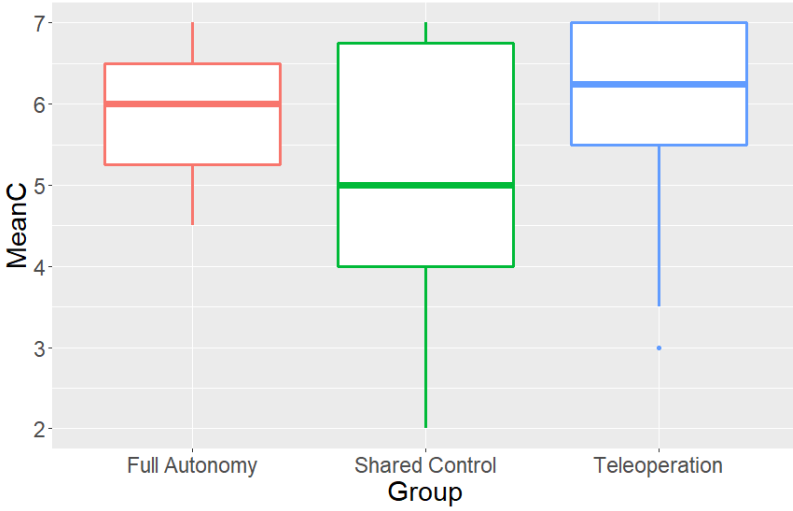


Fig. 5.17: Mean Comfort scores for all three conditions

for condition 3: ($Mdn = 6$). The results from the Kruskal-Wallis test show that there is no significant difference between the groups $H(2) = 1.244, p = 0.5369$. The effect size between all three groups is bigger than 0.05, therefore, the effect size is small.

Number of Breakouts

We calculate the number of breakouts per minute by dividing the number of breakouts by total time. We repeat the calculation for each participant from groups 2 and 3. The data is visualised in Figure 5.18.

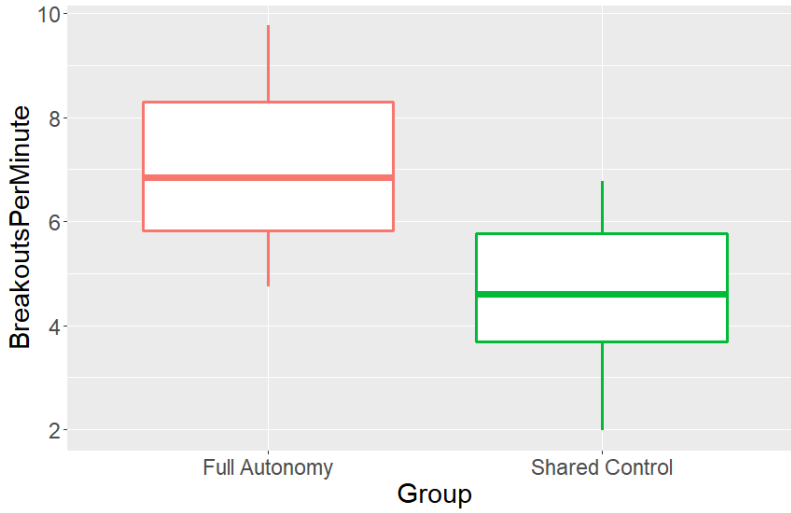
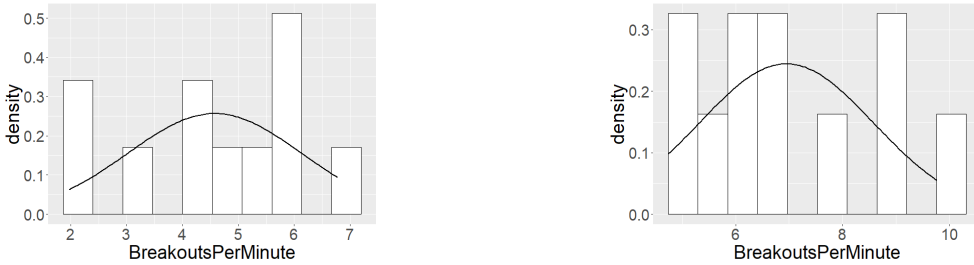


Fig. 5.18: Breakouts per minute in conditions Shared Control and Full Autonomy

The number of breakouts per minutes, can be treated as parametric data, however it still needs to meet the criteria of being normally distributed. The histogram of breakouts per minute for condition 2 and 3 are presented in Figure 5.19. The median for condition 2:($Mdn = 4.6$), for condition 3: ($Mdn = 6.84$).

We run additionally the Shapiro test. The p-values for condition 2: $W = 0.9518, p = 0.667$, and condition 3: $W = 0.9439, p = 0.5674$, are both higher than 0.05, therefore, data can be treated as parametric.

Since it is a two-condition experiment with one independent variable we use the repeated measures t-test. The result from the test $t(10) = 3.6571, p = 0.00441$, shows that there is a significant difference between the groups in the breakouts per minutes. We run the Cohen's d test to check for the effect size and the result $d = -1.10$ shows that the effect size is large. This means that participants in condition 2 have significantly fewer breakouts per minute than in condition 3.



(a) Shared Control **(b)** Full Autonomy

Fig. 5.19: Histograms showing number of breakouts per minute for both conditions

raw NASA-TLX

We calculate the results from the raw NASA-TLX by using the sum of ratings from the six scales as the estimation of overall workload [39]. The data is visualised in Figure 5.20. The median for condition 1: ($Mdn = 13.5$), for condition 2: ($Mdn = 19$), for condition

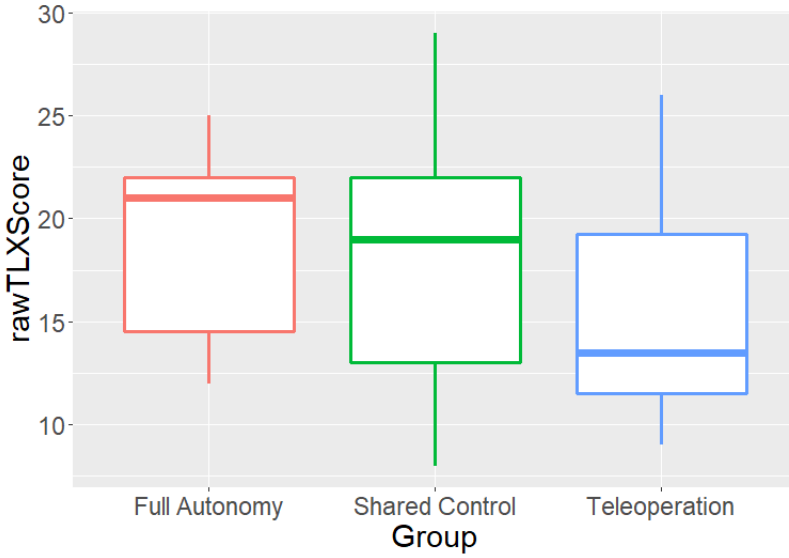


Fig. 5.20: raw NASA-TLX scores for all three conditions

3: ($Mdn = 21$). Since the data in not parametric and we have three test conditions with one independent variable we are are running the Kruskal-Wallis test. The results $H(2) = 1.5669, p = 0.4568$, show that there are no significant differences between the

groups.

We run the Pairwise Wilcoxon Rank Sum Tests to check if there are any big effect sizes between the groups. However, the effect sizes between all three groups are $p > 0.05$, therefore, the effects are small sized.

5.4 Evaluation Discussion

The evaluated data collected from the experiment, related to the measured time, shows that there are no significant differences between the conditions. However, on average the participants assigned to the Teleoperation mode finished the task the fastest. This applies to both total times and all three subgoal times.

Another measurement collected from the experiment is the number of breakouts per minute. The test results show that there are significant differences between participants in conditions Shared Control and Full Autonomy.

The next data, which was evaluated, is the data from the QUEAD questionnaire. We evaluate both the total QUEAD score as well as five subcategories. For all measurements, the results show that there are no significant differences except for the subcategory Perceived Usefulness, where the participants from the Teleoperation condition rated the control mode significantly higher.

The last evaluated measurement are the raw NASA-TLX scores. The results show that there are no significant differences between the conditions, however, similarly to previously mentioned measurements, participants from the Teleportation condition rated their condition as the one imposing the least workload.

To summarize, the conducted tests show three things. First, we got significant differences for two of the measurements, namely the number of Breakouts per minute, which were collected while the participant was controlling the robot. The result showed that participants assigned to Shared Control performed fewer breakouts than in the Full Autonomy condition. The second measurement, which showed significant differences between the conditions, was the Perceived Usefulness from the QUEAD questionnaire. Participants assigned to condition Teleoperation, scored significantly higher than in the two other conditions.

The perceived Usefulness can be described as the degree to which a person believes that using a particular system would enhance their job performance. Therefore, we can assume that the participants from condition Teleoperation perceived the technology to be more useful for what they attempt than in the other conditions.

Those result may indicate that participants are not as willing to give the control to the robot and prefer to control it by themselves.

Secondly, we did not got any significant differences for the rest of the collected measurements. Therefore, it is difficult to evaluate whether our few significant results can indicate differences between the conditions.

Thirdly, for all the questionnaires and data collected from the experiment, the participants assigned to the Teleoperation condition rated the control mode the highest and performed best. Although we did not get significant differences in the majority of tests, we observed this trend for all of them.

Chapter 6

Discussion

In this study we tried to find whether different LORA have an effect on obstacle avoidance in a USAR scenario where the operator cannot see the obstacles. We set up an experiment to test different autonomy modes and collect data through objective and subjective measurements.

Although we only used a robot in a simulation, we feel that this adequately represents the scenario we want to emulate. During a USAR mission, the robot operator will also sit in front of a screen and not be able to see the robot. Thus, it should not really matter for the experiment whether there is an actual robot driving somewhere or not.

In this project we define three different autonomy modes, which we compare in our experiment. We wanted to stretch over the whole spectrum, so we use the Teleoperation and Full Autonomy conditions, plus one in the middle, Shared Control. However, we are not sure whether the difference between Shared Control and Full Autonomy is large enough to make a significant impact. The only change is that in Shared Control the operator still is in control over the linear velocity, while the system is in full control in Full Autonomy.

Maybe it would be better to either add a fourth condition, which is somewhere between Teleoperation and Shared Control, or to move Shared Control more into the middle of the autonomy spectrum.

We also experienced that some participants did not use the test round as long as they maybe should have. Especially in regards to figuring out how their control mode, the test condition, behaves around obstacles, many could have taken more time to be more certain about it. We experienced that some participants, as soon as they knew how to control the robot, quickly drove to the green circle, to start the real experiment. This led to some confusion when they actually were confronted with the real invisible obstacles and took them some time to figure it out then.

We could potentially increase the length of the test round and somehow confront

the participants with obstacles so they get a better idea of what is happening.

In USAR scenarios the robot is usually controlled by an experienced and trained operator. Since those missions require the robot to scan the area or find the victims as fast as possible, it makes sense to use people with high expertise in this area as operators. Since our scenario resembles a USAR mission, it would make sense to test the system on experienced operators. However, since we did not have access to such group, we could not use them for testing. For future development of the system and testing we would suggest to use experienced robot operators as a test group.

Additionally, to make our USAR scenario even more realistic, we can add thermal camera imaging to the screen for the operator. Through this, we can add the task of finding victims, which is a main goal of most USAR missions. By adding the thermal image we do not only make the whole scenario more realistic but also make the obstacle avoidance task seem less important. The participant's attention could be more focused on finding the victims than to avoid obstacles, which then acts as a distractor.

This could add also additional measurement for our tests - which would be number of found victims.

Naturally, in a real application environment, the map will look differently, screen sizes and layout may vary, but since these are most likely not identical across different platforms anyways, our Gazebo simulation should be an accurate example for our purpose.

The TurtleBot3, which we use in the simulation, is a small and agile mobile robot and could very well be used in a similar real life scenario. However, it has a number of shortcomings.

- In a house with obstacles of unknown size and shape, as well as doorsteps and carpets, the robot needs to be able to climb small surface irregularities and be able to drive on any surface material. We did not check whether the TurtleBot3 has these capabilities.

- Obstacles can be of different heights and materials. The Laser Distance Sensor on the TurtleBot3 only scans a two-dimensional plane at a height of ca. 190 mm. This means that the sensor will not recognise any obstacles, which are lower than this. Those obstacles are a great threat to the TurtleBot3, since it might not be able to drive above them. This could for example be anything from shoes to rubble. Ideally, a USAR robot would be equipped with a three-dimensional laser sensor.

  Furthermore, obstacles can be made from different materials, some of which might not be detectable with the sensor. However, we have not checked if this is a problem. It is just something to be aware of.

- This specific TurtleBot3 model, the Burger, is not equipped with a camera. For the task of USAR a camera is necessary to locate possible victims. This can either be a standard RGB camera or, as often used, a thermal camera.

An issue regarding the TurtleBot3 in the simulation, which came up repeatedly, is that many participants had difficulties telling where the front and the back of the robot was. Eventually, they got used to it, but by marking it clearly in the simulation, maybe through an arrow, we could have avoided this issue.

The participants received the experiment instructions in written form and we did not experience that anybody had difficulties understanding the scenario or what was asked of them. In the QUEAD questionnaire the questions are about the control mode and in the beginning, a few participants were not quite clear about what that was referring, but after we rewrote the instructions more clearly, it was not a problem anymore.

It comes as no surprise that the majority of participants in a university experiment, are university students. Though it has often been discussed how well this sample group represents the entire population, we are quite content with our participant profiles. This is also due to the fact that the typical USAR robot operator is most likely not a random person from the population, but a technically experienced and trained person. Therefore, we feel that it does not matter much that our sample group was not adequately representing, for example, the elder population. By asking for the participant's subjective PS3 controller skills and previous robot control experience, we also got an impression of whether technical control skills do have an influence on the performance.

Generally, it would be beneficial to test on real USAR robot operators, as mentioned before, but this was not possible for us.

When creating the world, we tried to simulate a realistic layout of a house, with walls at right angles, doors, and hallways. Real houses will, of course, have an infinite variety of layouts with all kind of different features, but the world used in this experiment should provide an valid general example for a house layout.

Regarding the goal of eliminating any measurement noise, we noticed several times that participants had problems reaching goal points. They unknowingly drove in a wrong direction, trying to reach the goal from a different direction and needed a long time to eventually find the correct path. We designed the obstacle layout in a way that ensured that only one general path was available towards each goal, in order to achieve comparable results where participants were not able to reach the goals faster by choosing a shorter path. However, this meant that we had to block all the other paths, which led to a big number of obstacles with many possibilities of getting lost. Especially, before the yellow subgoal many participants tried to approach the goal from the bottom and needed a long time before being able to find the small passage towards the goal.

We fear that these occurrences distort the actual completion times, since they do not happen due to the specific condition but randomly. However, since we did not observe any of the conditions encountering this problem more often than others, we hope that they cancel each other out between the conditions.

To solve this problem, the world could be designed in an even more restrictive way, where it is less likely that the participants stray away from the optimal path. This way, we could bundle the focus as much as possible on the actual problem, i.e. obstacle avoidance.

Besides this problem, we are generally positive about the layout and type of obstacles, with a mix of shapes and sizes we hope to accurately represent different possible obstacles.

Just like with the screen layout, we cannot say with confidence what type of controller the robot operator would use in a real USAR mission. This does most likely depend on the type and functionality of the robot. In this experiment, the participant was using a PS3 controller and we feel that it was an appropriate choice. Most of the participants were already familiar with the PS3 controller specifically or other gaming console controller in general. This familiarity made it easier for the novice users of our system to more quickly figure out how to control the robot.

On the flip side, we could pretty quickly observe that certain participants with little experience in gaming, had difficulties using the PS3 controller. However, this problem would most likely not have been solved with another controller, since the PS3 controller works on the most simple principles (i.e. move control knob forward to move the robot forwards). Some of the less experienced participants had in particular the problem of seeing things from the robot's perspective. Since the control directions are relative to the robot (i.e. move control knob to the right, the robot rotates to the right), it is necessary for the participant to adopt to this change in perspective. The participants who experienced problems were trying to control the robot relative to the map (i.e. move control knob to the right, robot drives to the right on the map, independent of the robot's actual rotation).

This particular problem could of course be solved by fixing the robot in the simulation, so the map turns around the robot and the robot always is facing up/forward on the screen. This, however, could confuse other participants who are used to the usual control mode.

Some participants expressed their wish to be able to increase the velocity of the robot beyond the limits that we set. While a real robot operator will most likely know and accept the reason for velocity limits, a participant in this experiment might have gotten frustrated and annoyed.

These issues highlight again the relevance of experience in a task like this, since we believe that after using the system a couple of times, everybody would get used to this control mode. However, since we did not give them several tries, the test run might not have sufficed in levelling these skill differences.

A central part of the project was the system's ability to differentiate between walls and obstacles. During the experiments we did not encounter an occasion where the system mistakenly identified an obstacle as a wall or vice versa. However, we also did not specifically test the performance of the identifier in all kind of extreme scenarios, like tiny obstacles, or obstacles directly next to a wall. The performance of the system was definitely adequate for our purposes. In more advanced versions, the system could maybe not only differentiate between walls and obstacles but also between different types of obstacles, e.g. chair, table, or rubble. This information could possibly be used to invoke specific type of avoidance behaviour. A more advanced three-dimensional sensor, as mentioned earlier, might make this classification possible.

The central part of the system is the supervisor. Here, all information from other parts of the system is gathered and processed. Through a FSM the supervisor decides who is in control of the robot, the operator or the system itself. The supervisor, additionally, sets speed limits relative to the distance of the robot to the nearest object and provides the operator with feedback about who is in control of the robot.

While the basic functionality of the supervisor system works for the experiment, there is always space for improvements.

There is a variety of different feedback methods in HRI. For this project we chose to go with audio feedback for different reasons. We use audio feedback in two situation.

- To indicate to the operator that the robot is in (partial) control.
- To give the operator an idea of the distance to and direction of the nearest obstacle.

Using audio for giving feedback who is in control was generally well received by the participants. They mentioned that it was very clear when the robot took over control and that it helped with understanding when they made a successful breakout. The sound itself was not seen as disturbing, annoying or distracting.

Using audio in the other case, for the Feedback Assisted Teleoperation received mixed reviews from the participants. Some of them mentioned that it was not always clear to them from which side the audio signal came in the headphones, i.e. on what side the obstacle was. This could of course be due the quality of the used headphones, however, most participants did not name this as a problem.

Other participants complained about the volume and pitch of the audio signal, calling it irritating and distracting. This is of course a very important point to follow up on, since we chose audio feedback instead of visual feedback so it would not give extra

distractions visually, but if the audio is perceived as distracting, it could critically reduce task performance.

To improve the solution, it would be valuable to additionally test different feedback methods, like visual or haptic feedback.

One of the collected measurements during the course of the experiment was breakouts per minute. From these measurements we initially hoped to gain information about the participant's willingness to let the robot be in control. Although the results show that there are significant differences between the conditions, we are unsure about the reliability of these results. The breakout mechanism in both conditions works slightly differently. Participants in Full Autonomy condition, simply needed to move the PS3 control knob in any direction to break out. This made the breakout fairly simple, but it could also sometimes be performed unintentionally. In Shared Control condition, the participant had to steer the PS3 control knob either to the maximum left or right position to break out from the system's control. This mechanism could have made the breakout action to be more conscious than in the Full Autonomy condition. Since we are unable to evaluate if the breakouts performed by the participants in condition Full Autonomy, were intentional or not, we are aware that the collected data might be difficult to compare and the results, may be not reliable. Additionally, we collected only the data on total number of breakouts per participant. We assume it would be more useful to collect the data about number of breakouts between each of the subgoals as well. As mentioned previously, some participants got stuck between some subgoals, so it would make sense, that they used the breakout function more often. By looking at the number of breakouts between each subgoal we can minimize the impact of those situations, which could increase the validity of the collected data.

While investigating the topic of shared control in HRI, multiple studies [3], [24] provide information that shared control may be influenced by other factors derived from HRI. One of which is trust. Trust in human robot interaction is a multidimensional concept, which includes many components. One we think might be particularly interesting in the context of our experiment is responsibility, which can be defined as to what extent the robots explains and informs the user about its decisions and intentions. A study [25] shows, that users are more willing to trust and interact with the robot if they fully understand their intentions and get information about the robot's decisions. We could implement an algorithm, which would show the user the predicted robot path around the obstacle. This may increase the user's understanding of the situation and the robot's intentions, and may heighten the user's willingness to give the control to the robot. Furthermore, if the robot was able to predict the operator's overall goal, the autonomy of the system could be increased, therefore, the presented levels of LORA could be even more distinct, which might make a significant difference on the results.

Chapter 7

Conclusion

The development of more advanced robotic technology gives a rise in possible applications. A field with great potential is the area of shared control. Here the human operator and the robot are both involved in solving the task at hand. This can happen at various degrees of autonomy, from where the robot does almost nothing to fully autonomous solutions. Often in shared control tasks, the responsibilities are split according to the individual abilities. The need to share control can also arise, if neither of the involved parts have access to all the information required to solve the task. One such task is USAR with a mobile robot, where the operator only has access to a high level map of the building, while the robot can sense obstacles locally. In this experiment we investigate whether using different degrees of autonomy during obstacle avoidance has an effect on task performance. We state an open exploratory research question:

Does the LORA in a mixed initiative USAR scenario influence the task performance in obstacle avoidance when the operator cannot see the obstacles?

We design and test three different conditions with increasing degree of robot autonomy, from teleoperation, over shared control to full autonomy. The change in autonomy is through robot initiative, but operators have the authority to reclaim control. In the (semi-)autonomous conditions, we use a potential field method for reliable and efficient obstacle avoidance. Audio feedback about changes in autonomy are provided to the participants. Furthermore, in the Teleoperation condition we inform the user about the distance and position of the obstacle with audio feedback, which works similarly to the parking assistance systems. We measure how fast the participants complete the task and find no significant differences between the conditions. We, additionally, collect the information over number of breakouts from participants in the (semi-)autonomous con-

ditions. Although we find significant differences between the conditions, we are aware that the measurements might not be reliable, and therefore cannot for sure indicate differences between the conditions. In subjective measurements, we use the QUEAD and NASA-TLX questionnaires and find only one significant partial result, favoring teleoperation. However, we observe a general trend, both in objective as well as subjective measurements, that favor the use of the teleoperation condition. We conclude that participants might feel more comfortable having full control over the robot, but that further research is needed to investigate additional scenarios, autonomy modes, and ways to transfer control. Additional research can also be done in investigating how much influence on the participants the fact has that they do not have a freedom of choosing, when the robot is assisting them. The robot is taking the control independently of the operator's will, therefore, it may be perceived as forced. Therefore, participants may rate and perform worst in both of the conditions in which the robot is taking over control, due to the fact that they do not have the same freedom as in Teleoperation mode. In conclusion, we believe the study gives an interesting insight into the field of human-robot shared control in the context of different LORA in USAR tasks, however, despite the trend favouring Teleoperation, it does not provide any evidence that the different LORA might influence the participant's performance during obstacle avoidance in USAR missions.

References

[1] *Autonomy Measures for Robots*, ser. ASME International Mechanical Engineering Congress and Exposition, vol. Dynamic Systems and Control, Parts A and B, 11 2004. [Online]. Available: https://doi.org/10.1115/IMECE2004-61812

[2] P. Aigner and B. McCarragher, "Human integration into robot control utilising potential fields," in *Proceedings of International Conference on Robotics and Automation*, vol. 1, Apr. 1997, pp. 291–296 vol.1.

[3] N. Amirshirzad, A. Kumru, and E. Oztop, "Human adaptation to human–robot shared control," *IEEE Transactions on Human-Machine Systems*, vol. PP, pp. 1–11, 01 2019.

[4] F. Andy and H. Adam, *How to design an report experiments*, 2003.

[5] M. Baker and H. A. Yanco, "Autonomy mode suggestions for improving human-robot interaction," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, vol. 3, Oct. 2004, pp. 2948–2953 vol.3, iSSN: 1062-922X.

[6] R. Banz, M. Bolliger, G. Colombo, V. Dietz, and L. Lünenburger, "Computerized Visual Feedback: An Adjunct to Robotic-Assisted Gait Training," *Physical Therapy*, vol. 88, no. 10, pp. 1135–1145, 10 2008.

[7] J. M. Beer, A. D. Fisk, and W. A. Rogers, "Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction," *Journal of Human-Robot Interaction*, vol. 3, no. 2, p. 74, Jun. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=3109833

[8] J. Brookshire, S. Singh, and R. Simmons, "Preliminary results in sliding autonomy for assembly by coordinated teams," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 1, Sep. 2004, pp. 706–711 vol.1.

[9] D. J. Bruemmer, D. A. Few, R. L. Boring, J. L. Marble, M. C. Walton, and C. W. Nielsen, "Shared understanding for collaborative control," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 35, no. 4, Jul. 2005.

[10] D. J. Bruemmer, D. D. Dudenhoeffer, and J. L. Marble, "Dynamic-Autonomy for Urban Search and Rescue," p. 5.

[11] E. Bustamante and R. Spain, "Measurement invariance of the nasa tlx," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 52, pp. 1522–1526, 09 2008.

[12] M. Chiou, N. Hawes, and R. Stolkin, "Mixed-Initiative variable autonomy for remotely operated mobile robots," *arXiv:1911.04848 [cs]*, Oct. 2020, arXiv: 1911.04848. [Online]. Available: http://arxiv.org/abs/1911.04848

[13] A. De Luca, G. Oriolo, L. Paone, and P. Giordano, "Experiments in visual feedback control of a wheeled mobile robot," 02 2002, pp. 2073 – 2078 vol.2.

[14] M. Desai and H. A. Yanco, "Blending human and robot inputs for sliding scale autonomy," in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, Aug. 2005, pp. 537–542, iSSN: 1944-9437.

[15] M. R. ENDSLEY and D. B. KABER, "Level of automation effects on performance, situation awareness and workload in a dynamic control task," *Ergonomics*, vol. 42, no. 3, pp. 462–492, Mar. 1999. [Online]. Available: https://doi.org/10.1080/001401399185595

[16] J. C. Gerdes and E. J. Rossetter, "A Unified Approach to Driver Assistance Systems Based on Artificial Potential Fields," *Journal of Dynamic Systems, Measurement, and Control*, vol. 123, no. 3, pp. 431–438, Sep. 2001. [Online]. Available: https://asmedigitalcollection.asme.org/dynamicsystems/article/123/3/431/460292/A-Unified-Approach-to-Driver-Assistance-Systems

[17] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: A survey," *Foundations and Trends® in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007. [Online]. Available: https://doi.org/10.1561/1100000005

[18] B. Hardin and M. A. Goodrich, "On using mixed-initiative control: a perspective for managing large-scale robotic teams," in *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, ser. HRI '09. New York, NY, USA: Association for Computing Machinery, Mar. 2009, pp. 165–172. [Online]. Available: http://doi.org/10.1145/1514095.1514126

[19] S. Hart, "Nasa-task load index (nasa-tlx); 20 years later," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, pp. 904 – 908, 2006.

[20] Y. Horiguchi and T. Sawaragi, "Effects of probing behaviors to adapt machine autonomy in shared control systems," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, Oct. 2005, pp. 317–323 Vol. 1, iSSN: 1062-922X.

[21] H.-M. Huang, K. Pavek, J. Albus, and E. Messina, "Autonomy levels for unmanned systems (alfus) framework: an update," in *Proceedings of SPIE*, vol. 5804, no. 1. Bellingham WA: SPIE, 2005, pp. 439–448.

[22] K. Keane, D. McCrea, and M. Russell, "Personalizing feedback using voice comments," *Open Praxis*, vol. 10, p. 309, 12 2018.

[23] D. Kortenkamp, R. P. Bonasso, D. Ryan, and D. Schreckenghost, "Traded Control with Autonomous Robots as Mixed Initiative Interaction," p. 6.

[24] D. P. Losey, C. G. McDonald, E. Battaglia, and M. K. O'Malley, "A Review of Intent Detection, Arbitration, and Communication Aspects of Shared Control for Physical Human–Robot Interaction," *Applied Mechanics Reviews*, vol. 70, no. 1, 02 2018, 010804. [Online]. Available: https://doi.org/10.1115/1.4039145

[25] S. M. Merritt and D. R. Ilgen, "Not all trust is created equal: Dispositional and history-based trust in human-automation interactions," *Human Factors*, vol. 50, no. 2, pp. 194–210, 2008, pMID: 18516832. [Online]. Available: https://doi.org/10.1518/001872008X288574

[26] A. M. Okamura, "Haptic feedback in robot-assisted minimally invasive surgery," *Current Opinion in Urology*, vol. 19, no. 1, p. 102–107, 2009.

[27] A. Orahovac, "Mine action after the floods regional synergy in emergency response, technology development and capacity building." *In: 22nd Organization for Security and Co-operation in Europe (OSCE) Economic and Environmental Forum;*, Sep 2014.

[28] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, no. 3, May 2000.

[29] C. Pearce and R. J. Ackley, "Audiotaped feedback in business writing: An exploratory study," *Business Communication Quarterly*, vol. 58, pp. 31 – 34, 1995.

[30] H. Saeidi, F. Mclane, B. Sadrfaidpour, E. Sand, S. Fu, J. Rodriguez, J. R. Wagner, and Y. Wang, "Trust-based mixed-initiative teleoperation of mobile robots," *2016 American Control Conference (ACC)*, 2016.

[31] J. Schmidtler, K. Bengler, F. Dimeas, and A. Campeau-Lecours, "A questionnaire for the evaluation of physical assistive devices (quead): Testing usability and acceptance in physical human-robot interaction," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 876–881.

[32] T. B. Sheridan and W. L. Verplank, "Human and Computer Control of Undersea Teleoperators:," Defense Technical Information Center, Fort Belvoir, VA, Tech. Rep., Jul. 1978. [Online]. Available: http://www.dtic.mil/docs/citations/ADA057655

[33] S. Soekadar, N. Birbaumer, M. Slutzky, and L. Cohen, "Brain-machine interfaces in neurorehabilitation of stroke." *Neurobiology of disease*, vol. 83, 12 2014.

[34] P. Supriyono and P. Marjuki, "Ultrasonic sensor parking assistant with arduino uno," *International Journal of Advanced Research in Engineering and Technology (IJARET)*, vol. 11, pp. 26–33, 06 2020.

[35] S. Thrun, "Toward a framework for human-robot interaction," *Human-computer Interaction*, vol. 19, pp. 9–24, 06 2004.

[36] E. Tidoni, P. Gergondet, A. Kheddar, and S. M. Aglioti, "Audio-visual feedback improves the bci performance in the navigational control of a humanoid robot," *Frontiers in Neurorobotics*, vol. 8, 2014.

[37] J. Wang and M. Lewis, "Human control for cooperating robot teams," in *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2007, pp. 9–16, iSSN: 2167-2148.

[38] C. R. Wottawa, B. Genovese, B. N. Nowroozi, S. D. Hart, J. W. Bisley, W. S. Grundfest, and E. P. Dutson, "Evaluating tactile feedback in robotic surgery for potential clinical application using an animal model," *Surgical Endoscopy*, vol. 30, no. 8, p. 3198–3209, 2015.

[39] Ženija Roja, V. Kaļķis, H. Kaļķis, and I. Pencis, "Assessment of firefighters-rescuers' work severity in relation with interaction between physical and mental load," *Proceedings of the Latvian Academy of Sciences. Section B. Natural, Exact, and Applied Sciences.*, vol. 63, no. 6, pp. 264–270, 2010. [Online]. Available: https://doi.org/10.2478/v10046-010-0002-0

Appendix A

Questionnaires

The control mode is useful

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

The control mode enhances my working performance

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

I accomplished the given task rapidly

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

I was able to perform precise motions with this control mode

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

This control mode is easy to use

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

This control mode is rigid and inflexible

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

This control mode feels cumbersome

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

This control mode would be helpful to me

1	2	3	4	5	6	7
Entirely desigree	Mostly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Mostly agree	Entirely agree

Fig. A.1: First part of the QUEAD questionnaire

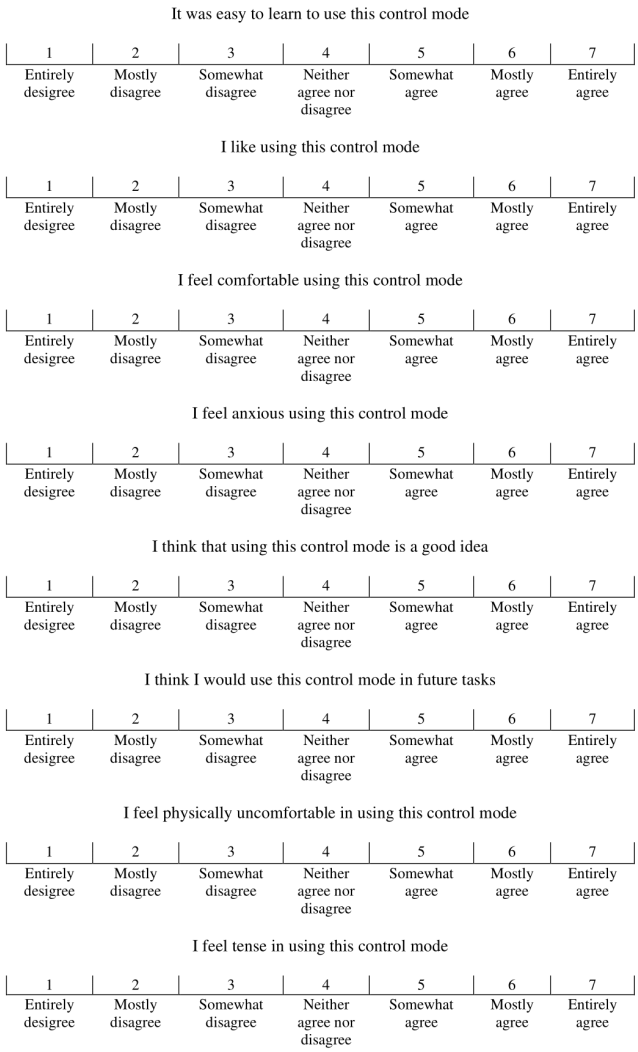


Fig. A.2: Second part of the QUEAD questionnaire

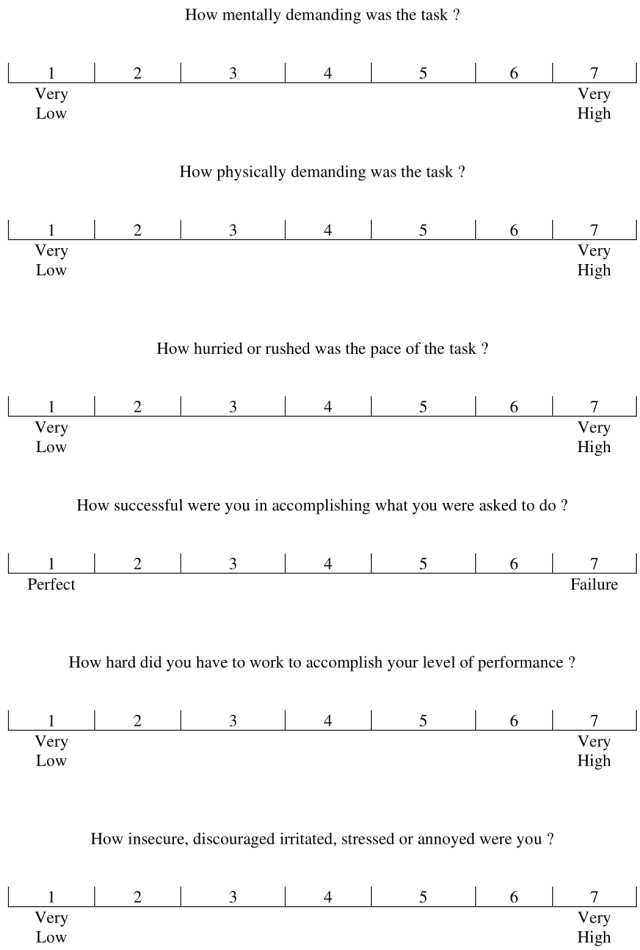


Fig. A.3: raw NASA-TLX questionnaire