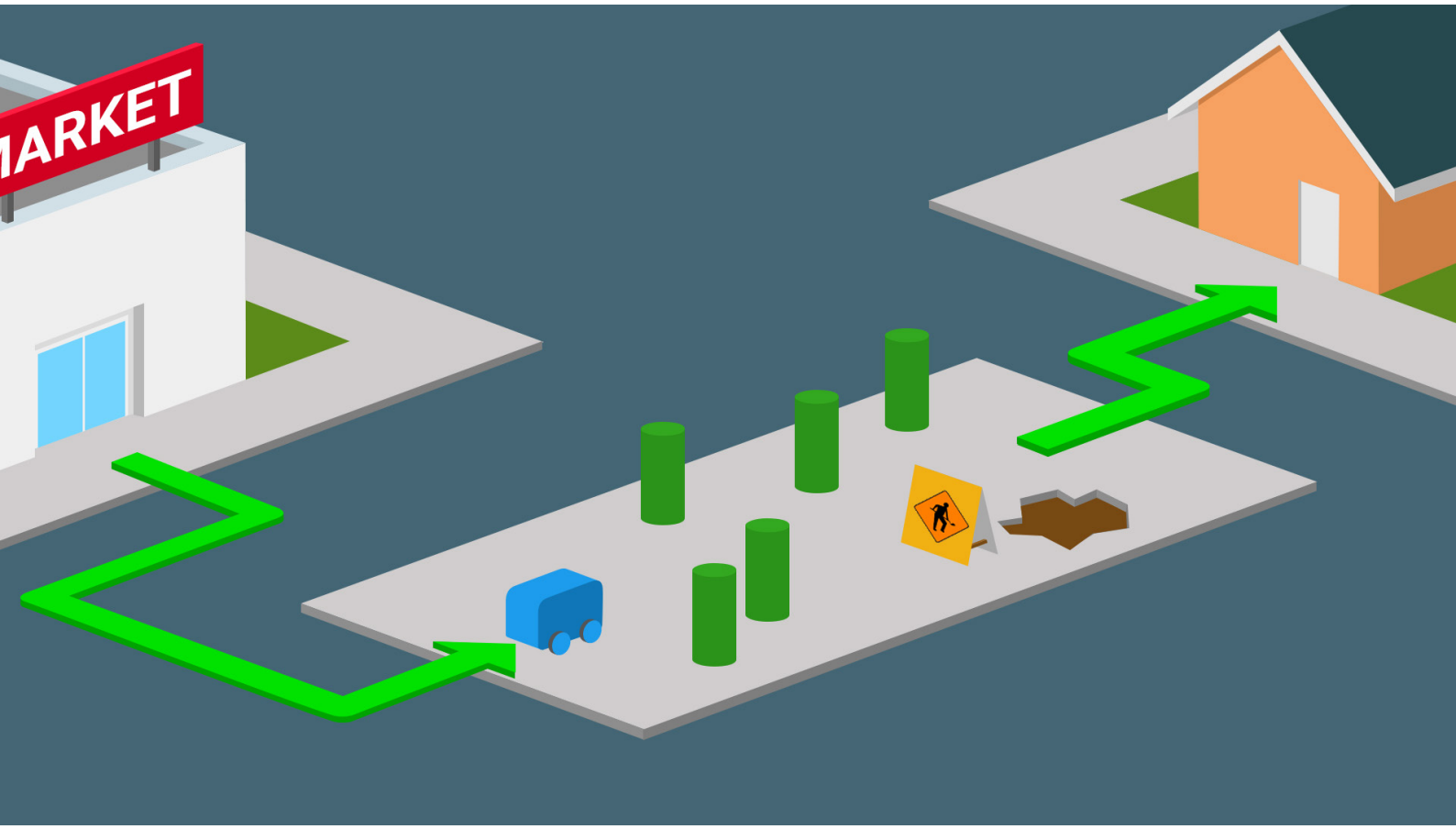
Designing socially adaptive behavior for mobile robots

Master thesis
Integrated Product Design

Patrick Keesmaat



TU Delft

Master thesis by Patrick Keesmaat
February 2020
Faculty of Industrial Design Engineering,
Master: Integrated Product Design

Delft, University of Technology

Chair: Dr. Zoltán Rusák
Title: Assistant Professor Industrial Design Engineering

Mentor: Adrie Kooijman
Title: Lecturer Design Engineering

Acknowledgements

Working with autonomous robots as part of my graduation project was an opportunity I could not decline. Programming has been a long term hobby of mine, and combining design with programming felt like a graduation project which suited me perfectly. The journey has been a rollercoaster of ups and downs. At times it could be frustrating and time consuming, trying to solve the many bugs I encountered. But all the more rewarding were those moments when it finally worked and I could just lean back for a few minutes, admiring the social navigational model at work.

The result of what I have achieved would not have been possible without the people who have helped and supported me.

First of all, I would like to thank Dr. Zoltán Rusák, for thinking along and pushing me to always do better. And thank you, Adrie Kooijman, who has been my mentor in all the right ways; supporting me in my work and being there to discuss not only the project, but also life in general.

Secondly, I would like to give my thanks the Applied Labs team in the faculty of Industrial Design Engineering. Thank you for providing me with the room, the materials and the occasional helping hand which allowed me to work on and achieve the many ROS and ROSbot related parts of this project.

Thirdly, I would like to thank Dr. ing. Sergio Grammatico who helped me out in a time of need. When our ROSbot was delayed for several weeks, he provided me with one from his own lab to work with. Without his generousness I would have severely fallen behind schedule.

Fourthly, I give my thanks to my family. Thank you for supporting me, for helping me out when I needed a pilot participant, for thinking along when I encountered problems and for facilitating the work environment as you have.

And lastly, I give my thanks to my girlfriend Lotte. Thank you for being there when I needed it. I could always come to you when I needed to get something of my chest, and you helped motivate me when I was experiencing one of those rollercoaster lows. Thank you for thinking along and giving advice because without you, this journey would have been much more difficult.

Executive summary

An autonomous guided vehicle can be used for the delivery of goods. To deliver these goods to your home, the mobile robot will be driving in pedestrian rich environments. In these environments the robot will need to socially navigate itself in a way which is both pleasant for the pedestrians and progressive for the robot.

During its drive, the robots behavior should be adapted to the situation it finds itself in. The underlying theory for this is the Social, Technology and Service triangle. This triangle dictates the balance between three aspects. The technology aspect focuses on power consumption and efficiency. The service aspect is all about delivery time and location. The social aspect is all about minimizing social disruption and facilitating intuitive behavior.

The behavior of pedestrians can be modelled with the Social Force Model. In this model the objects and pedestrians apply social forces onto each other which determine their movements. The robot can use this SFM for its own social navigation.

Multi-Policy Decision Making (MPDM) can be added onto the Social Force Model. This allows for switching between three basic policies: go-solo, follow and stop. Through forward simulations the robot can predict and decide which of the three policies brings the most progress and the least social disruption.

However, SFM-MPDM on its own does not cope well with passing and crossing; thus methods were designed to handle these situations. In passing the robot looks ahead and makes room, often keeping right. In crossing the robot slows down and deviates to cross behind the pedestrian.

The Social Force Model, the Multi-Policy Decision Making and the passing and crossing all combined form the social navigational model. This model consist of the many different parameters which govern the behavior and is part of the overall computational mechanism. This mechanism takes in the environment through sensors such as LiDAR. It preprocesses these inputs and sends them to the model. The output of this model is what determines the locomotion and behavior of the robot. To find the behavior and its underlying parameters an Evolutionary Algorithm was employed.
The learnt behavioral parameter sets were tested with participants (n=42) and a physical ROSbot. They were asked to rate the robots behavioral performance on comfort, predictability and communication of intent.

From these results, it became clear that the robots behavior influences the experience of the pedestrians, but it is unclear which parameter exactly influences which part of the behavior. There are certain expectations to what impact a parameter or group of parameters has, but at which times and to what effect this controls parts of the behavior is difficult to identify.

Nevertheless a good foundation has been laid down for future projects through a social navigational model and a way to learn parameters for the many different situations an autonomous delivery robot can encounter. A recommendation is to improve the Evolutionary Algorithm in order to facilitate parameter learning which better matches the desired behavior in the STS-triangle.

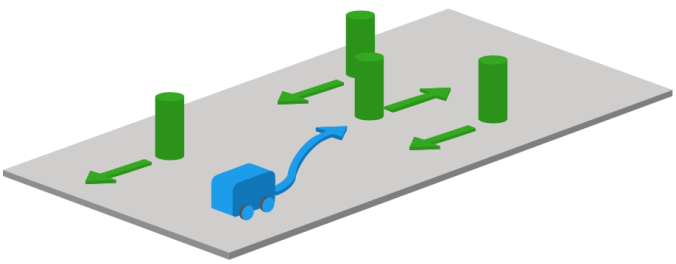


Table of contents

Introduction

Context

As the number of 'home delivered goods' is experiencing a strong growth which is not expected to halt soon, it is interesting to look into new ways of delivering these goods to the customer. One of these new ways is autonomous delivery. If the autonomous vehicle is to help with the delivery of these goods in a similar manner to a delivery man; then it needs to be able to navigate on the sidewalk. As not all residences are directly adjacent to streets suitable to larger vehicles.

The advantage of autonomous (self-driving) vehicles on the public streets is that they are guided by set (traffic) rules. This is not the case within the context of autonomous navigation on the sidewalk. While one is navigating on the sidewalk, you need to deal with pedestrians and thus deal with the dynamic rules which apply in these pedestrian rich environments.

The challenge

An autonomous guided vehicle (AGV) in pedestrian rich environments should be designed in such a way as to perform on technological, human interaction (social) and payload delivery (product service) aspects. The AGV should be able to deal with a range of situations and be able to pick the optimal* approach for a given situation.

*Optimal can be defined as a trade-off between the factors from the three above mentioned aspects, these factors can be: distance, time, energy, social disruption, communication of intentions, intuitive interaction, etc.

The project challenges include, but are not limited to:

- Autonomous guided vehicles might be experienced as obtrusive and provoke a response from pedestrians.
- The AGV should be optimally guided in the given situation, this requires a computational mechanism build from one or a combination of machine learning methods.
- The computational mechanism cannot be developed on the technology, social and (product) service aspects without prototyping and testing.

The assignment

The assignment is defined as such:

"Research and design a computational mechanism for an autonomous guided vehicle based on the social, technology and product service needs by utilizing a machine learning method to deal with a variety of situations while expressing its (change of) intention."

The computational mechanism's goal is to optimally navigate the AGV through the pedestrian rich environments while taking into account the technology, social and product service aspects. The computational mechanism should encompass a variety of trained models/parameters for specific situations. Some examples of situations are: near-empty streets, high density pedestrian areas, passing, overtaking, narrow or (partially) obstructed paths, etc. The performance of the robot (and thus its computational mechanism) should be tested which users and evaluated to use as input for iterations.

Approach

The project is divided into two main phases; the Research and Exploration phase and the Development and Testing phase. In the first phase the literature will be studied and the context of autonomous guided vehicles will be explored. This will include, among other things, looking into pedestrian traffic, social navigational methods, intent communication, parameters which impact vehicle performance and defining the product service. The Research and Exploration phase will be concluded with a Program of Requirements and Wishes and a scope and design goal. The second phase will, as the name suggests, be about the development and the testing of the computational mechanism. This will include various kinds of digital simulations and a physically driving robot based on the customized computational mechanism. These simulations and the physical robot will be combined with user tests to evaluate performance on the three aspects (Social, Technology and Product Service).

Research and exploration

In this phase an overview of the literature will be given and the context of AGVs will be explored. The Research and exploration phase will be concluded with a Program of Requirements and Wishes and a scope and design goal.

Situations in pedestrian traffic

Pedestrians encounter many situations during their navigation from A to B. These situations can be as simple as walking along an empty sidewalk, or as complex as navigating through a crowded shopping street. At the same time the pedestrian has to take into account the obstacles on its route, the generally accepted but unwritten rules of pedestrian traffic and any unexpected situations emerging from the dynamic environment.
In this chapter, a range of situations is laid out and a selection is made from this range which are coined the 'core situations'. Next, the question is discussed if an AGV should adapt its behavior dependent upon the core situation it finds itself in.

What situations can the AGV encouter?

A brainstorm on the situations an AGV can be in while it's navigating through pedestrian traffic, resulted in an ever growing mindmap of situations and conditions. A few examples are:

- Navigating around obstacles on the road
- Using a pedestrian crossing
- Taking traffic lights into account
- Overtaking relatively slow moving pedestrians
- Crossing a busy flow of pedestrians
- Dealing with a playing child or group of children
- Driving on/off a curb or ramp

The full mindmap with 40+ situations can be found in Appendix 'B. Mindmap of AGV situations'. To make this project and the many different situations more manageable, a selection of eight core situations was made. These core situations are the basis of what the AGV should be able to handle and some of which on their own will already pose quite a challenge for the AGV.

The eight core situations, ranked based on relative expected difficulty, are defined as:

1. Empty street.
2. Obstacles in path.
3. Single pedestrian.
4. Normal street situations (a few pedestrians passing and some behind and ahead).
5. Slow moving (group of) pedestrians ahead.
6. Crossing a bi-directional flow of pedestrians with regular gaps.
7. Crossing a busy bi-directional flow of pedestrians without gaps.
8. Omni directional pedestrian traffic (such as a busy shopping street).

An illustrative overview of these eight core situations can be found on the next page in Figure 1 through Figure 8.

Figure 1. Empty street.

Figure 2. Obstacle in path.

Figure 3. Single pedestrian.

Figure 4. Normal street situation.

Figure 5. Slow moving (group of) pedestrians.

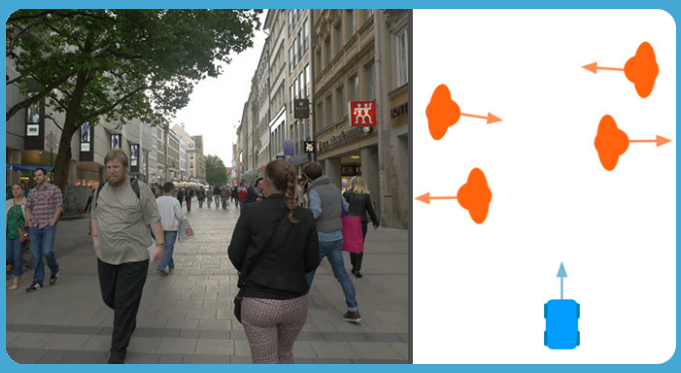


Figure 6. Crossing bi-directional flow (regular).

Figure 7. Crossing bi-directional flow (busy).

Figure 8. Omni-directional pedestrian traffic.

Should the AGV adapt its behavior based on the situation?

As a human walking among other pedestrians, one adapts its behavior to the situation. When the streets are near empty, you look further ahead, your maximum comfortable velocity increases and you follow a more straightforward trajectory.
When the crowd is dense, you slow down to cope with the number of people, the radius of your personal sphere shrinks and you weave through the traffic, possibly following someone ahead, to reach the goal you're aiming for.

Imagine a robot employing just one behavior for all situations. If this behavior is fine tuned for dense pedestrian environments, the AGV might invade the relatively larger personal sphere in less crowded environments and drive too slow. Vice versa it will cause a social disruption if it tries to navigate through the dense crowd at relatively high speeds while trying to maintain a straightforward trajectory. You might think to solve this issue by taking a kind of average of all core situations appropriate behavior and applying that everywhere. This will result in a robot which will just perform at mediocre levels at best.

Instead what you want is a robot which can be interacted with intuitively (predictable) and which is experienced as comfortable (ease of use). As de Groot, S. (2019) states,

"Pedestrians should be able to use their current mental models about sidewalk behavior and technology interaction, to interact with the robot."

Or in other words; pedestrians should be able to interact with the robot in a way they are used to with other pedestrians. The robot should behave in the way a pedestrian would while within the bounds of its own technological ability.

By changing its behavior to the situation the AGV finds itself in, it navigates in a social manner. To further the perception of a socially aware robot, the AGV should occupy intuitive and comfortable behavior. It should thus express its intentions, take into account personal space and try minimize social disruption in general (de Groot, 2019).

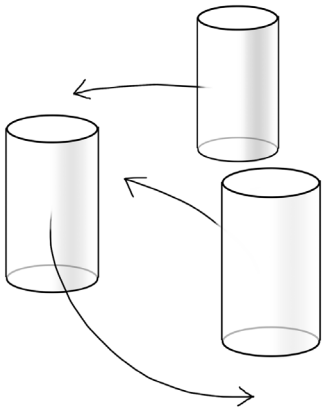
This is but one aspect however. As the AGV is also a product with a purpose, delivering its parcels, the product service aspect should also be included. And finally there are technological aspects that come into play when using an autonomous vehicle. These three aspects together form the triangle from which the behavior of the robot should arise, see chapter 'Social, Technology and Service triangle'.

Social, Technology and Service triangle

As the AGV makes its way through pedestrian traffic, it has to adjust its behavior to suit the situation. The underlying principle of the AGVs behavior comes from the Social, Technology and (Product) Service triangle, as shown in Figure 9. The triangle illustrates that when a situation demands a more social behavior, this will be on the expense of one or both of the other aspects. The exact balance is dependent on the specific situation and should be identified autonomously by the robot. Each aspect contains underlying factors which are inherent to that aspect and which govern the AGV behavior. The three aspects are used as an overarching framework in which the research and exploration findings are subdivided. Each subdivision concludes with a discussion and set of requirements derived from these findings.

Social

When talking about the social aspect, one can think of underlying factors such as social disruption, predictability, expression of intent, intuitiveness of the interaction and personal space. These underlying factors belong to two categories, social navigational methods and communication of intent. These two categories form the main chapters of the social subdivision.



Technology

When looking at the technological aspect, the underlying factors are for example; battery life, acceleration, velocity, suspension, turning radius and mass. In this subdivision the physical appearance and behavior of the AGV are discussed alongside with how they impact the technological performance. Next to this, the capabilities and the limitations of the AGV test platform, the Husarion ROSbot 2.0 Pro, are presented. Lastly, the theory and limitations of a common AGV machine learning method is discussed alongside the cost function buildup.

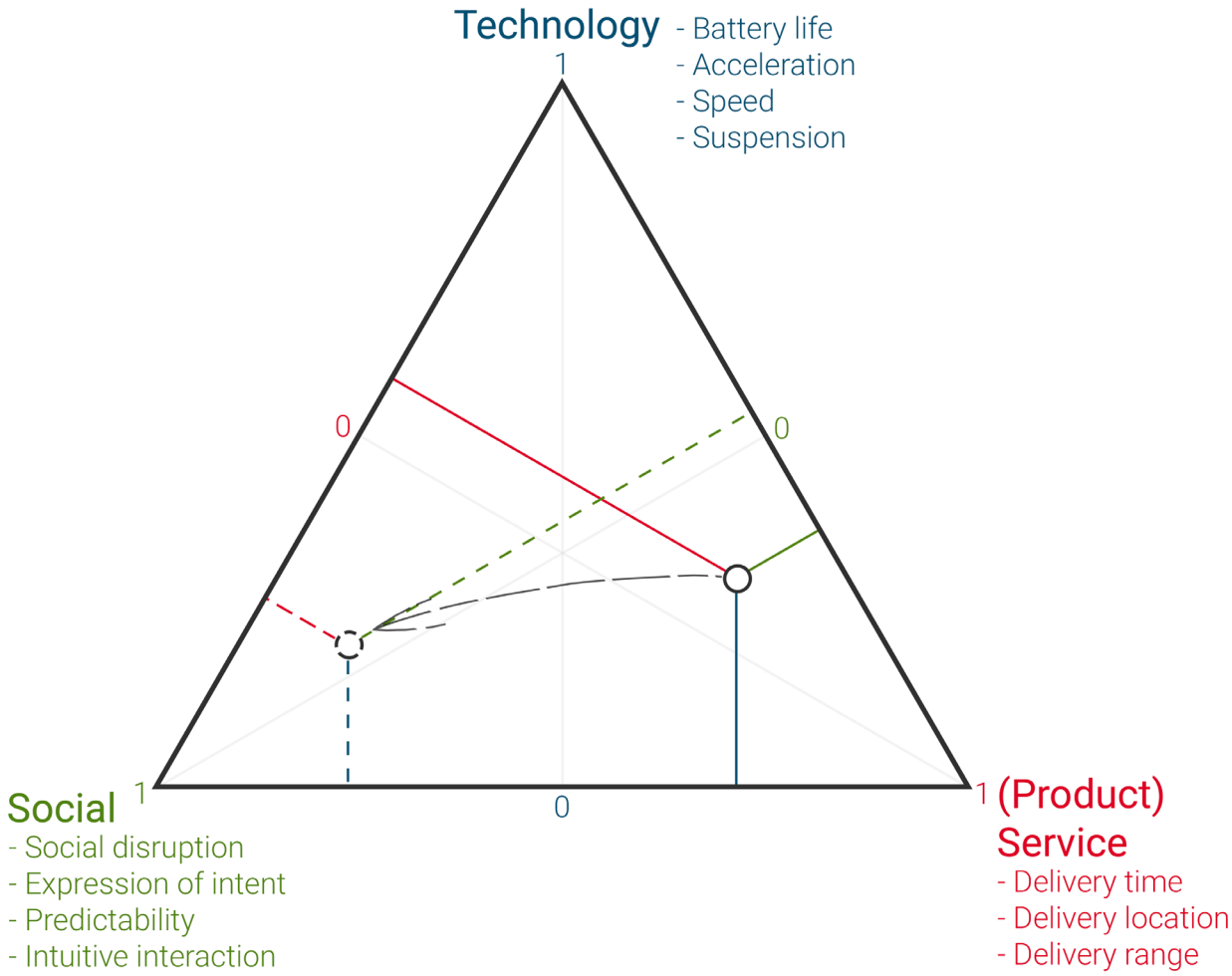
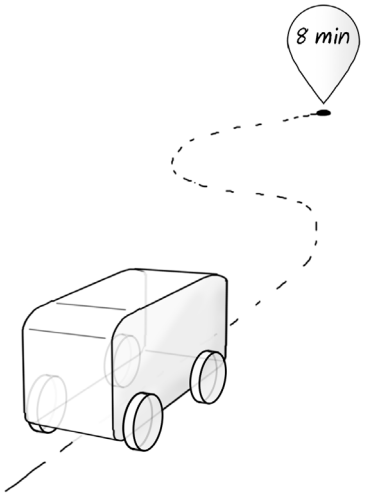


Figure 9. Social, Technology and (Product) Service triangle with accompanying example factors for each aspect.

Product service

In the third subdivision of the framework the product service aspect is discussed. This aspects has underlying factors such as delivery time, delivery location, effective delivery range and parcel compartment size. The desired service is provided by the AGV to the end user (the owner of the parcel that is being delivered) and in turn presents its own requirements. The AGV might be socially and technologically sound, but if it is unable to deliver parcels it will be useless as an autonomous parcel delivery robot.



Social navigational methods

According to Laffey & Amelung (2010) social navigation is defined as, "A construct that represents being aware of what others are doing as a primary guide for one's own actions." A social navigational method can then be defined as a tool or approach which is utilized in order to navigate while being aware of where others are and what they are doing. This chapter first discusses pedestrian wants and needs and how this can be simulated through a Social Force Model (Helbing & Molnár, 1995). Next it presents the findings of de Groot (2019) on Robot Acceptance within the context of pedestrian traffic and his simulations of the Social Force Model. Finally it presents two alternative socially aware navigational methods with their theory and underlying socially aware rules.

Social Force Model

Pedestrians have certain desires and characteristics while navigating among other pedestrians. These desires and characteristics govern their behavior, but their behavior also depends on the complexity they find themselves in. Helbing and Molnár (1995) have made the distinction of two complexities: 'simple or standard situations' and 'complex or new situations'. When complexity is low, the reaction is automatic, well predictable and can be modeled by a social force model. When complexity is high, the reaction is a result of evaluations and part of a decision process, it is probabilistic and can be modeled by a decision theoretical model. They argue however, that as pedestrians are normally used to the situations they are confronted with, their reaction can be described as automatic. They thus reason that it is possible to put rules to pedestrian behavior and model their motion. The following rules can be extracted from their paper (personal comments have been written in italic):

1. A pedestrian desires to reach a destination with as much comfort as possible, meaning they wish to follow the shortest path possible. *It is assumed here that the shortest path possible equals the most comfortable, but it doesn't take the state of the path into account. A longer path might be more comfortable if it means the pedestrian can avoid having to cross a rough surfaced road or can stay covered from undesired weather conditions.*

2. A pedestrian wishes to maintain its desired velocity and will accelerate to this desired velocity after a deceleration was required to, for example, avoid a collision. *This hints at a desired velocity for pedestrians, see also point 8, but it does not take the situation the pedestrian is in into account. The more crowded a place is, the lower the (desired) velocity. Similarly, one would increase the maximum desired velocity when the street is near empty. Another factor here that is overlooked is whether the pedestrian is in a hurry, or the opposite; enjoying the view of the city as a tourist.*

3. Pedestrians want to keep their distance to other pedestrians, depending on the pedestrian density and speed, this is called the private sphere. The closer another pedestrian is, the more uncomfortable they become. *The distance towards other pedestrians is important. Another factor might be the dominance and/or approachability of the other pedestrian. The angle between the two should also plays a role, see point 6.*

4. A pedestrian tries to maintain distance to borders such as walls, streets and

other obstacles. Their discomfort will increase when the distance to the border decreases as they will have to pay closer attention to avoid this border and prevent harm. *The distance towards said border is important. I suspect that the repulsive force range should be short but intense once within range. Repulsive force is also dependent upon the kind of border. You will stay further from a row of thorn bushes than from a row of common laurel bushes. The trajectory angle also is an important factor here, see point 6.*

5. Pedestrians can be distracted and become temporarily attracted to another pedestrian (e.g. a friend) or obstacle (e.g. a statue). *Helbing & Molnár also state that the strength of this attraction decreases over time. It is not mentioned if this decreases gradually or, what I would find more realistic, hyperbolic (steady at first but then dropping off rather quickly). It also doesn't mention the frequency at which this 'distraction' should occur.*

6. Situations behind a pedestrian have a weaker attractive and repulsive effect. *This does not include pedestrians on ones sides. People can walk side by side and not experience much sideways force. The forward direction should be full force, but it should rapidly decrease towards a fraction behind. What the specific value of that fraction is, is yet to be determined.*

7. The direction and velocity of the pedestrian are governed by the sum of all attractive and repulsive forces. *A side note here is; what happens when the sum of forces rapidly rotates 180 degrees? In real life the pedestrian might do a step backwards, but certainly wouldn't start walking backwards. Similarly, a pedestrian in motion won't do a 180 degree turn in a split second, especially not when in forward motion.*

8. The desired maximum velocity of a pedestrian is Gaussian distributed around a mean of 1.34 m/s with a standard deviation of 0.26 m/s. *This translates to ~4.8 km/h, which can suit as a starting point for the maximum velocity of the robot in situations where pedestrians are involved. But, as mentioned before, velocity should decrease as pedestrian density increases.*

The rules as stated above might seem abstract, de Groot (2019) helps us here with an illustration of a pedestrian situation with it's the social force vectors, see Figure 10.

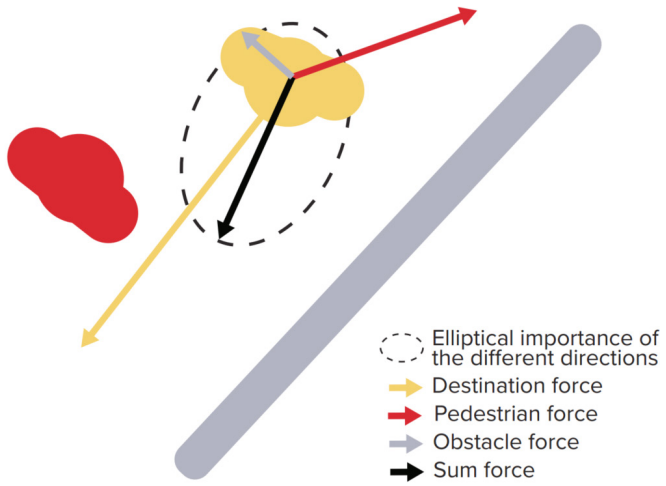


Figure 10. Social Force Model illustration by de Groot, S. (2019): "Example of a pedestrian situation and its social force vectors."

By using these rules to build a mathematical model, Helbing and Molnár were able simulate pedestrian traffic in a range of different scenarios such as walkways scaling from 2 to 20 meters, and the scenario of two larger groups trying to pass a narrow door in opposite directions. In the walkway scenario, pedestrians started forming lanes which number scaled linearly with the walkway width. In the narrow door scenario, pedestrians alternated passing in groups based on the balance of both sides and how easily on could follow the other ahead of them 'pushing through' the narrow gap. From these simulations and their results they conclude that individual pedestrian motion can be described by a simple social force model.

It should be noted that although the 'simple social force model' simulations could describe individual pedestrian motions, there are limitations as it is a simplified representation of the real world. Maximum desired velocity wasn't altered based on pedestrian density, pedestrians were allowed to make an instant 180 turn (during motion) or walk backwards for several meters. All border types were assumed equal, same for all pedestrians; approachability was not taken into account. All floors were considered ideal and desired as opposed to for example pavement being preferred over grass or sand. Lastly, the SFM does, by design, not implement any social conventions such as keeping to the right. It is understandable that (some of) these simplifications had to be made, but this leaves us with the question of how accurately the model can be used to simulate pedestrian behavior for robots in the real world.

Pedestrian behavior

Through direct observations of real life pedestrian situations, video material of pedestrian traffic and interviews with pedestrians, an understanding of the conventions in pedestrian dynamics can be created, see also Appendix 'C. Pedestrian observations' and Appendix 'D. Pedestrian interview and data'. The investigated situations are normal walking, passing, overtaking and crossing.

In a normal walking situation, the Dutch convention is that people remain to their right half of the walkway, analogous to vehicular traffic. The same 'right hand rule' analogy applies, to an extent, to passing and overtaking, see Figure 11 and Figure 12. For

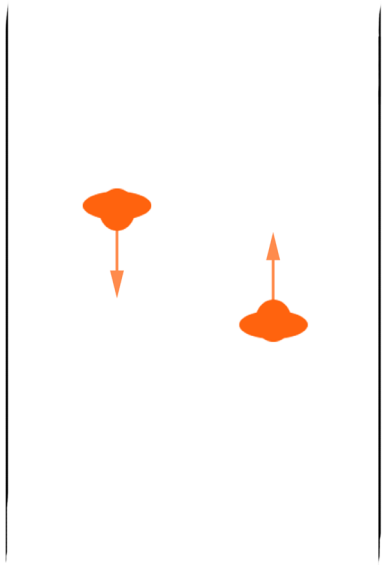


Figure 11. Passing.

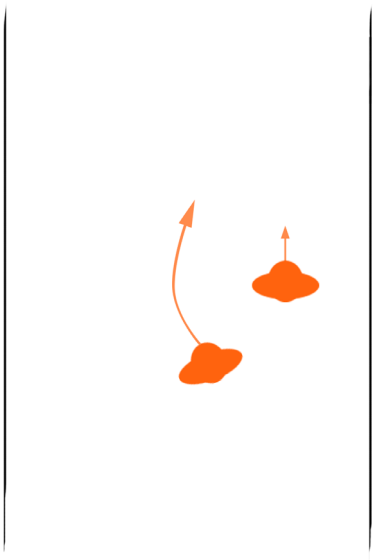


Figure 12. Overtaking.

the robot to behave as a pedestrian would, it would be wise to integrate a feature such as a right keeping bias into the social force model as to fit to (Dutch) social convention.
While passing, the phrase "people are supposed to stay on the right" is supported by all interviewees. Generally, people pass on the side they are mostly already on with regard to the 'to be passed' pedestrian. In most cases, when keeping right, this means you will pass each other on the right, having the other on your left. If a pedestrian keeps to the left even though they should keep to the right (according to convention), the opposite pedestrian will instead pass the left-keeping pedestrian on their own left. Against convention but forced by the opposite pedestrian. The exceptional scenario here where the strong headed pedestrian remains to their left could prove to be interesting to deal with later on when found in the social force model. The robot should 'see' that abandoning convention is preferred over blocking the strong headed pedestrian.

While overtaking, people speed up and alter their heading to their left (having the pedestrian they are overtaking on their right). This is again analogous to vehicular traffic rules. In rare situations pedestrians may choose to overtake on the right, against convention, when the balance of available space is shifted to such a degree that against convention is the preferred, more comfortable option. The exact value of this balance depends on the person itself and the situation that person resides in. Whether or not the robot is capable of deciding upon this balance is yet to be seen. An option might be to determine the most comfortable side to overtake based on available distance, conventional bias and visual clues, determined by an image classification neural net.

In the situation where the pedestrian is crossing another pedestrian, the vehicular traffic analogy falls short. Contrary to what the 'right hand rule' would dictate, there is no priority in crossing based on the relative position of the pedestrians. While crossing, pedestrians try to maintain their heading and speed unless this would bring them uncomfortably close. In that case they alter their speed and/or heading slightly to cross. Usually the one that alters the most is the one that crosses behind, see Figure 13. If a collision is nearly certain, speed and/or heading are greatly altered to avoid the collision. Again the one that alters the most is usually the one that crosses behind. Interviewees mention they do not give priority to the pedestrian on their right, or left for that matter. The decision is based on a more personal evaluation of the situation taking into account the others velocity, timing to crossing point, the empathy the other evokes and/or the level of dominance the other radiates. When the situation is exactly equal they will, "engage in a short, mostly non-verbal communication with the other." in

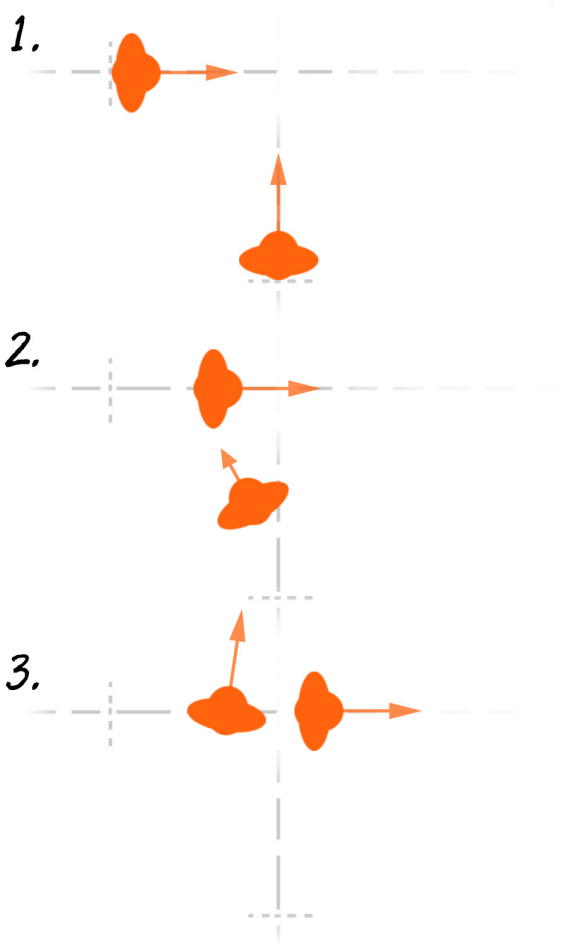


Figure 13. Crossing.

an attempt to solve the issue.
In the case of a pedestrian crossing the robot's path, the crossing must be comfortable for the pedestrian. This leads me to believe the robot should be submissive to the pedestrian; the robot should deviate more and pass behind, allowing the pedestrian to comfortably keep his heading and velocity. There are of course nuances to this. A pedestrian might be considerably slower in which situation it would be more comfortable for the pedestrian if the robot would cross (at a comfortable distance) in front. The reason for this is that the substantial deviation the robot would otherwise have to make could in itself also be considered uncomfortable by the pedestrian due to unpredictability or possibly a form of 'guilt'.

Robot Acceptance

According to de Groot (2019), "... pedestrian interaction should be low effort and intuitive. Pedestrians are not the customer and do not want to adapt their behavior dramatically to cope with tens of delivery robots on their sidewalk stroll". His tests show that favorable interactions could arise from a design in which the behavior of the robot is modeled after standardized pedestrian behavior. He states that pedestrians will feel in control when the delivery robot merges into the natural flow of pedestrian traffic and behaves predictable. Cues such as the orientation of the body and wheels, the leaning of the body as a reaction to acceleration and deceleration, and the path the robot is following are all assigned to making the robot interaction intuitive and lower the effort on the side of the pedestrian.

The social force model simulations as performed by de Groot in a virtual reality environment showed that the SFM is a first step towards desirable robot behavior, see Figure 14.

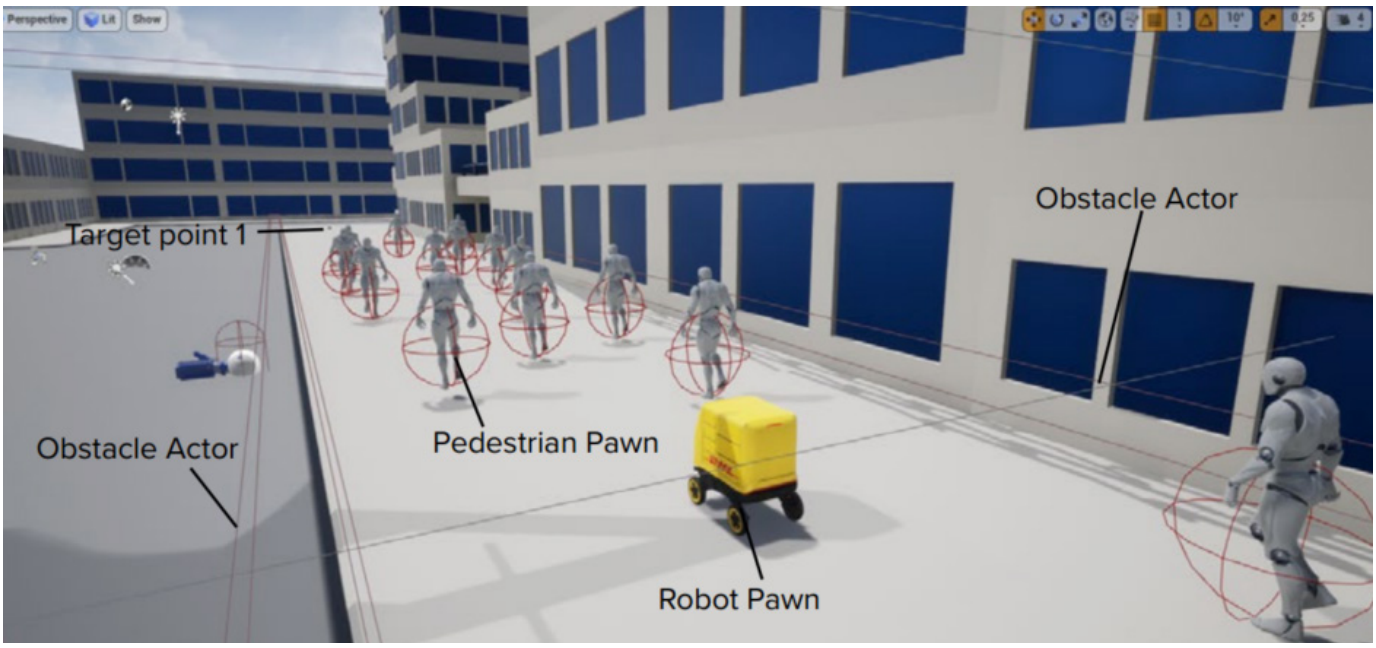


Figure 14. VR simulations by de Groot, S. (2019)

He concluded that his delivery robot should behave less dynamically then pedestrians do; steering slower, and moving with less changes of speed. It should communicate its intentions early, for example a few meters before encountering a pedestrian. De Groot's conclusion strongly reminds me of the way people operate mobility scooters through pedestrian traffic; is the concluded desired behavior similar to what you expect from someone who is controlling a mobility scooter?
De Groot added a few notes to take into account when looking at his implementation of the SFM;

- Overtaking was not tested
- Crossing was not tested
- The robot did not try to predict pedestrian behavior and the impact of available decision options was not taken into account.
- Communication can be seen as late, e.g. when the sidewalk is mostly empty pedestrians can already be communicating at a distance of ~10 meters.

Especially the last two notes are worthy of interest, the social force model is typically a model which operates on shorter distances and doesn't take pedestrian density into account. It also does not predict pedestrian behavior or analyze past pedestrian trajectories to determine their heading. The additions of looking ahead in time (predictions) and in space (low density long distance communication) could be very valuable in achieving a more intuitive and comfortable pedestrian-robot interaction.

Socially aware navigation with MPDM

Besides the SFM as implemented by de Groot, other alternatives are available. Mehta, Ferrer & Olson (2016) propose a method for socially aware navigation where the trajectory of the robot is not statically planned, but dynamically adapted based on a set of closed-loop behaviors in which the behavior with minimal 'cost' is selected; Multi-Policy Decision Making (MPDM).

In their setup, they allow the robot to choose from a set of policies which is formed by 'go-solo', 'stop' or 'follow other', in which the other can be any of the surrounding pedestrians. This results in a domain which ranges from 2 to 2+N possible policies where N is the number of available pedestrians to be followed. In the paper by Mehta et al. the people behind the robot are also considered as candidates 'to be followed'. I think it would be wise to exclude them from the domain in order to decrease calculation time. Anyone behind and moving in the opposite direction of the robots goal is not a viable candidate for progressing towards the goal.

In short, go-solo-policy applies the Social Force Model as described above where the robot is moving towards its goal while avoiding pedestrians and obstacles based on the 'social forces' exerted on the robot. The follow-policy uses the same Social Force Model, but here the goal is temporarily interchanged with another pedestrian. In case of the stop-policy, the SFM is ignored and the robot will decelerate to a stop applying a prescribed maximum deceleration force, see also Figure 15, Figure 16 and Figure 17.

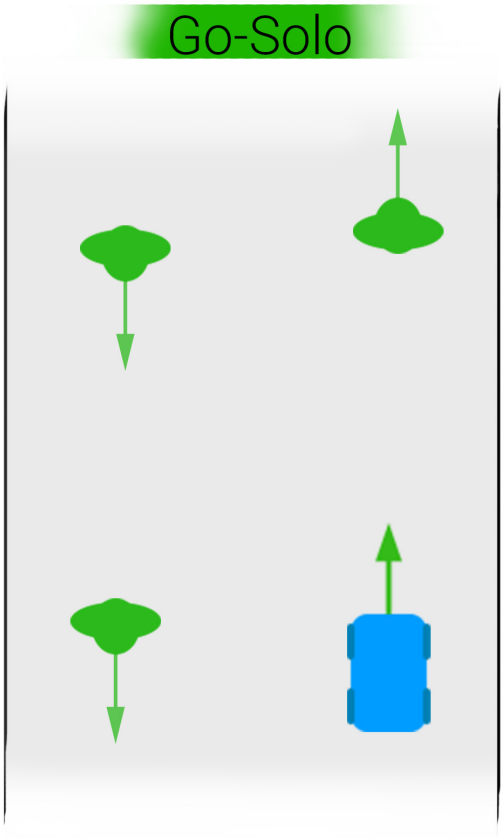


Figure 15. MPDM, Go-Solo policy.

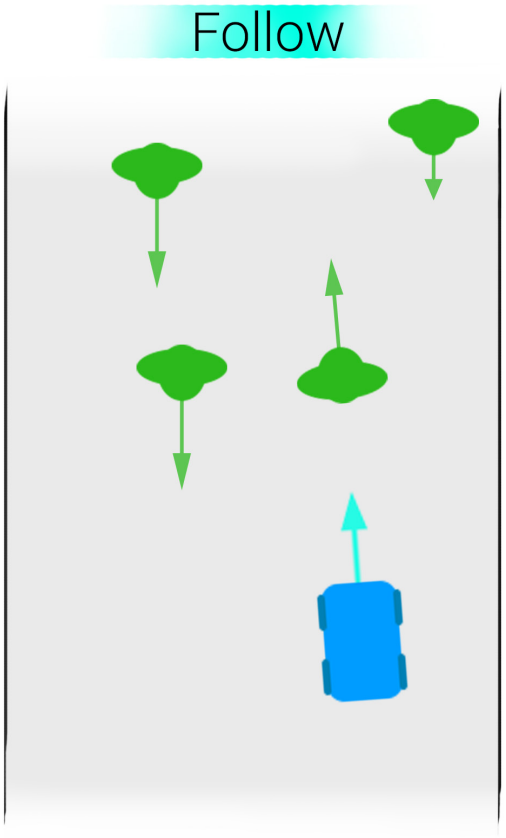


Figure 16. MPDM, Follow policy.

In order for the robot to know which policy to choose, the environment is sampled and used as an input to the MPDM. The MPDM repeatedly calculates and predicts the trajectories in a receding horizon fashion. A policy is chosen from available policies by an objective function which returns the policy with the minimal cost for all sampled agents. The objective function, or cost function, is build up from two parts: the potential disturbance the robot causes in the environment (Force) and the potential progress made towards the goal (Progress).

Progress is calculated as the 'distance-made-good' during the planning horizon. This means the difference between the position vector at the start towards the goal minus the position vector at the planned horizon towards the goal.

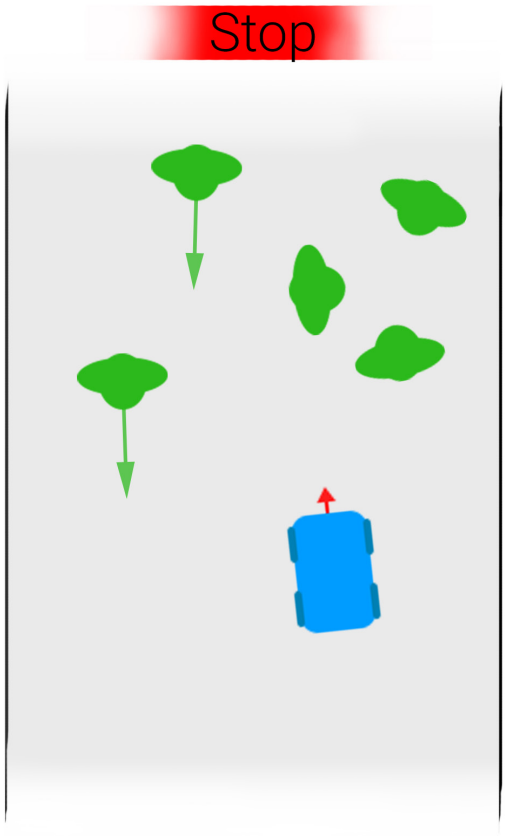


Figure 17. MPDM, Stop policy.

Force is the sum of all the maximum repulsive forces exerted on other agents (except the leader if in follow mode). *Note: You could state that using the maximum force is not the best solution to find the policy with the most disturbance caused in the environment. Instead I would propose to integrate the exerted repulsive forces over time and sum these up to calculate disturbance. This would help with picking a policy with mostly low disturbance and one short high peak over another where there is a constant and high amount of disturbance, but slightly lower than the aforementioned peak.*

The cost function is given as:

C(state, policy) = -alpha * PG(state, policy) + F(state, policy)

Where the state is the agent's/robot's combined position, velocity and goal vectors. Alpha is a weighing factor for the progress made.

By adding Multi-Policy Decision Making on to the SFM, Mehta, Ferrer & Olson (2016) have attempted to improve the robots performance of completing its navigational task of reaching the goal, while at the same time minimizing the inconvenience caused by its presence in the dynamic pedestrian environment.

Something to take into account is that the addition of MPDM to the SFM will strongly increase the number of calculations, especially in situations where many pedestrians are observed. Other factors which influence the number of calculations is the observation range, the frequency of deciding upon a policy and the length of the forward simulated horizon in seconds. It is to be seen what the hardware of the delivery robot can handle and what the optimal balance is between additional calculations (and thus the need for higher spec hardware) and improvement in the robot's socially aware navigational behavior.

Socially aware navigation with deep reinforcement learning

There is another common approach for social navigation, instead of using the Social Force Model. A robot can also be trained using socially aware deep reinforcement learning (Chen, Everett, Liu, & How. 2017b), see Figure 18. In socially aware deep reinforcement learning, an objective function is set up which evaluates the state the robot is in based on a set of rules. By stating what *not to do* based on social norms, a robot can be taught to avoid collision and navigate in a socially aware manner.

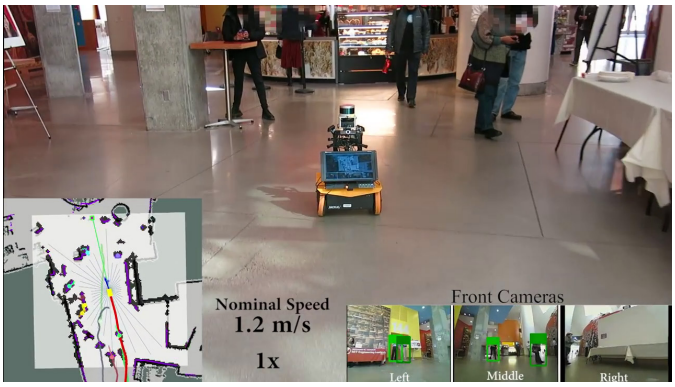


Figure 18. Socially Aware - Collision Avoidance with Deep Reinforcement Learning (SA-CADRL).

They formulate the collision avoidance problem as a 'sequential decision making problem' in a reinforcement learning framework. Where the agent's state s_t and the agent's action u_t at time t are taken into account alongside with the state of a nearby agent, $s^{\sim}_t$. The nearby agent state consist of an observable part s^o (position, velocity and radius) and a hidden part s^h (goal position, preferred speed and orientation). The agents action is set equal to the agents velocity, $\mathbf{u}_t = \mathbf{v}_t$. The goal is to develop a policy π: $(s_t, s^{\sim o}_t) \rightarrow \mathbf{u}_t$, that minimizes the expected time to goal $E[t_g]$ while avoiding collisions with nearby agents.

The right-handed social norms are induced through penalty areas for passing on the left, overtaking on the right and crossing with someone in front, see Figure 19.

Figure 19. Right-handed social norms and their penalty areas. Illustration by Chen et al. (2017b).

The image is translated to mathematical formulas describing the conditions, personal comments are written *in italics*:

The overtaking penalty area (blue) comes into effect when the agent is within 1 meter sideways (left), within 3 meters in front, the heading is within an angle difference of π / 4, the speed of the robot is greater than the speed of the agent and the goal is at least a distance of 3 meters ahead. *Although this is a solid method for teaching the robot to not have pedestrians on its left while overtaking, this also means the robot will never overtake or pass someone on the right like a pedestrian would in the case the aforementioned balance favors this as the more comfortable option (see chapter 'Pedestrian behavior'). The only time the robot would 'overtake' on the right is when the other pedestrian will be fully outside the blue penalty area, resulting in the robot making a wide path around the pedestrian.*

The passing penalty area (green) comes into effect when the agent is within 1 to 4 meters ahead, the agent is within 2 meters sideways (right), the angle difference is greater than 3π / 4 and the goal is at least a distance of 3 meters ahead. *Similarly to the previous note, this teaches the robot to follow pedestrian navigational conventions but penalizes in the scenario where a strong headed pedestrian keeps to the right of the robot while passing (against convention). The robot will thus make a stronger deviation than necessary in order to satisfy the learned rules.*

The crossing penalty area (gray) comes into effect when the agent is within 2 meters of the robot, the relative rotation angle between the agent and robot is greater than 0, the angle difference is between -3π / 4 and - π / 4 and the goal is at least a distance of 3 meters ahead. *A note here is that the robot will deviate from its trajectory to achieve the lowest penalty, even though realistically the robot should not have to deviate as the pedestrian is moving left and already to the left of the robots intended path. By the time the robot will arrive at the position the pedestrian was, he/she will already have moved on. The robot should of course respect the pedestrian's personal space, and thus not come too close or collide with the pedestrian.*

A deep neural network is fed the state of the robot and the observed states of three of the surrounding agents. It then uses a reward function based on the penalty areas as described above. An addition to the reward function are the additional rules of being rewarded for reaching the goal and being penalized for being too close to or colliding with other agents, as defined in the preceding paper by Chen, Liu, Everett, & How. (2017a).

After simulation training and validation of the trained model on test cases, the robot model is then loaded onto a real robot with LiDAR and four color cameras and physically tested in an indoor environment. The robot navigates according to social norms, overtaking on the left, passing on the right and waiting for others to cross before continuing while maintaining a maximum velocity of 1.2 m/s. *Despite the shortcomings as mentioned with the set up penalty rules, the robot appears to competently navigate through the pedestrian environments, as seen in their video. In this video, a new shortcoming becomes apparent; the robot is cutting corners, leaving no room for potential oncoming traffic, see Figure 20. This is due to one of the rules which was* not *taught to the robot; the default behavior of maintain mostly to the right of a walkway as per social convention.*

Deep reinforcement learning in pedestrian rich context is a sound method of social navigation, as proven by the results of Chen et al. The above description, however, only scratches the surface of the underlying mathematical models and the deep neural network setup for reinforcement learning. It is important to note that their work is a complex whole which builds upon their previous and equally complex work. Despite the complexity of their work, the robot still shows flaws in its behavior, as noted above in italics.

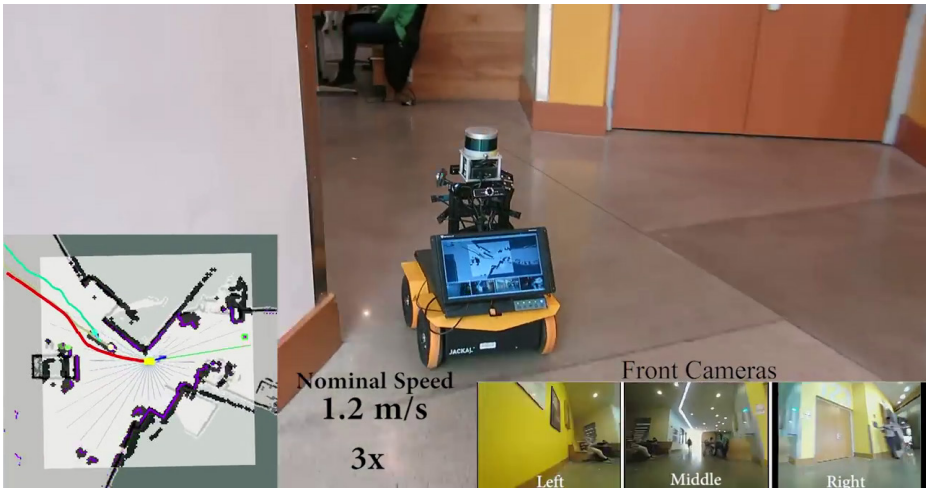


Figure 20. The robot is following right hand rules, but is not keeping to the right; resulting in the robot cutting corners and possibly running into someone hidden behind the corner.

Communication of intent

As mentioned earlier, the robot should occupy intuitive and comfortable behavior for human-robot interactions. Part of this behavior is the expression of its intentions. This chapter first discusses the awareness and intent communication in autonomous vehicle-pedestrian interactions (Mahadevan, Somanath, & Sharlin. 2018). Next it looks into the usage of movement in intent communication as applied with Assistive Free Flyers (Szafir, Mutlu, & Fong. 2014). After which an overview is given of existing autonomous delivery robots and their methods of intent communication. Finally the findings of an investigation in robot predictability are discussed as presented by de Groot (2019).

Why communicate intent?

The communication of intent is inseparably connected to an intuitive and comfortable human-robot interaction. According to Risto, Emmenegger, Vinkhuyzen, Cefkin, & Hollan (2017), pedestrians experienced discomfort when confronted with a vehicle which did not communicate awareness nor intent. Pedestrians use the robot's communicated intent to make predictions of the robot's planning and as an input to their own planning.

Awareness and intent in autnomous vehicle-pedestrian interaction

In regular vehicle-pedestrian interaction, the driver can communicate its awareness of- and the intent to the pedestrian through facial expression, eye gaze, eye contact, hand gestures, body movement, light signals, vehicle speed, vehicle position and audio. In the case of autonomous (and thus driverless) delivery robots, many of the communication methods are not available. This means that an AGV navigating on the sidewalk must have additional interfaces and/or stronger vehicular movement cues in order to compensate for the lack of driver related communication. The paper of Mahadevan et al. looks into the usefulness of interfaces (besides the vehicle movement) which are designed to communicate awareness and intent. The scenario the paper focused on was between an autonomous car and a crossing pedestrian on a crosswalk.

According to Risto et al. additions to autonomous vehicles for vehicle-to-human communication are insufficient due to shortcomings in visibility and understandability (light conditions and on vehicle positioning, non-intuitiveness and complexity). Instead, they argue that *"movement in context is a central method of communication for coordination among drivers and pedestrians."* Drivers purposefully communicate their intentions through vehicular position, speed and acceleration. They suggest that (interface) designers should take into account that human drivers and pedestrians communicate via vehicle movement.

Mahadevan et al. expand onto and oppose this suggestion by stating that designers should not rely on the vehicle movement *alone* but should also include interfaces to help communicate awareness and intent; *"Designers should consider autonomous vehicle movement patterns as a key layer of interaction with pedestrians, providing baseline information that should be reinforced by other explicit communication cues."*

The literature shows a disagreement on whether the additions of interfaces is a must (Mahadevan et al.) or if they will be insufficient due to the shortcomings in visibility and understandability. It is clear however that

vehicle movement such as vehicle position, speed and acceleration is key in the communication of intent and awareness.

Usage of movement in Assistive Free Flyers

Within the context of Assistive Free Flyers (AFFs), Szafir, Mutlu & Fong, (2014) have investigated the impact of the alteration of an aerial drone's movements on the communication of its intent. One of those alterations was the ease in - cruise - ease out, which meant the drone would slowly ramp up the speed to cruising and slowly decelerate in advance before reaching the goal. This was seen as predictable and comfortable and has potential to be translated from aerial drone to AGV. Other alterations where the use of arcs instead of straight paths and the use of 'anticipation' where the drone would first fly a short distance in the opposite direction of its goal before moving towards it, see Figure 21. The 'easing in and out' is an interesting mechanic which might be achieved in ground based AGV by limiting the acceleration, deceleration as well as the jerk (change of acceleration).

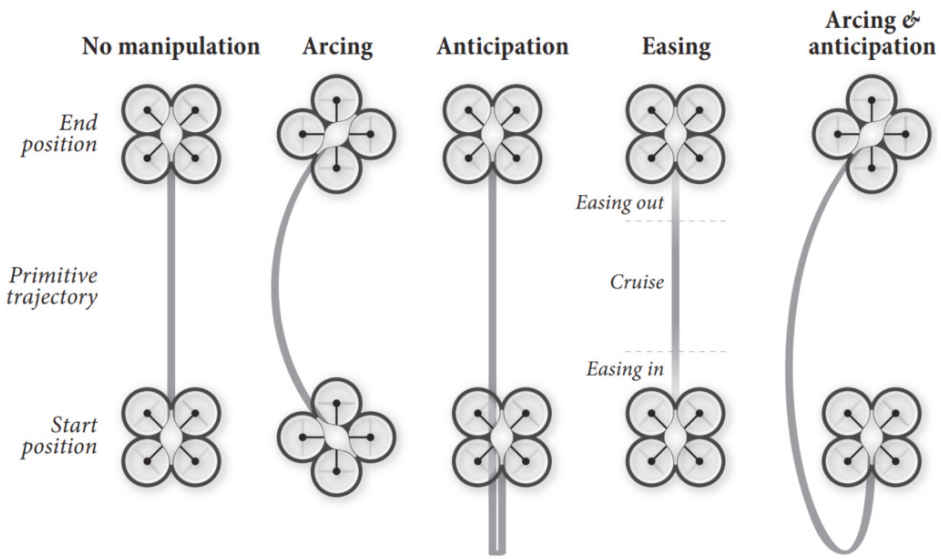


Figure 21. Movement manipulations in Assistive Free Flyers.

Existing robots and their intent communication methods

There are already quite a number of AGVs in development and on the road. The intent communication methods of these AGVs can be looked into in order to get a grasp of the communication approaches already implemented.

Starting with the AGV that was most preferred according to de Groot's user benchmarking; the Starship delivery robot (Starship Technologies, 2018), see Figure 22.

Figure 22. Starship Technologies delivery robot.

The Starship robot utilizes:

- Red LED strips for tail lights, always on.
- Orange flag on a pole with orange LEDs for increased visibility (the robot is relatively short).
- White headlights, low down and in front.
- On the spot turning due to its wheel configuration and individually controlled wheels.
- It indicates the storage area can be opened by a green LED strip on the back in between the red tail lights.
- Slows down with pedestrians and/or obstacles nearby.
- Angles itself towards the direction it wants to go to with its body, but its wheels do not 'steer' by angling themselves.
- When it's blocked in its path by a pedestrian for longer than 6 seconds, it might politely ask *"Excuse me, would you please let me pass?"* through speakers.

The Starship robot thus uses the core vehicle movement method of intent communication in combination with visual cues such as brake lights and auditory cues such as playing pre-recorded audio messages.

Next is the Marble delivery robot (Marble, 2018), see Figure 23.

Figure 23. Marble delivery robot.

The Marble robot utilizes:

- Red tail lights, always on.
- Front white LED strip as headlights, always on.
- Slows down with pedestrians and/or obstacles nearby
- Front wheels turn in the direction of movement.

The Marble robot is simpler in its intent communication compared to the Starship robot. It uses the core vehicle movement mechanics, but does not use visual signaling or auditory messages. What is interesting here is the front wheels which turn in the direction of desired movement in order steer, similar to a car. This provides extra cues to the pedestrian such as the turning radius and if the robot will continue to turn/steer.

There are two more robots with similar intent communication setups. There is the Domino's Robotic Unit, DRU (Domino's, 2016) and the Amazon Scout delivery robot (Amazon, 2019), Figure 24 and Figure 25. These also employ core vehicle movement as their method of communicating intent. In addition they have head and tail lights for communicating braking, but no other additional interfaces for communicating, for example, steering intent. These robots also steer through differential drive instead of angling the wheels. On one hand this is useful for minimizing the turning radius which in turns allows the robot to turn on the spot and maneuver in cramped areas. On the other hand it can be seen as less predictable by the surrounding pedestrians as there is no use cue for how strong and for what duration the robot might be turning.

Figure 24. Domino's Robotic Unit, DRU.

Figure 25. Amazon Scout delivery robot.

On the other side of the spectrum are the delivery robots employing (touch) screen interfaces to symbolically or literally communicate what their intentions are. The robots which use screens are the Kiwibot (Kiwi Campus, 2018), the FedEx Sameday bot (FedEx, 2019a) and the Postmates Delivery Robot (Postmates, 2019), see Figure 26, Figure 27 and Figure 28. The screens are not used to communicate steering intent however. They are mainly used as a way to 'greet' users and, in the case of the FedEx Sameday bot, indicate driving or stopping.

Figure 26. Kiwibot, on campus delivery robot.

Figure 27. FedEx Sameday bot.

Figure 28. Postmates Delivery Robot.

There are only two delivery robots which indicate steering with more than just core vehicle movement. These are the earlier mentioned Postmates Delivery Robot and the ZMP CarriRo (zeropointmoment, 2019). The Postmates robot has two eyes which function as headlights but which will also look in the direction of (imminent) steering. The same applies for the CarriRo, which, in addition to eyes, also uses side-mounted orange blinking indicator lights and voice to communicate corner taking, see Figure 29.

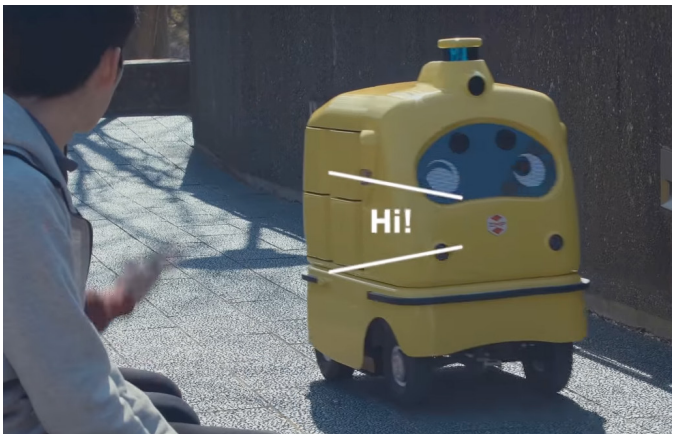


Figure 29. zeropointmoment CarriRo.

Out of all the aforementioned robots, the ZMP CarriRo is the one which communicates intent in the most expressive way (eyes, head and tail lights, orange indicating light, core vehicle movement). It also audibly greets pedestrians, has very dynamic eyes which look around, wink and change expression. The communication behavior of the CarriRo can be seen as obtrusive and annoying, but can perhaps be explained as a difference in culture; the CarriRo is designed to operate within a Japanese context. Within a more western context the employment of such an extrovert robot might prove to be unwise considering its task is to deliver goods and limit social disruption, not socialize and create it.

To sum up the overview of present intent communication methods; all robots natively use core vehicle movement. Some of them use just the main body movement, while others also have the cue of angling steering wheels added to their communication through vehicle movement. Many utilize red braking lights, though it could not be confirmed with all robots if these would light up when actually braking. A few robots have added eyes and/ or screens. Two use auditory messages, of which one only when blocked, while the other uses them haphazardly.

It has become apparent that core vehicle movement is vital as a basis for communication of intent. A nuance within the vehicle movement is whether or not the wheels angle themselves or if a differential drive is used for steering. Angling wheels seem to provide additional cues, but also technically limits the robot by introducing a turning radius. Furthermore, it is interesting to see is if vehicle movement is enough to communicate intent or whether something should be added. And what this 'something' might be. One of the first additions could be white head and red tail lights for visibility and the communication of (imminent) braking. Lastly, it is noteworthy that none of the aforementioned robots seem to use self-induced leaning or pivoting to communicate an (imminent) change in movement.

Robot predictability through special indicators

According to de Groot's research on the influence of special indicators on the predictability of the delivery robot, the use of red braking lights was easily associated with the robot slowing down. After only a few encounters with the robot in the virtual reality environment, pedestrians were able to predict that the robot was about to stop. However, on a few occasions, a double meaning was given to the red lights. Sometimes pedestrians interpreted them as emergency lights due to the unexpected position (lighting up the floor underneath the robot) and the intensity of the lights, see Figure 30. I think it would be wise to keep the head and tail lights as similar as possible to the way they are arranged and used in cars. This should help prevent confusion and make the interaction as intuitive as possible.

Figure 30. Floor lighting of the de Groot delivery robot (in this case white), left image. And the similar situation with the DRU delivery robot, right image.

Next to lighting as a method of communicating intent, de Groot also investigated pivoting. Before accelerating, decelerating or corning taking, the body of the robot would lean in the direction of imminent movement as a way of communicating near actions, see Figure 31. The pivot was experienced as too dramatic however. The artificial pivot was performed early, after which the natural pivot happened due to the movement. The resulting pivot, as the sum of the two, was seen as too intense. De Groot suggests to keep the early artificial pivot and make it a little more subtle, after which the natural pivot should be artificially countered in order to limit the overall experienced intensity. Whether or not the use of leaning is something which effectively communicates intent, and how it impacts the social perception of the robot, should be further investigated in this project.

Finally, de Groot also employed orange blinking indicator lights in his (virtual) design. The main purpose of these lights was not to signal every bit of steering, but only to communicate large changes. I think it is wise to not use the indicator lights for smaller movements, additionally; you might even think to not use them at all, as pedestrians also do not communicate large changes of direction in such a way. An aspect which is mostly under lit in all of the above sections is the aspect of sound. One could expect a clear association between a pitch shifting up with an increase in velocity and a pitch shifting down with a decrease in velocity. The usage of audio in a pitch-shifting way is another promising method of communicating intent, but care should be taken as sound is something which can quickly become obnoxious as well. A note here is that many of the methods for communicating intent are not analogues to those of pedestrians, but instead are more derived from (autonomous) cars. This connects back to two more overall questions; to what extent should pedestrian behavior be mimicked and is there a point where the pedestrian like behavior becomes too human or too limiting and starts producing counteractive effects?

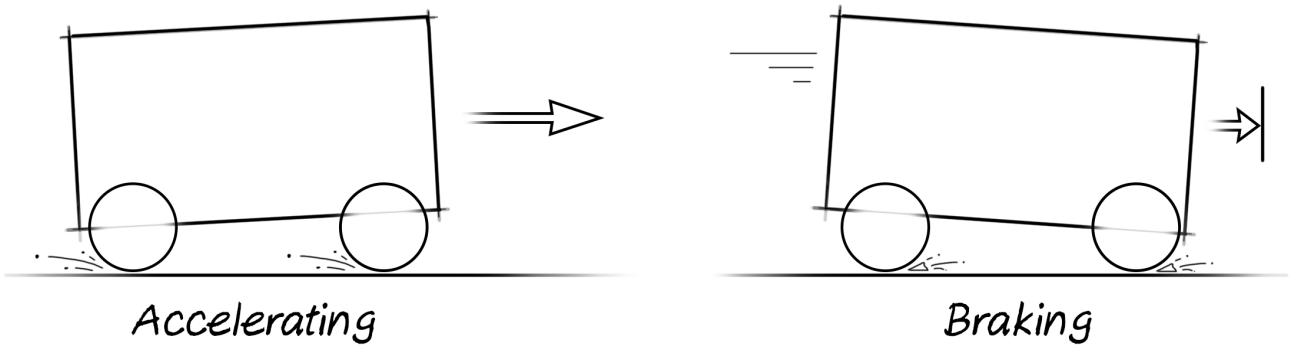


Figure 31. Pivoting movement of the AGV when accelerating and braking.

Discussion and requirements

In this chapter the discussion points of the previous chapters are discussed and requirements and/or wishes are presented from these discussion points. The requirements and wishes can be found in chapter 'Program of Requirements and Wishes'. In this discussion they are referred to with R. SX for a requirement and W. SX for a wish where the X is replaced with its index number.

From the 'Situations in pedestrian traffic' chapter it became clear that the robot should change its behavior based on the situations it finds itself in; R. S1. This is closely tied together with the finding that the robot should occupy intuitive (predictable) and comfortable (ease of use) behavior; R. S2. It should thus express intentions, take personal space into account and minimize social disruption; R. S4, R. S11 and R. S12. De Groot's tests show that favorable interactions could arise from a design in which the behavior of the robot is modeled after pedestrian behavior; R. S3, R. S5, R. S6, R. S7.

It also became apparent that there are certain shortcomings in the social force model. For example, the shortest path isn't always the most comfortable path, this also depends on the path's surface and other environmental factors such as weather; R. S13. Additionally, it does not model the effect of the dominance or approachability of a pedestrian on the strength of the repulsive force. It assumes all borders are equal, not taking into account the nature of the border (e.g. a row of thorn bushes versus common laurel bushes). Both of these are reflected in W. S1. Another shortcoming is that of pedestrian turning speed; a pedestrian in motion won't do a 180 degree turn in a split second; W. S2.

From the 'Multi-Policy Decision Making' and 'Robot acceptance' chapters it became clear that the robot should take pedestrian density into account in order to adjust parameters such as maximum desired velocity and the size of the pedestrians' personal spheres; R. S1. The robot should look ahead in time by predicting pedestrian trajectories in order to find the most comfortable but at the same time most progressive trajectory for itself; R. S8. The robot should also look ahead in distance, especially when pedestrian density is low, in order to timely adjust its trajectory, which could achieve a more intuitive and comfortable pedestrian-robot interaction; W. S5. While following, the robot should respect the pedestrians comfort by not tailing the pedestrian too closely or for too long to forego the uncomfortable feeling of being followed; R. S9 and R. S10.

As mentioned before, the robot should be able to communicate its intentions. The core of which is the vehicle movement itself; R. S11 and R. T2. In addition to this, it was found that less abrupt movements (which can be seen as chaotic) and thus smoother movements have a positive effect on robot predictability and pedestrian comfort. It is theorized that this could be achieved by limiting the acceleration, deceleration as well as the jerk (change of acceleration); W. S3 and W. S2.
The ZPM CarriRo robot demonstrated a large amount of self-initiated social interaction. In a more western context this self-initiated social behavior is undesirable, especially when considering its task is to deliver goods and limit social disruption, not socialize and create it; R. S12. Lastly, it is interesting to see is if vehicle movement is enough to communicate intent or whether something should be added. I think it would be wise to keep the head and tail lights as similar as possible to the way they are arranged and used in cars. This should help prevent confusion and make the interaction as intuitive as possible. One of the first additions should thus be white head and red tail lights for visibility and the communication of (imminent) braking; W. S4 and R. T1.

Physical (and dynamical) parameters and their impact

The most important operational aspect of the AGV is energy. Energy capacity and energy consumption govern the effective delivery range and the operational duration (on time) of the robot. In this chapter, the physical and dynamical parameters and how they impact the AGV's performance are investigated.

The first and foremost parameter that should be controlled is AGV mass. A lightweight robot driving at the same velocity as a heavyweight robot has much less kinetic energy. The larger the mass of the AGV, the more energy it costs to attain a certain velocity and this energy has to come from the batteries. It is therefore vital to keep the AGV lightweight.
Another way to express this is, is that the 'energy capacity over mass' is a ratio which should be kept high. The rate of energy consumption will be directly noticeable in the effective range of the AGV. If one desires a considerable range, they should have a lightweight AGV with a high capacity over mass ratio. This reasoning can also be flipped around; the desired range can also be used to determine the battery capacity and AGV mass combination. Exact values are dependent upon the individual setups and wishes but the above mentioned rules should be used as a guideline.

A more specific mass related parameter is that of the wheels. Larger wheel mass is tied with larger wheel inertia which in turn results in more work for the motors to accelerate those wheels. This could be especially noticeable when taking corners while using a four wheel differential steering (skid and steer) over an Ackerman steering approach. At the start of taking a corner, the inner wheels have to slow down while the outer wheels have to speed up. Similarly, at the end of the corner, the outer wheels must slow down while the inner wheels have to speed up, see Figure 32. This speeding up of the wheels in corner taking costs more energy compared to driving in a straight line. Additionally, the driven path of a weaving robot is longer than that of a robot driving in a straight line, further increasing energy costs.

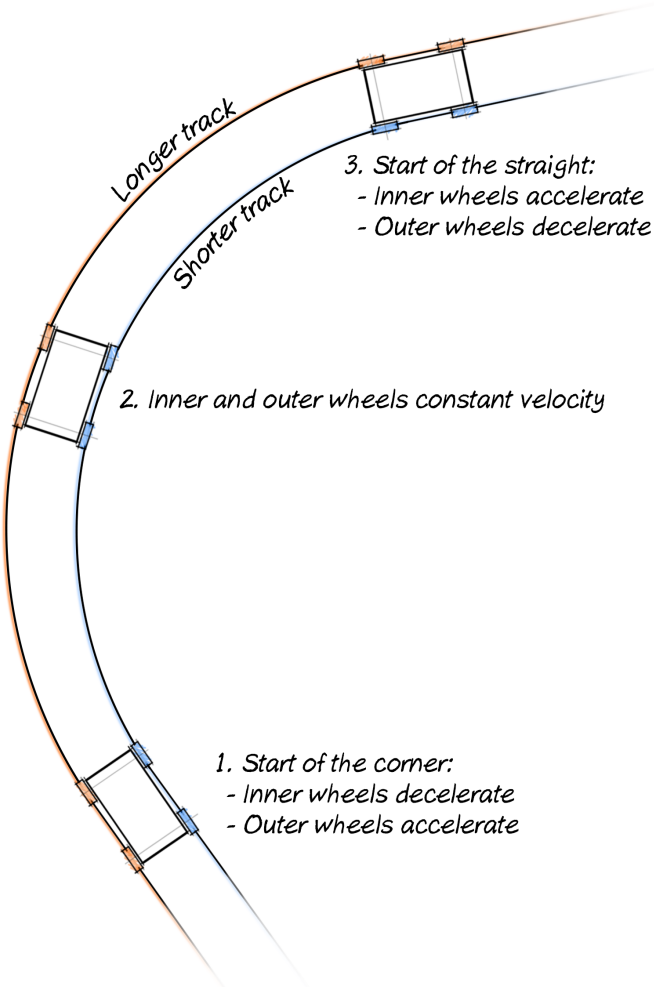


Figure 32. Corner taking with a differential drive (skid and steer).

Besselink, Wang, & Nijmeijer (2013) and Wang (2016) showed that the driving velocity in a battery electric vehicle (BEV) has a large impact on energy consumption. It was shown that energy consumption grows exponentially and range decreases with driving velocity in BEVs, see Figure 33 and Figure 34. Interesting to note here is the asymptotic behavior of the graphs at very low speeds. These effects were explained by Besselink et al. due to higher rolling resistance at low speeds and lower drive train efficiency at reduced power levels. Whether the general plot holds true when translated to the low speeds of delivery robots in pedestrian rich environments is uncertain, but for sure it can be said that operating on higher speeds drains the battery more rapidly.

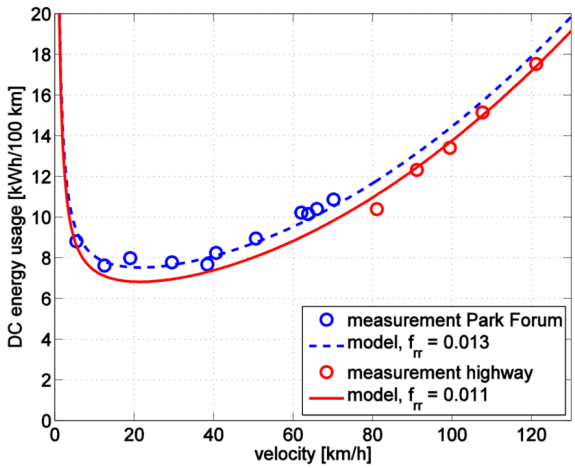


Figure 33. Constant velocity - energy consumption, Besselink et al. (2013).

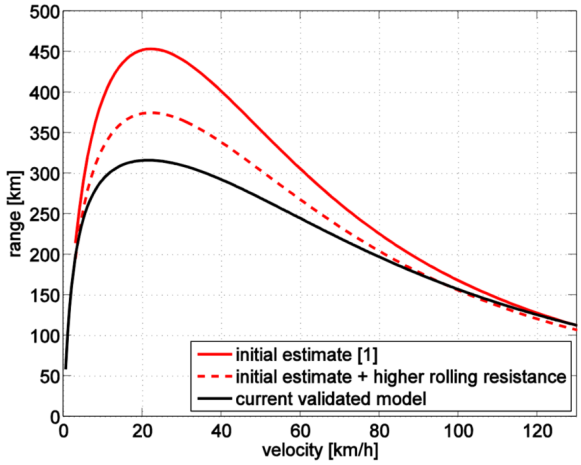


Figure 34. Constant velocity - range, Besselink et al. (2013).

The road conditions are also mentioned by Wang (2016). The coarser the roads, the higher the rolling friction and thus the larger the energy consumption at equal speeds. The same applies for driving up slopes, except here the added force comes from the force of gravity on the car in combination with the angle of the slope.

The ambient temperature has a strong effect on the performance of batteries. Battery University (2019) mentions that a battery functions best near room temperature. Increasing battery temperature improves acute performance but prolonged exposure will shorten overall battery life. They state that *"Cold temperature increases the internal resistance and lowers the capacity. A battery that provides 100 percent capacity at 27˚C (80˚F) will typically deliver only 50 percent at –18˚C (0˚F). The momentary capacity-decrease differs with battery chemistry."* This implies that the effective range of the delivery robots will decrease as temperature drops, see also Figure 35. It also hints at more permanent decrease in battery performance when exposed to higher temperatures, which in turn also impacts effective range and operational duration.

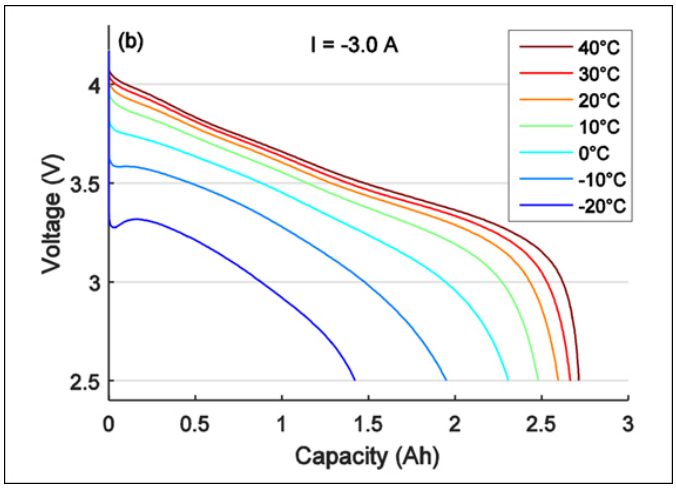


Figure 35. Battery capacity and voltage at different temperatures.

Another important factor is that of motor efficiency and internal friction inside the AGV. These affect the chemical to mechanical energy efficiency ratio which can be seen as a hard percentagewise reduction of the AGV range and operational time. Greater internal losses, e.g. due to axle friction, motor efficiency and internal resistance, results in a lower overall efficiency. In addition, in an ideal world the wheels would have zero slippage with the ground. Realistically, some slippage will always occur and this again subtracts from the AGV range.

Finally there are the intent communication actuators, main functional sensors and the controller to consider. Actuators such as an animated suspension and head and tail lights, sensors such as LiDAR and RGB-D, and the controller as required for operation all require electricity to operate. The usage of the main functional sensors and the controller are unavoidable, but can be picked out or set up in such a way as to limit energy consumption and in turn increase range and/or operational time. The energy consumption of the intent communication actuators is something which is mostly unknown and depends strongly on which intent communication methods turn out to be successful. However, one can expect that suspension actuators will cost more energy to use than lights and sounds might cost.

In short, the battery drains faster when energy consumption is high, with a more rapidly draining battery comes a decreased effective delivery range and a decreased operational time. The AGV's range and durational performance is decreased in cases where:

- The robot is not optimized for low mass.
- A fast driving speed is to be maintained.
- A very slow driving speed is to be maintained (due to relative high roll resistance and reduced efficiency in lower power levels).
- The robot decelerates and accelerates regularly.
- The robot is driving up slopes.
- The robot is steering a lot:
  - By lengthening its path more energy used.
  - In case of a differential drive; the motors have to slow and speed of the wheels each turn, here wheel mass also comes into play.
- The surface on which the robot is driving is rough (gravel, sand, potholes).
- The wheels of the robot have a large amount of slippage.
- The ambient temperature is lower than the ideal battery temperature (such as ~20 degrees Celsius).
- Sensors and actuators require a large share of the power.

Some of the above mentioned conditionals can already be tackled in the physical design phase, such as robot mass, wheel slippage and sensor and actuator power consumption. Others can be seen as part of the robot (designed) behavior, such as driving speed, steering and regular deceleration and acceleration. And finally there are some environmental conditionals which you cannot chance, but which you can try to deal with, such as slopes, road surface roughness and ambient temperature.

Capabilities and limitations of the ROSbot 2.0

As the ROSbot 2.0 Pro (see Figure 36) will be the physical platform for this project, it is important to know its capabilities and limitations. This chapter first lists the specifications of the ROSbot and then discusses them with their respective impact.

Figure 36. The Husarion ROSbot 2.0 Pro, front view (left) and back view (right).

ROSbot specifications

The Husarion ROSbot 2.0 Pro has the following specifications:

Dimensions with camera and LiDAR:	200 x 235 x 220 mm [L x W x H]
Weight with camera and LiDAR:	2.84 kg
Wheel diameter / Clearance / Wheelbase:	85 mm / 22 mm / 105 mm
Maximum translational velocity:	1.25 m/s
Maximum rotational velocity:	420 deg/s (7.33 rad/s)
Maximum load capacity:	10 kg
Battery life:	1.5h - 5h

- CORE2-ROS controller with UP Board
- Intel® Atom™ x5-Z8350 64-bit processor with 4 GB of RAM, 1.92 GHz, 32GB eMMC
- Intel® HD 400 Graphics
- Orbbec Astra RGBD camera
- RPLIDAR A3 laser scanner (max 25 meters, 16000 samples per second, specialized indoor and outdoor modes).
- MPU 9250 inertial sensor (accelerometer, gyro and compass)
- 4x VL53L0X time-of-flight distance sensor
- 3 x 3500 mAh Li-Ion batteries with protection circuits

Discussion of the specifications

One of the foremost limitations of the ROSbot are its small dimensions. They are nowhere near the dimensions of what the delivery robot should be and this in turn influences the way the robot is perceived. This means either the robot's body should be customized to match that of the delivery robot or any physical test conclusions taken, should be taken with the disparity into mind.

A second limitation is that of wheel clearance. The robot only has a wheel clearance of 22 millimeters from the bottom of the coach work to the floor. This, in combination with the strong recommendation to not use the ROSbot in outside weather conditions results in a test setup which will mostly partake indoors.

The maximum translational velocity of 1.25 m/s is actually quite on point with regard to simulating walking speed. A limitation is that the robot will not be able to perform a short print of increased velocity to overtake a pedestrian which walks just slightly slower.

The maximum rotational velocity easily accommodates desired turning speed. In fact, care should be taken to not turn the robot too quickly for this could have negative consequences for any nearby pedestrians' comfort.

The maximum load capacity of 10 kg allows for physical expansion onto the robot, such as building a to-scale foam-board representation of the designed delivery AGV on top of the ROSbot.

The 4 GB RAM, 1.92 GHz CPU should suffice for running the SFM, but care should be taken when using MPDM as well as to not overload the hardware. Aspects which influence this are the number of observed pedestrians (which is also tied with observation range), simulation frequency and the length of the simulation horizon.

A similar situation comes from the Intel® HD 400 graphics card, it is to be seen whether the (tiny) Yolo v3 object classification algorithm can be run on the graphics card alone or if part of the CPU should be shared for this, further limiting other parts of the software. One option for both aforementioned limitations might be to use the ROSbot purely as an 'sensor' and 'actuator' while the full computations are performed remotely on a capable PC where information is then communicated to and from using the ROS master-node network.

The Orbbec RGB-D camera and RPLIDAR A3 are both fully capable of viewing the surroundings and detecting objects. A point of attention is that the RGB-D camera is forward facing, and thus aid in the localization of objects and pedestrians on the sides and behind the robot might be required. Secondly, the RPLIDAR only scans in a horizontal plane, anything below (or above) this plane is cannot be scanned and is thus not 'seen'.

Finally, the additional infrared ToF distance sensors and the MPU inertial sensor might prove to be nice additions but as of yet are not planned to be used.

Machine learning with AGVs

In machine learning it is often the case that an algorithm tries to find the best action or best solution for a given state or task. From this arises the question of how to determine what is 'best'. This is discussed in the first chapter, 'Reward and penalty parameters'. The second chapter, 'Evolutionary algorithms', concerns itself with a machine learning method which tries to emulate genetics and Darwinian evolution to find an optimal solution. It discusses the theory behind that particular machine learning approach along with the many options within evolutionary algorithms.

Reward and penalty parameters

As explained before in the chapter 'Social navigational methods' a certain cost or objective function has to be set up for certain machine learning (ML) algorithms in order to evaluate themselves. The cost functions are divided up in penalties and rewards.

A penalty is there to penalize the ML algorithm for negative occurrences. The goal is to have as few penalties as possible and thus to minimize cost. Examples of negative occurrences within the context of a socially navigating AGV are:

- Social force exerted on pedestrians (social aspect); this is the amount of discomfort the pedestrians feel by the presence of the robot.
- Collisions with pedestrians (social aspect); any actual physical contact should be heavily penalized.
- Delivery time (product service aspect); the longer the delivery time, the larger the penalty.
- Driven path distance minus direct path distance (product service aspect); any additional distance driven takes away from the pre-calculated range.
- Amount of time spent accelerating and decelerating (technology aspect); as mentioned in the previous chapter, braking and speeding up costs more energy than driving with a consistent speed.

A reward is the opposite of a penalty and thus is considered as negative cost. Rewards are given for positive occurrences. The goal here is to have as much rewards as possible to again minimize cost. Examples of positive occurrences within the context are:

- Reaching the goal (product service aspect); the robot gets rewarded for reaching its end goal.
- Progress (technology aspect); the distance made good between time intervals

Each reward and penalty also has a factor with which it is multiplied. This factor or weight signifies the importance of that reward/ penalty. The weight of each penalty is to be determined dependent upon the situation the robot is in and can be linked to the social, technology and product service triangle.

In an empty street, the robot should go all out on product service and technology aspects and ignore social aspects. When people are involved, the robot should include social aspects and lower the importance of technology and product service aspects. What these values should be is yet to be investigated, but a good approach would be to use a machine learning method for this; evolutionary algorithms.

Evolutionary algorithms

According to Shiffman (2015) in 'The Nature of Code' one can find a 1D array of floating point values between 0 and 1 at the basis of a traditional genetic algorithm (as part of evolutionary computing). Each entry in the array can be seen as an individual trait that the simulated agent possess. The floating point values (traits) can be mapped to another value which corresponds to an agents parameters. Another way to look at it is to visualize the array as the string of DNA from which selection, crossover and mutation can be performed, see Figure 37. Within the context of the delivery robot and a Social Force Model, the traits could correspond to a desired velocity, pedestrian force strength, pedestrian range, obstacle force strength, obstacle range, base anisotropic factor value, maximum acceleration, forward simulation horizon, forward simulation range, etc.

functions with their rewards and penalties as discussed before.

The probability of an agent being selected is based on the fitness of that agent. Those who did better have a higher chance of being selected for crossover and mutation. In the case the probability is directly proportional to the fitness of an agent, the selection method is called Roulette Wheel Selection (Jebari & Madiafi, 2013). One known drawback of this method however, is the tendency to prematurely converge to a local optimum. A different way to select, which tries to counteract this tendency, is Linear Rank Selection. As the name suggest the selection is based on the rank of an agent, which is derived from the agent's fitness relative to the others. Worst scoring agent gets the first rank, best scoring agent the nth rank. In this way the probability of an agent being selected is

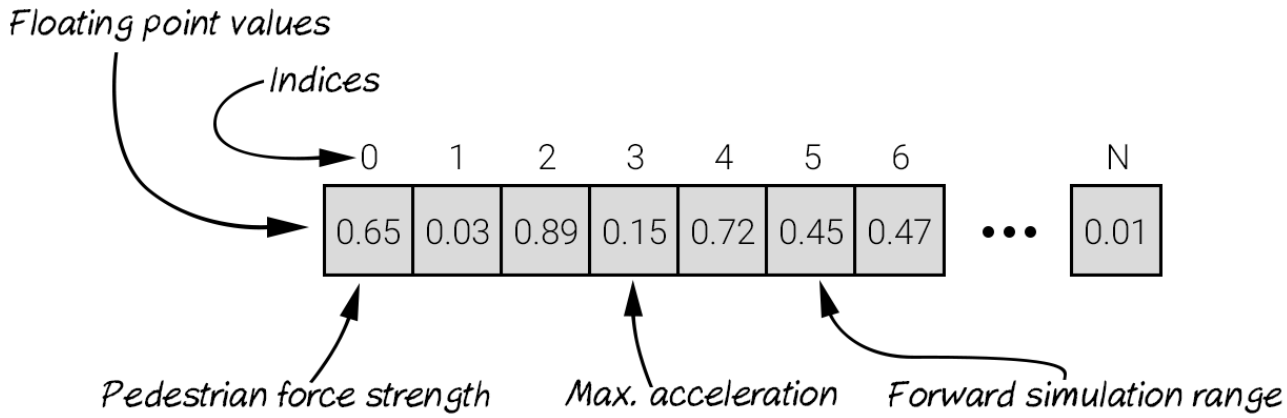


Figure 37. 1D array of floating point values from 0 to 1, representing the traits of an agent 'DNA'.

The selection process is a process in which the agents are selected according to their fitness. Fitness is defined as 'how well did the agent accomplish his task?' An example of this could be a single measure; 'how far did the agent get?' or a combination of measures such as progress, collisions and discomfort. The fitness function should be defined in such a way that it suits the objective the most. It is thus closely related to cost / objective

not directly proportional to its fitness value. Thus the likelihood of converging on a singular out performing agent, which might be outperforming due to a local minimum, is counteracted.

In some cases the fitness of the Linear Rank Selection is scaled exponentially in order for the best agents to have a higher chance of being selected to promote convergence, this is called Exponential Rank Selection.

A fourth method of selection is the Tournament Selection. In this method a random subset of agents is taken and the best agent within the subset is selected for reproduction. This is repeated n times where n is the population size. This method generally converges quicker but because of that risks having a low genetic diversity.

The above mentioned selection methods all try to balance exploration and exploitation. Those with a faster convergence are prone to exploitation; finding a (local) optimum solution quickly, but at the cost of exploring other solutions. Other methods are slower and explorer more in an attempt to find a better (global) optimum. It is found that it is often favorable to use a selection method which at first favors exploration and gradually increases convergence pressure to favor exploitation (Saini, 2017). In the context of the SFM-MPDM delivery robot, one way this might be done is to increase the exponential factor in the Exponential Rank Selection method based on the generation number relative to the maximum number of generations.

After successful selection comes crossover. Crossover can be seen as taking the two arrays of the selected parent agents and building a new array for the child agent. This is done by, for each index in the array, picking the value of the array from either the first or the second parent. In this way you end up with a new array that is a mixture of both selected 'parent' arrays (Shiffman, 2015).

After crossover comes mutation. Mutation means with a small probability alter each value of the resulting array to a either a completely new value generated between 0-1 or instead pick a new value based on the old value with a standard deviation (Eiben, Hinterding, Michalewicz, 1999). Mutation generally has a small probability (1~2%). With a mutation rate that is too high, the model does not converge. And without mutation it could be that the optimal solution is never found because it happened to not be part of the initial random weights (Shiffman, 2015).

In essence the idea of fitness, selection, crossover and breeding is not a complex whole. What makes it complex is the range of different methods and different parameter values which come with evolutionary algorithms. Eiben et al. state: "The values of these parameters greatly determine whether the algorithm will find a near-optimum solution and whether it will find such a solution efficiently." Parameter values can either be tuned or controlled. Tuning means to find good values by hand and use these fixed values during the run. Control means to start with initial parameter values which then change during the run.

Is it common practice in evolutionary computation to tune the parameters by hand, but this is known to have drawbacks. Instead it is suggested to use parameter values that change over time, e.g. the mutation rate starts high and goes lower each generation. First to explore (large mutation rate) later to fine tune (small mutation rate), which seems similar to increasing selection pressure over time, as mentioned before, see also Figure 38.
Some important insights were gathered from studying the evolutionary algorithm literature:

- There are many parameters to tune (mutation rate, population size, selection size, selection strategy, maximum number of generations, convergence pressure, etc.).
- Tuning parameter values by hand is common practice, but can be time consuming and does not guarantee good values.
- Evolutionary algorithms do not guarantee the optimal solution to a given problem, instead it gives the 'best solution it could find'. It is therefore important to first give the algorithm the chance to explore the solution space before converging.
- Mutation rate should start small and decrease even further over time while selection pressure should start low and increase over time. Both help to first promote exploration and then promote convergence.
- The fitness function which evaluates the agent's performance should be set up in such a way that it reflects the rewards and penalties as discussed in the previous chapter.
- As the floating point values in the 'DNA array' need to be mapped to actual SFM-MPDM parameters, a minimum and maximum of these parameters needs to be defined before optimization can take place.

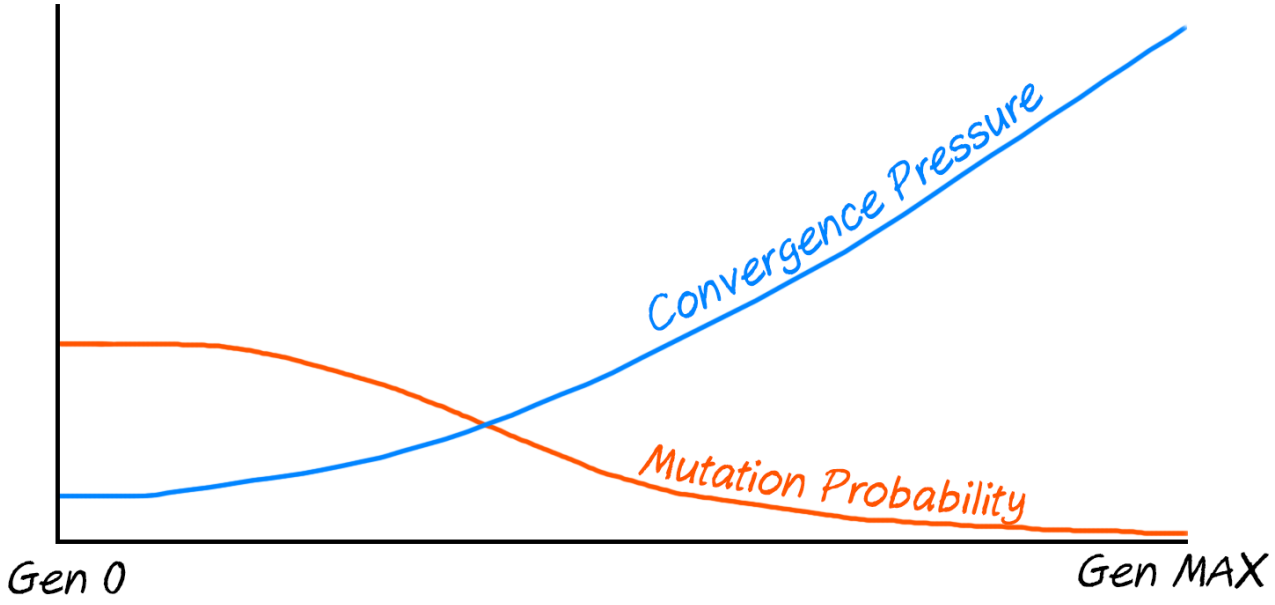


Figure 38. Change of mutation probability and covergence pressure over time.

Discussion and requirements

In this chapter the discussion points of the previous technology chapters are discussed and requirements and wishes are presented from these discussion points. The requirements and wishes can be found in chapter 'Program of Requirements and Wishes'. Besides the S, a T is now also included to indicate a Technology requirement or wish, e.g. R. T1.

In the 'Physical and dynamical parameters and their impact' chapter it became clear that the battery drains faster when energy consumption is high, with a more rapidly draining battery comes a decreased effective delivery range and a decreased operational time. Listed below are conditionals which would decrease an AGV's range and durational performance, coupled with requirements and/or wishes. AGV range and durational performance is decreased when:

- The robot is not optimized for low mass; W. T2.
- A fast driving speed is to be maintained; R. T5
- A very slow driving speed is to be maintained (due to relative high roll resistance and reduced efficiency in lower power levels); R. T5
- The robot decelerates and accelerates regularly; R. T5.
- The robot is driving up slopes; R. T4.
- The robot is steering a lot:
  - By lengthening its path more energy used; W. T4
  - In case of a differential drive; the motors have to slow and speed of the wheels each turn, here wheel mass also comes into play; W. T3
- The surface on which the robot is driving is rough (gravel, sand, potholes); R. T3 and R. T4.
- The wheels of the robot have a large amount of slippage; W. T5
- The ambient temperature is lower than the ideal battery temperature (such as ~20 degrees Celsius); R. T6
- Sensors, actuators and logical processing require a large share of the power; W. T1

The chapter 'Capabilities and limitations of the ROSbot 2.0' is first and foremost a chapter written for this project in order to gain insights into the research's robotic platform. It was found that care should be taken with the robot's small size and the effect this has on any user test results. A way to deal with this downside is yet to be found, but one could think about resizing the test scenarios to match the scale of the robot or vice versa scale the robots perceived size by building onto it to mimic the delivery robots measurements. Next to this it became clear that the robot is not suited for outdoor environments. This is due to the small wheel clearance and because of the fact that the platform is not designed to operate in weather conditions; R. T8 and R. T9. The robot's translational and rotational velocities were found to be adequate for driving among pedestrians. The onboard RGB-D and LiDAR sensors were also found to be fully capable of viewing the surroundings and detecting objects, but a point of attention is that the RGB-D camera is only looking forward and the LiDAR only 'sees' a in horizontal plane. There is no vision on the sides or behind the ROSbot. Care should also be taken when trying to run intensive programs such as SFM-MPDM on the ROSbot 2.0 Pro hardware. Aspects which influence the programs intensity are the number of observed pedestrians (which is also tied with observation range), simulation frequency and the length of the simulation horizon; R. T7 and W. T1.

In the chapter 'Machine learning with AGVs' the reward and penalty parameters are discussed alongside evolutionary algorithms. From this it became clear that the algorithm which is to control the AGV should receive feedback on what positive and negative occurrences are. The preferred method for this is setting

up a cost or objective function with rewards and penalties. Many of the identified penalties are already captured by the requirements and wishes of the aforementioned physical and dynamical parameters. As well as from the requirements and wishes of the Social chapter; R. S2, R. S4, R. S5, R. S7, R. S8, R. S9, R. S10 and W. S4.

When looking at evolutionary algorithms, it became apparent that many selection methods are available. Each tries to balance exploration and exploitation, where those with a faster convergence are prone to exploitation and those which explore more in an attempt to cover the solution space are found to be slower. It was thus discussed that convergence pressure should start low to favor exploration and gradually increase to favor exploitation.
Another method which promotes exploration, in an attempt to cover the solution space, is mutation. In general, mutation probability should be kept low (1~2%). It can, however, start out slightly higher at first (exploration) and decrease over time to then promote convergence (exploitation).

Additional insights that were gathered from studying the evolutionary algorithm literature are listed below:

- There are many core E.A. parameters to tune (mutation rate, population size, selection size, selection strategy, maximum number of generations, convergence pressure, etc.).
- Tuning parameter values by hand is common practice, but can be time consuming and does not guarantee good values.
- Evolutionary algorithms do not guarantee the optimal solution to a given problem, instead it gives the 'best solution it could find'. It is therefore important to first give the algorithm the chance to explore the solution space before converging.
- Mutation rate should start small and decrease even further over time while selection pressure should start low and increase over time. Both help to first promote exploration and then promote convergence.
- The fitness function which evaluates the agents performance should be set up in such a way that it reflects the rewards and penalties as discussed in the 'Reward and penalty parameters' chapter.
- As the floating point values in the 'DNA array' need to be mapped to actual SFM-MPDM parameters, a minimum and maximum of these parameters needs to be defined before optimization can take place.

Service as provided by the AGV

In this chapter the goals, tasks and services the delivery robot should provide are explored as found within the product-service combination. Similarities are drawn between the intensively tested Starship Technologies product-service robot and the to-be-designed AGV.

The core service the robot provides is the (autonomous) delivery of goods. For this to be possible, to robot needs to be able to reach its delivery goal. That is to say, the goal should be within the possible bounds of where the AGV is able to go. Whether or not the AGV can reach the goal should thus be mapped in advance by a global motion planner, combined with the metrics of the robots capabilities and effective range as discussed in the 'Technology' section of the research and exploration phase.

Through experimentation with the Starship Deliveries robot delivery application, it was found that if the goal cannot be reached by a Starship robot, the service proposes an alternative which can be reached (Starship Technologies, 2018b), see also Figure 39. The user is then allowed to accept or propose their own solution. This process repeats until a valid delivery location is found and agreed upon. It might seem convoluted or tedious to keep repeating the few steps, but in general a delivery location is quickly found due to the service application helping the user in the process. The use of such a system would be wise addition to any service concerning itself with the autonomous delivery of packages or payloads with robots.

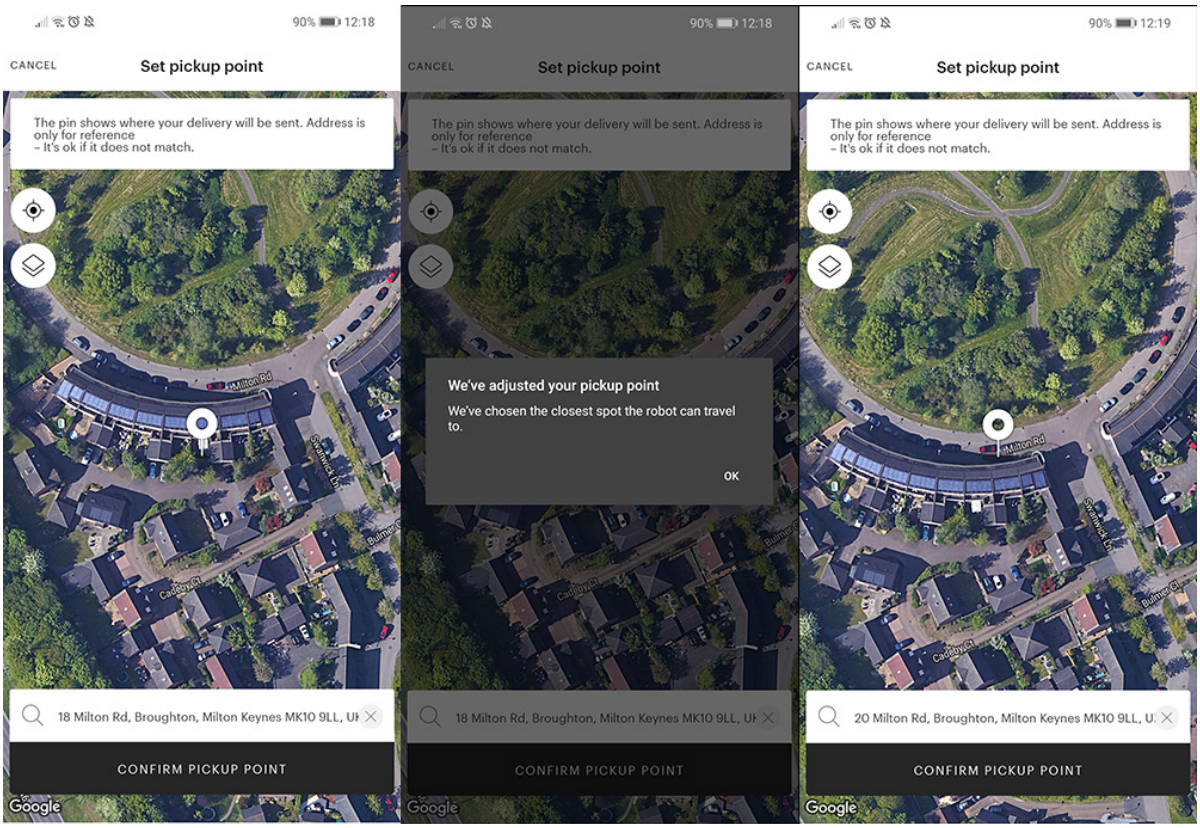


Figure 39. Starship delivery application. Left: User chosen location. Middle: Notification of pickup point adjustment. Right: Adjusted pickup point shown on the map.

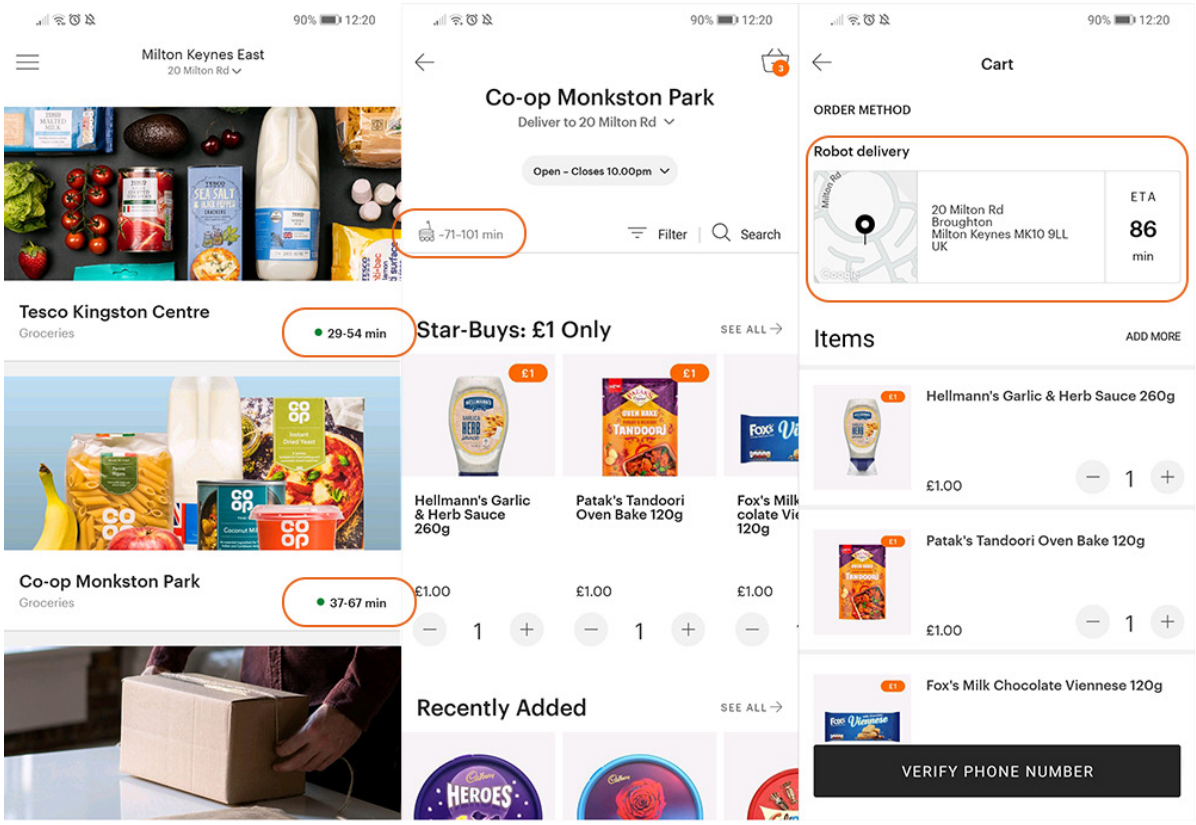


Figure 40. Starship delivery application. Left: User chosen location. Middle: Notification of pickup point adjustment. Right: Adjusted pickup point shown on the map.

When an agreement is reached upon a delivery location, the expected delivery time or time frame is communicated, see Figure 40. This, combined with real time tracking of the robots position, or real time updates of the robots estimated time of arrival, should provide the user with enough information to prepare for the reception of the payload. Upon arrival the user should be notified and should provide an option to unlock the specific compartment containing the user's payload.

The internal compartment size governs the maximum payload measurements which can still be transported by the robot. The compartment size and number of compartments is dependent upon the items which the robot is meant to deliver, e.g. a large range of medium sized parcel addressed to a range of recipients or two full bags of groceries for just a single user. It thus depends on the implementation of the robot. To give an indication, the Starship delivery robot provides service to only a single user at a time and thus utilizes only a single compartment. It delivers anything from pizzas and burgers to groceries and postal parcels. The Starship robot, for example, has an internal compartment size of 402 x 344 x 330 mm (Swiss Post, 2017). As another example, if one wishes to stack mail parcel boxes or pizza boxes on top of each other, the minimal internal width and depth should be 400 mm by 340 mm, while the height would mostly depend on the amount of boxes one wishes to stack; PostNL (2018), DHL (2019) & Premium Disposables (2019).

Along with payload dimensions comes payload weight. According to the Dutch occupational health and safety legislation for lifting and carrying objects, the maximum weight is allowed to be 23 kg during ideal working conditions (ministerie van Sociale Zaken en Werkgelegenheid, 2019). With a delivery robot such as Starship's, the vertical and horizontal distance to the ideal

position should be taken into account. With estimated values of 0.7 m vertical offset and 0.4 m horizontal offset, the maximum weight is roughly halved to 11.9 kg (Federatie Nederlandse Vakbeweging, 2011). Although this is still plenty for most applications. In the case of, for example, a large amount heavy groceries, it would be wise to split this up into several bags, see Figure 41. This should be beneficial for the end user, but also for the person who has the job of loading the payload into the delivery robot.

It can be the case that the item which is to be delivered should be kept hot or cold during transportation, e.g. a hot meal or a cold beverage. In these cases the service should provide thermal insulation to maintain payload temperatures in order for it to be within acceptable levels upon delivery, see Figure 42. This again strongly depends on the context in which the payload is to be delivered.

Figure 41. The splitting up of heavy payloads into smaller parts.

Figure 42. Themal insulation as can be found in a Starship 'hot meal' delivery robot.

A final concern is that of social sabotage. Although Starship Technologies has been testing their robotic delivery fleet since early 2018, no known cases of severe social sabotage have come up. The senior VP of Business Development at Starship even mentioned: "Many people believe that robots will be stolen but the reality is very different from this. In tens of thousands of autonomous deliveries, we've not had any robots stolen." (The Spoon, 2019). Despite this, the service should be able to deal with potential social sabotage and the robot should be able to withstand it.

One can think of situations where the delivery robot is intentionally blocked or closed in, situations where one would try to pry open the payload compartment or try to vandalize or steal the robot itself. In these cases it is important to provide protection to the payload and to take steps to prevent escalation. The payload compartment(s) should always be locked, only the service provider and the end user should be able to open the compartment which holds the end users item. A logical way to do this is to, at the delivery location, interact with the robot through the application or interface with which the robot was requested, see Figure 43. Another would be to, upon robot delivery request, provide the user with a unique and temporary (digital) key which they can input into the robot upon arrival.

When the robot is physically attacked by an assailant, the first step to take would be to try and deter the assailant. This could be done through visibly and audibly sounding an alarm on the robot itself, as well as in a control room. If this does not work, the robot should try to use its cameras to capture an image of the assailant and inform authorities of the issue. Authorities in this case would be the delivery robot company. In the case the robot is being taken, it could suggest to the delivery company to contact law enforcement. Next a second dissuasion attempt should be made towards the assailant by informing him/her of the alerted authorities. This could be done through a pre-recorded message or with an employee speaking through the robot. Finally, if all else fails, the robot should continuously send out its GPS location to keep track of where it is being taken.

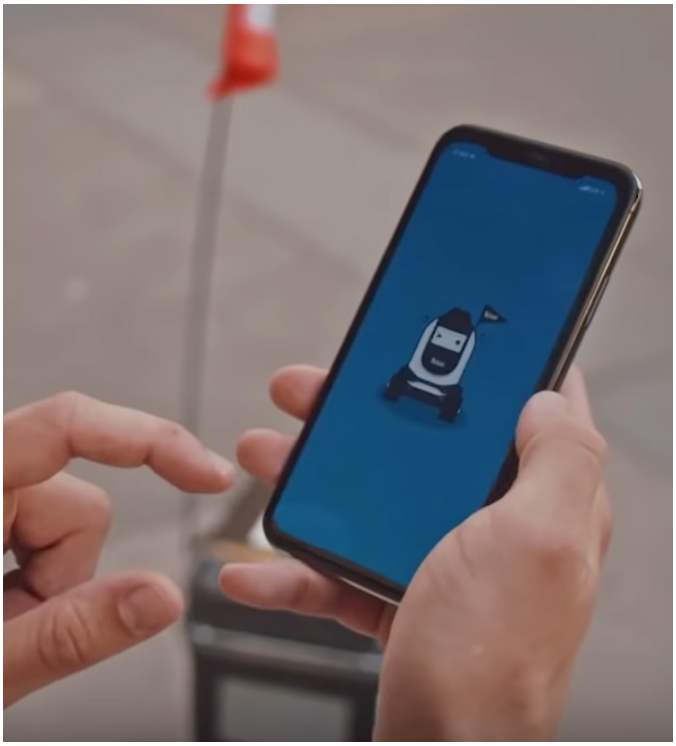


Figure 43. The user as he unlocks the delivery robot (in this case a Kiwibot) with his smartphone.

Discussion and requirements

In this chapter the discussion points of the product service aspect are discussed and requirements and wishes are presented that arose from these points. The full list of requirements and wishes can be found in chapter 'Program of Requirements and Wishes'. Alongside the S for Social and T for Technology, a P has been added to indicate Product service; e.g. R. P1.

In the product service analysis section it has become apparent that the service should allow the user the ability to choose a delivery location and should provide the user with updated information of the delivery status. First of all delivery location should be negotiated; R. P1 and R. P2. After agreement of location the user should be presented with an estimated time of arrival; R. P3. This time of arrival estimate should be regularly updated and the arrival should be communicated to allow the user to prepare for the arrival of their goods; R. P12 & R. P4. Upon arrival the user should be able to retrieve his/her goods; R. P6.

It was found that the goods themselves should be presented in such a way that both the payload loader and the end user can retrieve them without risking health and safety. If the weight goods exceed the proper limits, they should be split up in such a way that they can be separately retrieved from the compartment(s); R. P5.

The compartment size should accommodate for the dimensions of the goods which are to be delivered. For the sake of example, if one wishes to transport food (such as pizzas) or parcels a good minimum dimension would be a width of at least 340 mm and a depth of at least 400 mm; R. P7. A note here is that the height fully depends on the use case; if one user is to be served a single large compartment will suffice, possibly split up by dividers if so required. However in the case of a parcel delivery robot such as imaged by de Groot, multiple compartments are preferred to accommodate a larger amount of users, the height of the compartments should be chosen to then accommodate the heights of standardized parcel boxes. In the case where the temperature of the goods matter, the service should provide thermal insulation for the goods to be able to present them near the desired temperature upon arrival; W. P1.

Although in practice it seems to be going alright, it was discussed that the product service should be able to deal with social sabotage. The first step to this is to secure the payload in a locked compartment which only the delivery company and the end user can open; R. P8. Next, it was theorized that the robot should try to limit damage to the goods inside; W. P2. It was suggested that the robot should try to dissuade the assailant, for example by visually and audibly sounding an alarm; R. P9. The next step would be to try and capture imagery of the assailant which could prove useful in the case the aggressor needs to be identified; R. P10. It was then theorized that the robot should inform the authorities such as the delivery company and, in severe cases, indicate to the delivery company that they should inform law enforcement; R. P11. In the meantime the robot should continue to try and deter the assailant by informing him/her of the alerted authorities. And lastly, it was theorized that in the case the robot is being taken, the real time tracking should continuously update its position so authorities can locate it; R. P12.

Program of requirements and wishes

Product requirements and wishes can be derived from the discussions of the three main aspects. The requirements (R) and wishes (W) are divided up into the three aspects with which they are mostly related; social, technology or product service, this is denoted with an S, T or P. All requirements and wishes are stated such that they apply to the autonomous delivery robot as if fully implemented and operational within the pedestrian context.

Social

R. S1. The robot should adjust its behavior to the situation it finds itself in (adjust algorithm parameters based on the identified situation).

R. S2. The robot should occupy intuitive (predictable) and comfortable (ease of use) behavior.

R. S3. The robot should by default keep to the right half of the walkway; when no pedestrians or obstacles are nearby, the robot should stay within a 20% margin of the ideal line, where the ideal line is positioned at a distance of 25% of the walkway width as measured from the right boundary of the walkway.

R. S4. The robot should maintain comfortable distance to other pedestrians while driving
a. Overall minimal distance of 50 centimeters.
b. Minimal distance in front of a pedestrian of 100 centimeters.

R. S5. The robot should avoid physical contact with pedestrians and obstacles.

R. S6. The robot should behave as expected of one using social navigation, in default scenarios:
a. Overtake on the left.
b. Pass on the right.
c. Cross behind.
d. Adjust its speed based on environmental factors such as pedestrian density.

R. S7. The robot should be able to go against pedestrian traffic convention if this is 10+% more comfortable for the surrounding pedestrians.

R. S8. The robot should be able to determine the most comfortable and progressive trajectory.

R. S9. The robot should follow with a respectable distance as to not make the human uncomfortable and not give the feeling of being tailed; a minimum of 2 meters distance while following must be held.

R. S10. The robot is allowed to follow the same person for a maximum of 7 seconds to avoid the uncomfortable feeling of being followed.

R. S11. The robot should, at minimum, communicate its intent through core vehicle movements such as vehicle position, speed and acceleration.

R. S12. The robot should not cause social disruption by 'socializing' with pedestrians.

R. S13. The robot should take the path conditions into account when predicting pedestrian trajectories.

W. S1. The robot should take the effect into account that the approachability of a pedestrian and/or obstacle has on the repulsive force.

W. S2. The robot should exhibit a turning radius proportional to its translational velocity.

W. S3. The robot should ease-in and ease-out when accelerating and decelerating.

W. S4. The robot should avoid others colliding with it by timely communicating imminent changes of movement.

W. S5. The robot should adjust its trajectory at a distance, based on pedestrian density.

Technology

R. T1. The robot should be visible in low light circumstances.

R. T2. The robot should have minimal height of 80 centimeters as to not be overlooked and/or tripped over.

R. T3. The robot should be able to turn on the spot to prevent it from getting stuck.

R. T4. The robot should be able to identify local path conditions.

R. T5. The robot should attempt to find and maintain an energy-efficient driving velocity.

R. T6. The robot's batteries should be protected to allow for optimal operation within the range of 10 to 30 degrees Celsius.

R. T7. The robot should be able to adjust its forward simulation parameters in order to facilitate successful and timely predictions of pedestrian trajectories.

R. T8. The robot should have a wheel clearance of at least 70 mm.

R. T9. The robot should be able to withstand ingress of dust and water; IP53,

W. T1. The robot's hardware should be able to support the intensive calculations that come with forward simulations.

W. T2. The robot should be optimized for a low mass.

W. T3. The moment of inertia of the robot's wheels should be a low as possible.

W. T4. The robot should follow a straight path if conditions allow it.

W. T5. The robot's traction efficiency should be as high as possible.

Product service

R. P1. The service should be able to determine if a location is reachable by the delivery robot.

R. P2. The service should provide the closest viable delivery location to the user's proposed location.

R. P3. The service should communicate the estimated time of arrival with an error margin of 5 minutes.

R. P4. The service should notify the user of the successful arrival of the delivery robot.

R. P5. The maximum weight of individually packed loads should be compliant with Dutch occupational health and safety legislation.

R. P6. The robot should provide a secure compartment for the payload.

R. P7. The service should provide a means of interaction for the user to allow the user's payload compartment to be opened within 30 seconds of reaching the robot at delivery location.

R. P8. The internal compartment width and depth should at least be 340 mm by 400 mm.

R. P9. The robot should try to deter assailants.

R. P10. The robot should try to capture imagery of the assailant(s).

R. P11. The robot should inform authorities (company) of the attempt of sabotage.

R. P12. The robot should provide real time location tracking.

W. P1. In case of a hot or cold item, the service should provide thermal insulation to keep the item near the desired temperature.

W. P2. The robot should limit damage to the payload during social sabotage attempts.

Within the bounds of this project's time, not all requirements and wishes can realistically be taken into account. They are formulated in such a way to guide and aid the development of an autonomous delivery robot while thinking about the robot as fully operational within the appropriate context. The scope of this project does not extend to such a degree and instead a scope will thus be defined which will include a selection of requirements and wishes most appropriate for the available means of the project. This scope, along with a definition of the design goal of the project, can be found in the next chapter 'Scope and design goal'.

Scope and design goal

As aforementioned, not all requirements and wishes can realistically be followed up in this project. A scope needs to be defined which should serve as a clear boundary of what is part and what is not part of the remaining weeks of the project. In this chapter this scope and a design goal will be defined to serve that purpose.

The original design goal of the assignment as defined in the project brief (see Appendix 'A. Design Brief') was stated as:

"Research and design a computational mechanism for an autonomous guided vehicle based on the social, technology and (product) service needs by utilizing a machine learning method to deal with a variety of situations while expressing its (change of) intention."

This has not changed and is still the design goal of the project, but it can be further defined now that knowledge has been gathered on the topic through research and exploration.

With the midterm meeting comes the official start of the Development and Testing phase. Any development beforehand, such as the development of the Social Force Model with MPDM in Python will be introduced as part of this phase. In the development and testing phase, the behavior and intent communication methods of the autonomous robot will be prototyped, tested and iterated. The steps in this phase will be explained below, but can also be found as a visual representation in Figure 45 on the next page.

As a first step, a user study should be performed with the custom developed SFM-MPDM, see Figure 44. The goal of which is to identify the accuracy and unexpectedness of the behavior of robot as it navigates through a set of top down view scenarios. The 'robot', as represented by a circle with a colored ring, will not use any intent communication methods other than those which come with core movement.

In parallel to this, the physical robot platform must be gotten ready for its own user testing. This means getting familiar with working in ROS and with the ROSbot 2.0 Pro. The first step to this is to make a virtual ROSbot drive around based on script generated commands. Next, the custom build Python script with SFM-MPDM should be adjusted and ported to work in a ROS environment. At first pedestrians and obstacles could be

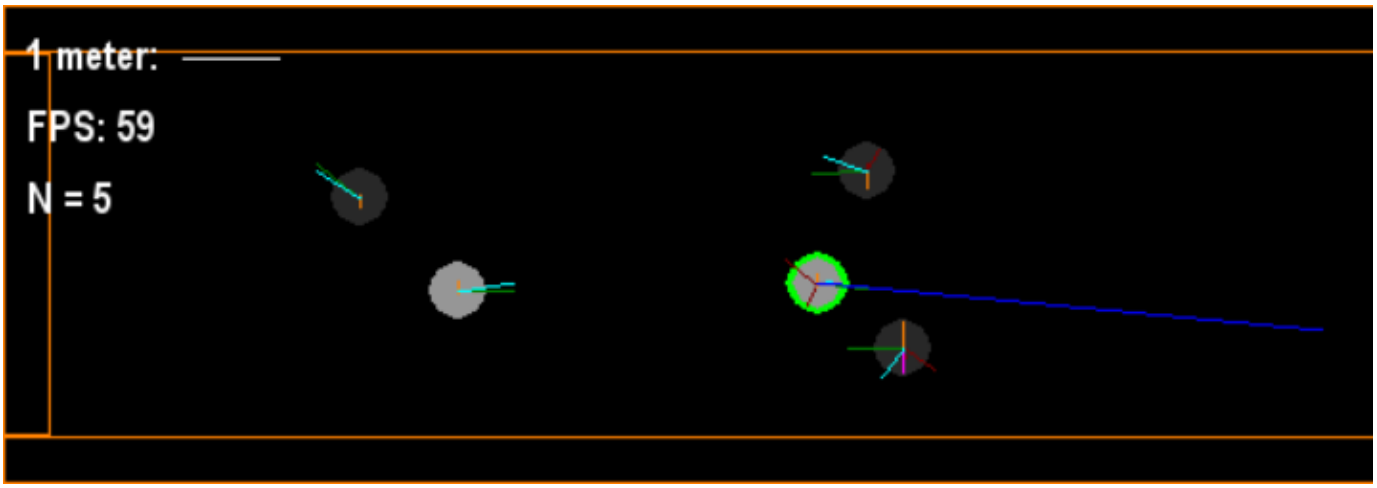


Figure 44. The Social Force Model with Multi-Policy Decision Making (SFM-MPDM) as made with Python.

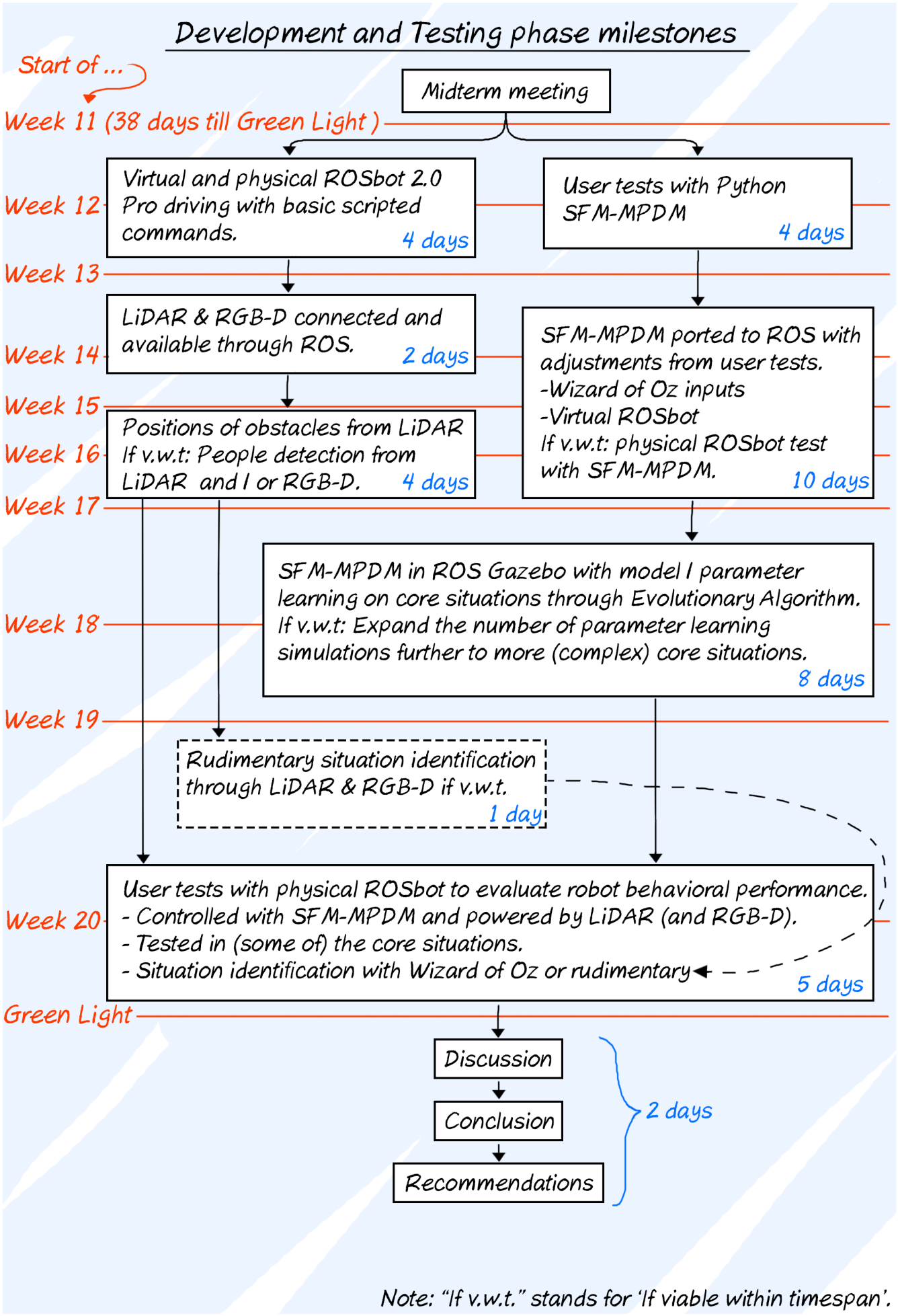


Figure 45. Development and Testing phjase milestones planning.

simulated through a 'Wizard of Oz' method, but later on they should be identified in the virtual Gazebo world through the use of LiDAR and/or the RGB-D camera. If everything is functional inside the virtual world, it should in theory all be 1:1 with the physical world and thus it can be implemented with the physical ROSbot.

After the ROSbot is functionally operational (either virtual or physical), the model can be 'trained' to adapt the robots behavior in different scenarios. The first step to this is to program an evolutionary algorithm (EA) to use as the machine learning method. The EA should find the optimal behavior of the robot for a core situation. As a start, the first three of the eight core situations should be simulated and their parameters learned. After those have been successfully simulated the 'Normal street' situation will be simulated and its parameters will be learned. After which a decision should be made if further simulating is worth the time spent or if it is wiser to move on. One of the aspects is if the set amount of time is depleted, another is if the simulation limits have been reached and a third is determining how much is gained by simulating and learning the additional of the eight core situations. See chapter 'What situations can the AGV encounter?' for a more detailed description of the eight core situations.

While training the models and finding the right parameters for the situations, the input of the obstacles and pedestrians (such as position and size) should be simulated and do not necessarily have to come from LiDAR or RGB-D.

The EA will determine behavioral fitness through a fitness function with the identified reward and penalty parameters (see also chapter 'Machine learning with AGVs). At first the solution space should be explored by the EA with low convergence pressure and relatively large mutation probability, after which convergence should be promoted with higher pressures and lower mutation probabilities.

The goal of all of this is to be able to test and evaluate the robots performance with users and obstacles in core situations. An important note here is the identification of the situations and the number of tested core situations. For the user tests the situation identification is assumed to be manually controlled using 'Wizard of Oz' techniques. If it is viable within the workload and time bounds however, to have autonomous detection and identification of the situation, then this is preferred. The same applies for the number of core situations which is to be tested. The empty street, obstacle in path, single pedestrian and normal street situations should provide an adequate range of test situations. If any additional core situations can be tested then this again is preferred, but will depend on available resources and whether the parameters for these situations could be learned with the evolutionary algorithm. The user tests will allow evaluation of the robot's social performance; intuitive (predictable) and comfortable (ease of use) behavior, part if which is the communication of intent by core vehicle movements. Next to this the result will allow for reflection with regard to the technology and product service performance.

The requirements and wishes which are included in the scope and which will thus be reflected upon are: R. S1, R. S2, R. S4, R. S5, R. S6, R. S7, R. S8, R. S11, W. S2, W. S5, R. T7, W. T1 and W. T4.

The results of the tests and evaluations will be used as inputs for a discussion, the conclusion and recommendations for any further development of the AGV.

Development and testing

In this phase the research and exploration will be put into practice through the development and testing of the AGV. First the computational mechanism is explained, after which the Social Force Model with Multi-Policy Decision Making is developed, described and tested with users. After adjustments based on the test results, the model is ported to the ROS environment and the Evolutionary Algorithm comes into view. The learned parameters are tested in another user test and the results are discussed.

Computational mechanism architecture

In this chapter the architecture of the computational mechanism will be discussed, starting with the inputs at the top and moving down the architecture through the input processing, situation and model parameter selection, the social navigational model and finally the outputs. An illustration of the architecture of the computational mechanism can be found in Figure 46.

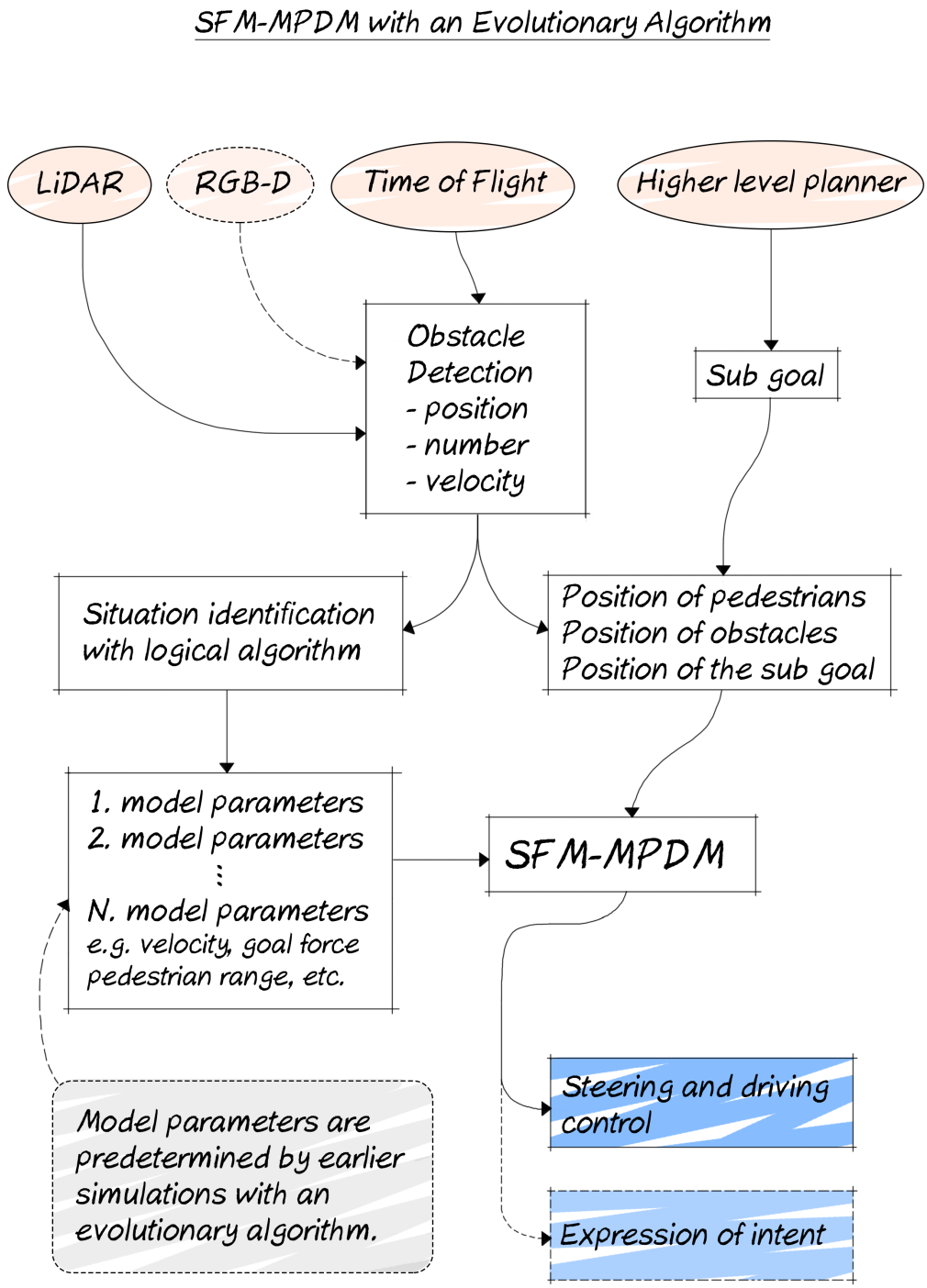


Figure 46. Architecture of the computational mechanism with SFM-MPDM.

Inputs

The computational mechanism has several inputs, most of which deal with perceiving the robot's surroundings. The first and foremost of these is the LiDAR. The LiDAR continuously performs a 360 degree planar scan of its surroundings. The environment is scanned and mapped to a point cloud format on which further processing can be performed.
The second perception method is RGB-D, a color and depth camera. Although not applied within this project, the camera is envisioned to help with identifying people and obstacles and help with the mapping of the robots surroundings. The camera's imagery can be used to run through an object classification network such as Yolo v3 or OpenCV which in turn can identify people and objects within view and match these to the LiDAR scan data.
The third perceptual input consists of four time of flight (ToF) range sensors. These four sensors are strategically placed on the front and rear of the robot and scan for objects which the LiDAR might have missed.

Next to these three perceptual inputs is a fourth input; the higher level planner. The planner's goal is to plan the robots sub goals in such a way that it can socially navigate itself to these sub goals without getting stuck. The higher level planner will take the starting point and end point of the run and maps a route with many different waypoints for the robot to follow. The high level planners direct output is thus a sub goal/waypoint which is part of the larger mapped route.

Pedestrian and object detection

Before the social navigational model can use the supplied point cloud data for calculations, it first needs to be translated into objects and pedestrians. The system of nodes takes in the point cloud data as input and outputs a list of wall segments, objects and moving objects (pedestrians). Each item in the lists gets assigned a position, size and velocity. This is enough information to output to the social navigational model. For more information on the obstacle detection see chapter 'Porting to ROS' on page 64.

Situation identification

The situation identification algorithm is envisioned to take in the data as produced by the obstacle detection setup and determines the situation the robot finds itself in. It is envisioned to does so by looking at the pedestrian density, the distance to the nearest obstacles (thus the available space) and the speed at which the pedestrians move. Based on those parameters the logical algorithm should be able to determine the situation such as 'empty street', 'obstacle in path', 'normal street situation', etc. It then outputs it's finding towards the model parameter selection script.
Note that the development of the situation identification was not included as part of the project. Interesting options are to use a classical algorithm of conditionals and effects, or to train a neural net to take in the parameters and output a likelihood score for all of the situations.

Model parameters

Based on the input as provided by the situation identification, the script selects a set of model parameters for the social navigational model to use. These model parameters are predetermined values which were learned through simulations with an evolutionary algorithm. Each set of model parameters contains the appropriate values for a specific situation, which are send to the social navigational model upon selection. See also chapter 'Learning parameters through an Evolutionary Algorithm' on page 67 for more information on the model parameters.

Social navigational model

The social navigational model is the core of the computational mechanism. It receives data from many different points and inputs these into the SFM-MPDM; the Social Force Model with Multi-Policy Decision Making. It is what makes the actual calculations based on the detected obstacles and pedestrians with the selected model parameters and the provided sub goal. With the result of the calculations it determines where to drive to and at which velocity. It continuously outputs steering and velocity control as well as expression of intent. Within the scope of this project the expression of intent will be limited to the core vehicular movement and thus is analogues to the steering and velocity control. For more information on the SFM-MPDM, see chapter 'SFM-MPDM in Python' on page 58.

Outputs

Finally there are the outputs of the computational mechanism. There are two outputs; the steering/velocity controls and the expression of intent. As aforementioned, within the scope of this project the intent will be expressed through core vehicular movement such as position, angle, rotational velocity, translational velocity and acceleration. As these are inherent to the robot itself and are controlled by the steering and velocity outputs of the social navigational model, there will be no separate output for expression of intent within this project.

The steering/velocity controls are part of the scope, as they make the robot drive around. The direct output of the models calculations is a vector with a magnitude (velocity) and angle (heading). This vector is then transformed into a translational velocity and a rotational velocity before being sent to the robots motor controller.

With this walkthrough of the architecture, a global overview of the computational mechanism is given. A more detailed description of some of the parts within the computational mechanism can be found in the upcoming chapters, starting off with the core itself; the SFM-MPDM.

SFM-MPDM in Python

In the previous chapter, the overall structure of the computational mechanism has been discussed. A vital part of this structure is the actual computational model itself; SFM-MPDM. This chapter discusses the model and the changes and additions made to it. Following this, a user test is presented with its setup and results. These are then used to evaluate the fidelity of the computational model and present possible changes to it. It closes off with the adjustments made to the model in order to finalize and prepare it for porting to the ROS environment.

Social Force Model

The Social Force Model as discussed in the chapter 'Social navigational methods' is imagined to serve as the core part of the computational model for the robot. In order to determine the viability of the SFM, it has been implemented in a Python pygame environment. This environment consists of obstacles and agents. The agents are given the tasks of moving between waypoints while avoiding obstacles and other agents by utilizing the SFM. A note here is that the implemented model is based on all of the extracted rules from the literature of Helbing and Molnár (1995) except for one; the rule of temporal distraction is excluded from the model. This rule was left out in an attempt to keep complexity at a manageable level and because it did not suit the desired robot behavior if implemented as part of the robots computational model. The SFM was found to be quite a solid foundation for basic social navigation. In some areas however, it was found to be lacking. By observing the behavior of the agents in the SFM, it became apparent that passing and crossing were not in line with pedestrian behavior as determined in chapter 'Pedestrian behavior'. Both in passing and crossing situations, agent's reactions were late and they often seemed to bump into one another. The third issue was the instant 180 degree turn, which is unrealistic for real pedestrians. For these reasons some customizations have been made to the SFM.

Customizations

During passing, no conventions were applied and this resulted in inconsistent behavior, especially when the two agent's trajectories were lined up to meet one another head on. The Dutch convention for passing is to apply the right hand rule; passing each other on the right, having the other on your left. It of course does not make sense to always force this rule if the agents are already aligned in such a way that passing on the left is more comfortable. This means that agents should pass on the side they are mostly already on and a bias should be included to slightly favor the right hand rule. Additionally, as reactions were found to be late the agents should look further ahead and identify other agents with which passing is likely. To achieve this, a long but narrow viewing cone was envisioned, where whoever was inside the cone was checked for likely passing, and if so, compensate the agent's own trajectory, see also Figure 47. To implement this in the SFM, the following pseudo code was applied:

[Passing pseude code]
for all agents in the environment:
-if not the agent itself:
--if the other agent is within the passing cone of the agent:
---if the other agent is moving towards the agent:
----set bias threshold angle to the goal angle rotated slightly counter clockwise.
----if other is left of the bias:
-----add a force perpendicular to the goal angle to the right.
----if other is right of the bias:
-----add a force perpendicular to the goal angle to the left.

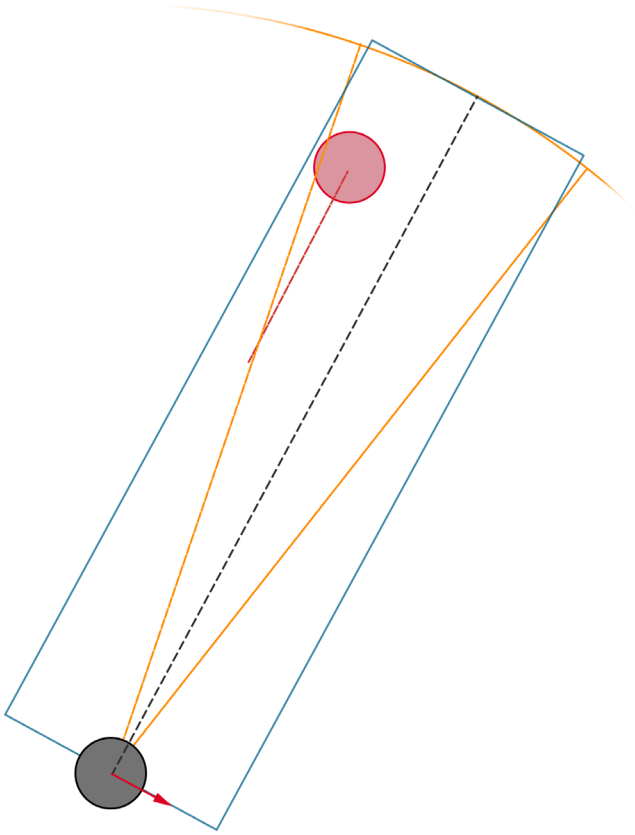


Figure 47. Passing cone in orange (and later passing rectangle in blue).

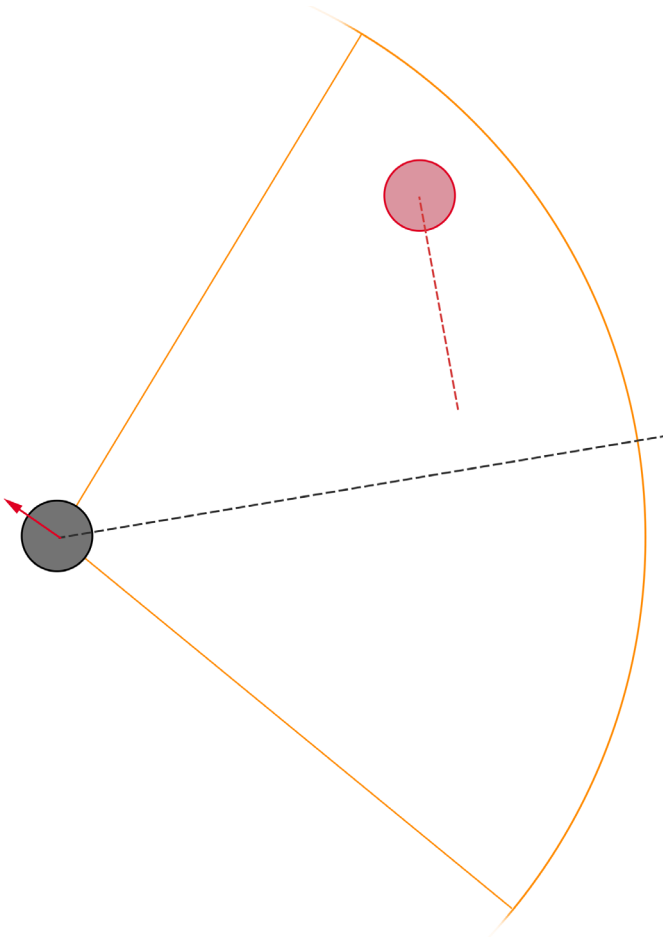


Figure 48. Crossing cone.

The passing forces were added to a list, and in the end summed up into one resultant force alongside all other forces.

During crossing, it was found that reactions were very late, and agents would often bump into one another or even block each other for several seconds.
As discussed in the pedestrian behavior chapter, there is no right hand rule convention for crossing. Instead, pedestrians try to maintain their trajectory and speed unless this will bring them uncomfortably close. They will then alter their heading and/or speed slightly to cross comfortably. The one who alters the most is usually also the one crossing behind. The decision of how to cross depends on the relative positions, timing and crossing point. In real life dominance and empathy also play a role in the evaluation, but this does not apply to identical agents. To implement crossing in the SFM, the decision will be based on the relative positions and headings of the two involved agents. To achieve the desired crossing behavior a viewing cone similar to that of passing was implemented, however this time it was shorter ranged but with a much wider angle. Any agent within this viewing cone was checked for their heading and position and if it was found that crossing was likely the agent would compensate for this in its trajectory, see also Figure 48. The pseudo code for this is as follows:

[Crossing pseudo code]
for all agents in the environment:
-if not the agent itself:
--if the other agent is within the crossing cone of the agent:
---if the other agent's heading is within crossing margins:
----if the other agent is on the right and moving left:
-----add a slowing force and a force to the right.
----if the other agent is on the left and moving right:
-----add a slowing force and a force to the left.

Again the calculated forces were added to a list, and in the end summed up into one resultant force alongside all other forces. For the actual passing and crossing Python code, see Appendix 'E. SFM-MPDM Passing and Crossing in Python'.

A third addition was the prevention of an instant 180 degree turn by introducing a rotational velocity to the agents. Before this addition, if the forces on the agent would rapidly flip, the agent would do so as well. If this would be translated to the real world, a pedestrian would either be walking backwards for a long duration or instantly turn, both of these are unrealistic and thus a rotational velocity was added which interpolates the current agent heading towards the desired heading as resulting from the resultant force.

With the addition of these customizations, the agents now exhibit behavior which is more in line with the pedestrian behavior as found in the chapter 'Pedestrian behavior'.

Multi-Policy Decision Making

As mentioned in the 'Social navigational methods' chapter, the addition of looking ahead in time could be very valuable. By trying to predict pedestrian behavior and future interactions, a more intuitive and comfortable pedestrian-robot interaction might be achieved. For this reason, Multi-Policy Decision Making (MPDM) has been added on top of the SFM for the robot to utilize. All agents will use SFM for their social navigation, but the robot will also use MPDM to find the best policy for the current situation. The model has been implemented as defined by Mehta, Ferrer & Olson (2016) with one slight change. When selecting possible leaders for the follow policy, all agents which headings do not, within a margin, match with the robot's goal heading are ignored. The underlying reason for this is to filter the amount of nearby agents in an attempt of lowering the number of calculations.

The addition of the Multi-Policy Decision Making part to the customized Social Force Model base, makes the computational model ready for evaluation with users through user testing.

User test

The model is supposed to simulate pedestrian agents navigating amongst each other in an environment representing a small square with connected alleys. A robot agent is also present in the environment and tries to move through this pedestrian rich environment with socially acceptable behavior, see Figure 49.
The goal of the user test is to evaluate the model on any shortcomings in pedestrian behavior and to evaluate the behavior of the robot by ranking the shown behavior on intuitiveness, comfortableness and predictability, as indicated by the participants. This goal is reflected in the following research questions:

1. *What shortcomings are present in the pedestrian simulation model?*
2. *How human like is the behavior of the robot?*
3. *How comfortable is the behavior of the robot?*
4. *How predictable is the behavior of the robot?*

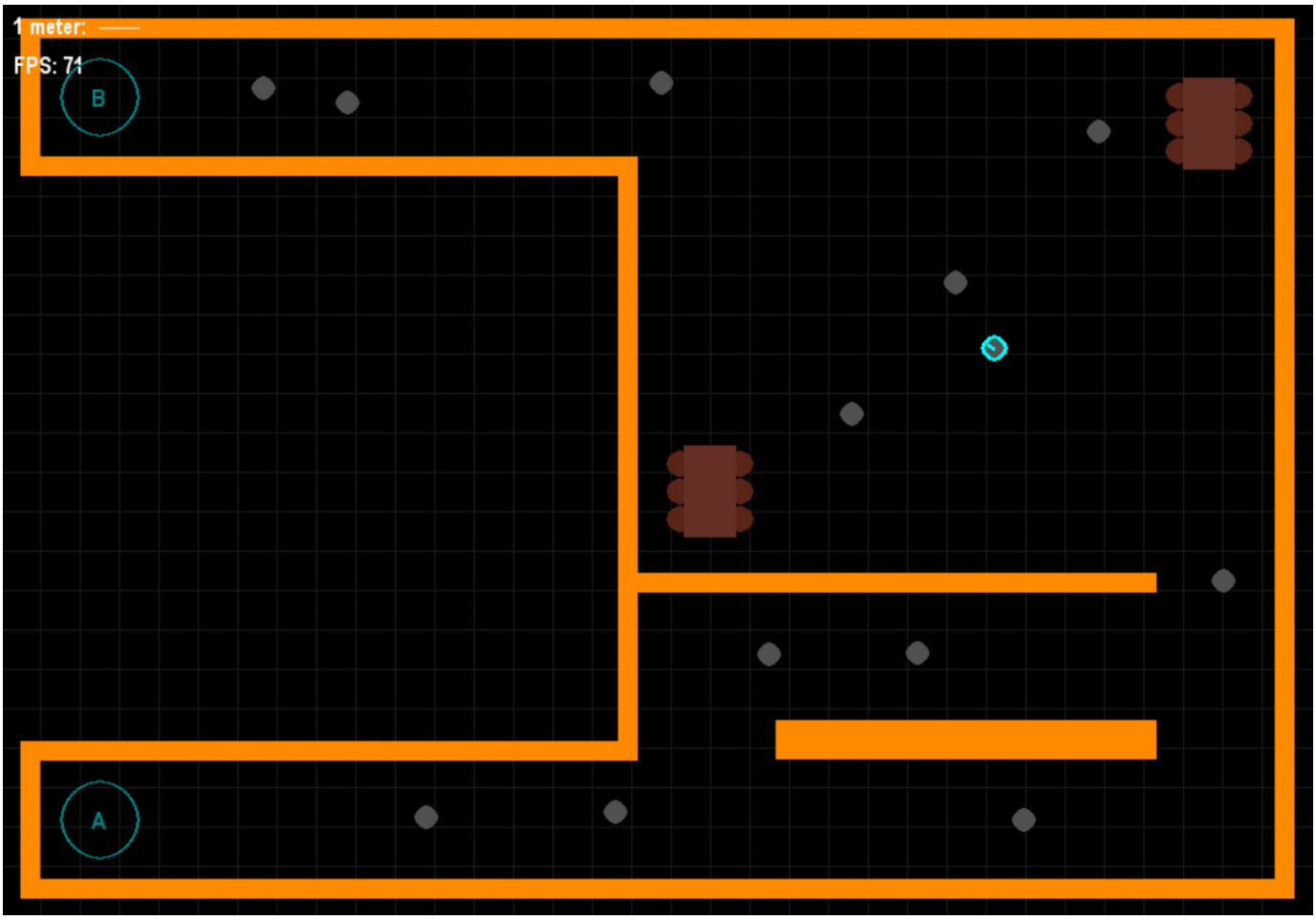


Figure 49. User test; pedestrian rich (Python) environment with hallways, a small square and some tables with chairs.

Setup

In order to evaluate the fidelity of the simulation in both the human behavior and the robot behavior, a user test was performed with participants (n=13). The participants were shown two scenarios. First scenario A was shown with just people walking around in an environment, after which scenario B was shown in which the robot was added among the pedestrians. The participants were asked to think out loud while watching the simulation and to then answer the questions in the form.

The insights resulting from the user test are to be used as an input for adjustments to the model before transfer to the ROS development environment, which in turn is to be implemented in a physical robot that can be tested in real world environments. The full user test setup, form and results can be found in Appendix 'F. User test setup of Python SFM-MPDM', Appendix 'G. User test form of Python SFM-MPDM' and Appendix 'H. User test results of Python SFM-MPDM'

Results

The results of the forms and the notes made during the user tests are used to answer the research questions, starting with the first.

In general the model was found to be quite representative of the way real people would behave (n=9), although the abstract representation does make it more difficult for some to image the circles as people (n=3).

The most commonly mentioned note was the late 'anticipation' of the pedestrians (n=5). Another interesting note is that the simulated people never stop to 'talk' or 'think', which some participants mentioned would increase realism (n=4). The 'random attraction force' was also part of the original Social Force Model as defined by Helbing and Molnár (1995) but was something which was chosen to be left out of the adjusted SFM in an attempt to keep the complexity of the base social navigational model at a manageable level. Other notes which were mentioned by more than one participant are:

- "People can walk in groups, which they don't in this case."
- "People went through walls."
- "People would cut corners."
- "People are quite random, which is realistic."
- "People don't collide, which is good."
- "People could stay to the right a bit more."

The three other research questions concern themselves with the robot behavior in terms of human likeness, comfortableness and predictability.

The robot was found to be quite human like (n=9), comfortable (n=8) and predictable (n=9), but improvements can be made. The most common improvement participants suggested is to increase the distance between the robot and the humans (n=6) and to make the robot slow down or stop to let others have priority (n=5). Another improvement mentioned is to alter the following behavior (n=4); the robot is seen as following too closely and/or for too long. One participant added to this by mentioning that the robot should, after a while, either slow down to increase distance or overtake the other in order to avoid the uncomfortable feeling of being followed.

Other aspects mentioned more than once mostly support the current behavior of the robot:

- "Is it good that the robot sometimes stops to let people pass or cross (especially near the right most corner after the open space)."
- "The robot follows straight lines when it can, this makes it more predictable; it doesn't turn suddenly or randomly."
- "The speed is good compared to the other people (similar speed)."
- "The robot doesn't necessarily show human behavior, but this can be good as humans can sometimes be egoistic or uncomfortable. Less human also makes it more predictable."
- "The robot (and people) stay quite far from the walls."

Discussion

The above mentioned notes should be taken as a main guideline for the improvements and adjustments of the SFM-MPDM model. The two main adjustments are:

- Increase the anticipatory behavior of the artificial humans.
- Increase the minimal distance the robot keeps towards artificial humans.

The issue of insufficient anticipatory behavior is most likely caused by the (late) detection of the other agents. This in turn is caused by a: the relatively short ranges of the passing and crossing cones and b: in the case of the passing cone, people are anticipating and moving aside, but the closer they get, the less side distance they keep due to the shape of the cone.

Increasing the anticipatory behavior can be achieved by altering the passing and crossing algorithms by for example increasing the maximum range or adapting from a viewing cone to an 'effective area', see also Figure 47.

The minimal distance is more complicated as there is no single measure to tune. The distance kept depends on the way the robot calculates the cost of social forces applied to others and the progress made, which need to be tuned together in a balanced way. Another option would be to add a sharply increasing cost factor for entering another's 'personal zone'. This should encourage distance keeping and decrease the number of collisions. Keeping a bigger distance in theory should also help with the perceived 'politeness' and 'giving others priority'. It could also help with the discomfort people imagine experiencing when the robot is following a person too closely. It is yet to be seen how an increase in follow distance affects the 'comfortably allowed' follow duration; but logic would dictate that an increased follow duration is to be expected.

Other possible adjustments to further improve the model can be made, such as making people randomly slow down or stop to improve the realism of the simulated pedestrians. Another option is to re-introduce pedestrians with corner cutting behavior by decreasing the amount of guiding waypoints in the scenario. A third option is decreasing wall force range (but increasing strength on short range) to not force the pedestrians and robot to the center of a hallway or alley. And a final adjustment could be to slow people down if they are about to walk 'through a wall' and let them turn on the spot if they are slowed down.

Conclusion

Several changes have been made to the SFM-MPDM based on user test results. The first of which is the selection of 'passing' agents. The viewing cone for passing has been changed into a viewing rectangular area which rotates with the pedestrian's 'angle to goal'. This change promotes a more consistent distance keeping when agents are nearing one another. Due to the narrow shape of the cone close to the agent, other nearby agents could have been incorrectly excluded from the 'passing agents' list. The rectangular area solves this issue, see Figure 47 on page 59.

A second change is the implementation of a minimum following distance to the leader. This measure has been added to the robot agent and only activates when using the follow policy. It should help with decreasing the possible discomfort people experience when being followed.

A third change is the adjustment of the SFM obstacle parameters. The range of obstacles has been decreased, but obstacle strength has been increased. This should help in hallways (width of ~2-3m) to not force the agents too much to the center of the hallway and thus makes wall distance keeping more realistic.

The fourth change that was made applies to the crossing algorithm. The crossing cone range has been increased by 50% to promote looking ahead more. This should help in creating a more realistic pedestrian crossing behavior as it was found to be 'short sighted' by the participants.

As a fifth change the walls have been made more "solid" to prevent agents from walking through them. This has been achieved by slowing down the agent when they are about to go through the walls.

A final, and only visual, addition were lines which show the heading of the pedestrians for better comprehension of pedestrian movements. With these adjustments, the SFM-MPDM model is ready to be ported the ROS environment, see chapter 'Porting to ROS'.

Porting to ROS

As mentioned in the previous chapter, after the adjustments based on the user test results, the SFM-MPDM model is ready to be ported to the ROS environment. In the Python version, everything was set up to be contained in a single python script and the world was perfectly known. This changes when moving to ROS. The world is now based on what the robot can perceive, which brings with it some necessary adjustments and additions.

Additions and adjustments

One of the most important additions to the system is the addition of obstacle detection. The robot perceives its world with a planar LiDAR, which produces a 2D laser scan point cloud. From these dots, obstacles and pedestrians should be detected. The first step in this is the addition of a ROS based obstacle detector package by Przybyla, M. (2018). The obstacle detection node structure takes in the 2D laser scan and transforms it into wall segments and circular obstacles, see also Figure 50 and Figure 51.

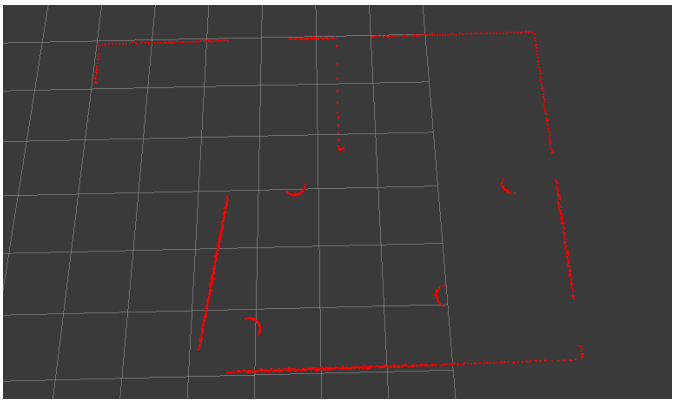


Figure 50. Laserscan data in RViz; input for the obstacle detector.

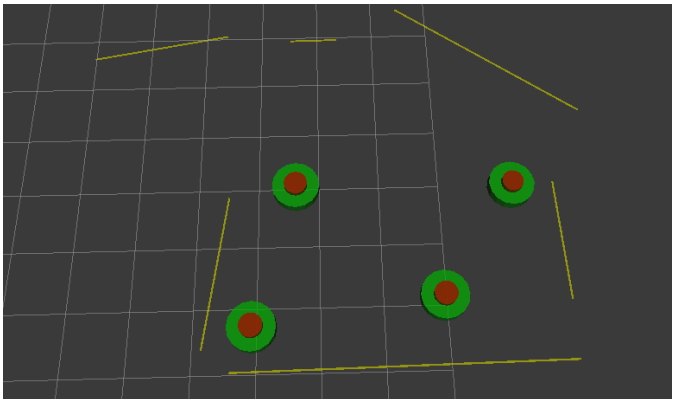


Figure 51. Obstacles and wall segments in RViz; outputs from the obstacle detector.

The obstacle detector, however, does not detect any pedestrians yet. This is where the custom written obstacle filter comes in. It takes in the obstacle data as published by the obstacle detector package and unpacks the data into wall segments and obstacles. The walls and obstacles are first filtered on a range of 5 meters. Anything outside this range is not needed for the SFM-MPDM to deal with. The range filter is followed up by an obstacle filter; the obstacles are differentiated based on their velocity and if their velocity is above a threshold of 0.1 m/s, they are marked as moving objects (agents). It visualizes this using pygame (see Figure 52) and then packs the three separate list types together which it publishes to the /sfmmpdm_obstacles topic with the custom SFMObstacles.msg format for the SFM-MPDM node to subscribe to. The SFM-MPDM node can now perceive its environment is a way of wall segments, static obstacles and moving obstacles, the last of which it treats as pedestrians (agents), see Figure 53.

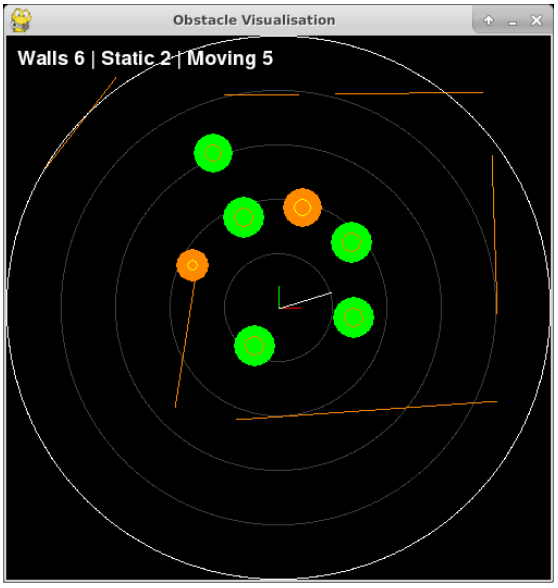


Figure 52. Obstacle filter with moving agents.

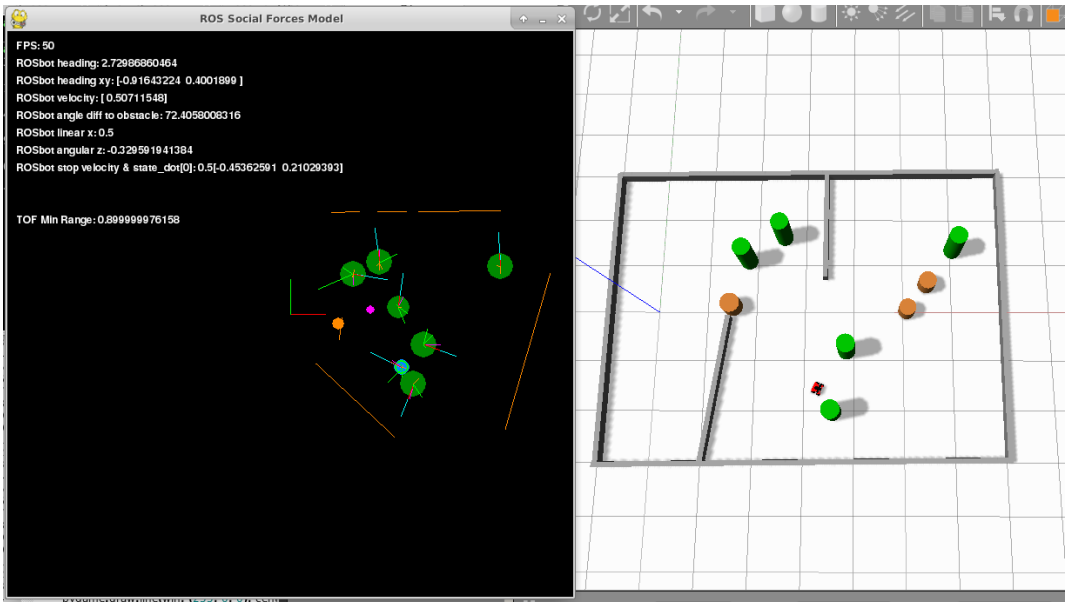


Figure 53. ROSbot perceived environment (left) and actual virual gazebo environment, (right).

Next to these additions, the SFM-MPDM code also required adjustments to deal with the change from the earlier circular 'robot' to a physics (simulated) ROSbot 2.0 and the new physics environment. This meant a recalibration of the parameter values such as the SFM obstacle range and strength, pedestrian range and strength, angular velocity, etc. And the inclusion of the front and rear Time of Flight distance sensors to slow and/or stop the robot when it's near walls or objects not detected by the LiDAR in order to prevent collisions.
The SFM-MPDM node could also no longer directly control the robot's x and y position. Instead, it needs to transform the results of the SFM-MPDM calculations into a message, after which is publishes this message on the /cmd_vel topic the ROSbot is subscribed to.

In the case of the simulated environment in Gazebo, pedestrians should also be present in the environment for the LiDAR to detect. This means the addition of a node dedicated to controlling pedestrian 'pawns' in the Gazebo world according to SFM rules. These pawns traverse the virtual world according to a waypoint system, where the Social Force Model applies to each individual pedestrian for the locomotion between each waypoint, see also Figure 54.

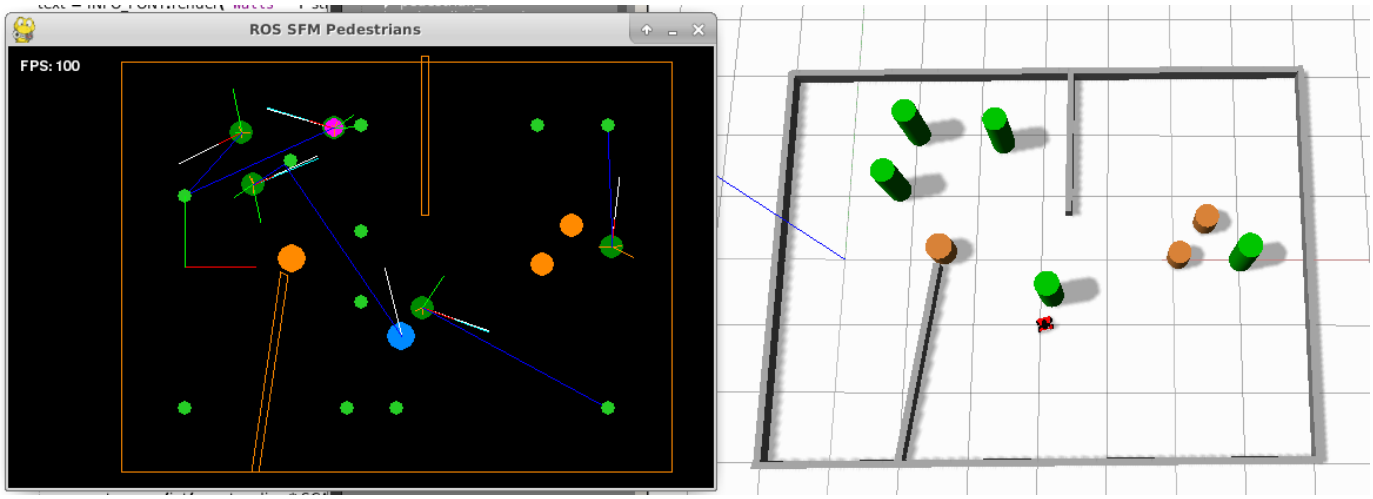


Figure 54. SFM Pedestrian controller (left) and gazebo environment with pedestrian pawns right).

Disparity issues between virtual and physical worlds

Switching from the virtual environment to the physical environment brings in some additional challenges. The real world isn't as 'clean' as the virtual world. Many smaller objects can be found which increase the noise of the perceived world. An object's height doesn't always rise above the plane which the LiDAR scans, which means it's undetected. Furthermore, the odometry (localization based on wheel displacement and inertia measurements) isn't perfect like it is in the virtual world. This results in a positional and angular drift. Lastly, due to the movement of the robot itself, static obstacles can sometimes be marked as moving obstacles, meaning they are treated as pedestrians.

Some of these issues can be solved. The 'cleanliness' of the virtual world can be reproduced by building a (user) test environment which is closed in and clear of clutter and low objects. The odometry drift issue in theory can be addressed with the use of SLAM (Simultaneous localization and mapping), but within the scope of this project will not be too big an issue as between each test run the odometry can be reset and drift won't be too noticeable between runs.

The last issue is more difficult to address. As the robot moves (due to translation, but especially rotation), static obstacles can be perceived as dynamic. One way to address the issue would be to take the robot's own movements into account and compensate for this movement in the differentiation between static and moving obstacles. For example by dynamically setting the threshold for this differentiation.

Nevertheless, a test with the physical ROSbot and basic (unlearned) parameters in a small closed of environment showed promise for the ROSbot's future social navigational abilities, see Figure 55. Furthermore, the virtual SFM-MPDM system is functional and without the above mentioned issues. It can thus be used for the parameter learning with the evolutionary algorithm, see chapter 'Learning parameters through an Evolutionary Algorithm'.

Figure 55. ROSbot 2.0 Pro in the real world with basic parameters applying SFM-MPDM.

Learning parameters through an Evolutionary Algorithm

In the previous phase in the chapter 'Evolutionary algorithms', an introduction was given to the concept of using an evolutionary algorithm for learning the robot's behavioral parameters. In the following chapters the method, limitations and the implementation of the evolutionary algorithm with its reward, penalties and fitness function, selection method, mutation and crossover are discussed. These are followed up by the results of the simulations and a discussion.

Method

The idea behind an evolutionary algorithm is to emulate genetics and Darwinian evolution in order to find a solution to a given problem. It does so by first creating the initial generation of individuals (chromosomes) where each chromosome consists of a list of parameter values. Next it runs simulations in which the chromosomes are employed in an environment. When all chromosomes of the first generation have completed their run, their fitness is evaluated using a fitness function. Next it selects individuals from the population, taking the fitness into account, and it builds new children from these individuals through parameter crossover and parameter mutation. Once a new generation is populated with chromosomes, the loop starts anew with simulating them in the environment. This continues until the maximum generation number has been met and at this point, in theory, the individuals should have gained parameters which are suited to the simulated environment, see also Figure 56.

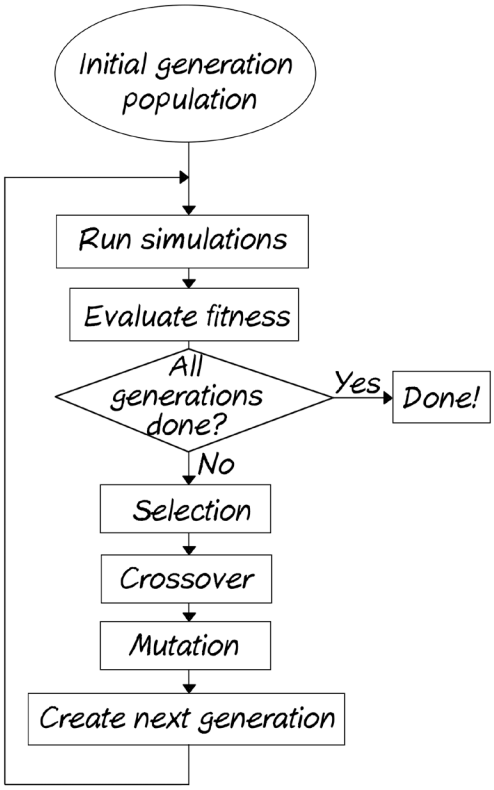


Figure 56. Evolutionary Algorithm loop.

As mentioned above, the evolutionary algorithm goes through a number of generations. Each generation consists of a number of chromosomes equal to the population size, see also Figure 57. In this case the population size is 60. Every chromosome in a generation is build up out of parameters such as maximum velocity, goal force strength, forward simulation horizon, crossing cone length, etc. There is a total of 22 such parameters in each chromosome, see also Appendix 'I. Evolutionary Algorithm parameters and their range and impact'.

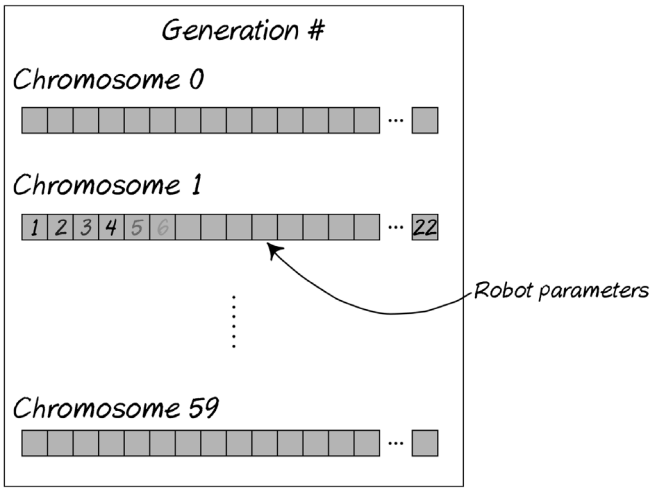


Figure 57. Generation with chromosomes and parameters.

Each parameter is part of the robots methods which together form the computational model of the robot. Thus altering a parameters value also alters the robots behavior and/or performance in the virtual physics environment.

The physics environment consists of a rectangular arena of 8 by 6 meters, see Figure 58. Several obstacles and walls are positioned in this arena in such a way that the robot will have to navigate around them. The walls also create a choke point through which the robot must navigate, which can be especially challenging when many pedestrians are also trying to get through this area. The environment can simulate 5 SFM pedestrians simultaneously, these are the tall green cylinders in Figure 58. These pedestrians patrol between their start and end points as indicated by the letters A through E and 'A through 'E. The robot, which is a virtual replica of the physical ROSbot 2.0, starts on the left and finishes its run when it reaches the red circled area on the right or when the maximum run time of 55 seconds has been reached.

In order to run the evolutionary algorithm and the required simulations in ROS, a structure of existing and additional nodes had to be set up. The existing nodes consist of the virtual physics environment with the ROSbot 2.0 in Gazebo and the earlier introduced obstacle detection node structure, see chapter 'Porting to ROS'. Next to these, three new nodes have been created, all dealing with a part of the evolutionary algorithm. There are two executing nodes and one controller node which houses the evolutionary algorithm itself. A graph of the node structure and their connections can be found in Appendix 'K. RQT Graphs of the ROS node structures'.

The simplest node of the three is the 'sfm_pedestrians_ea' node. The sole purpose of this node is to control the pedestrians (green cylinders) in the virtual environment. To be able to do this, the node needs to read from a csv

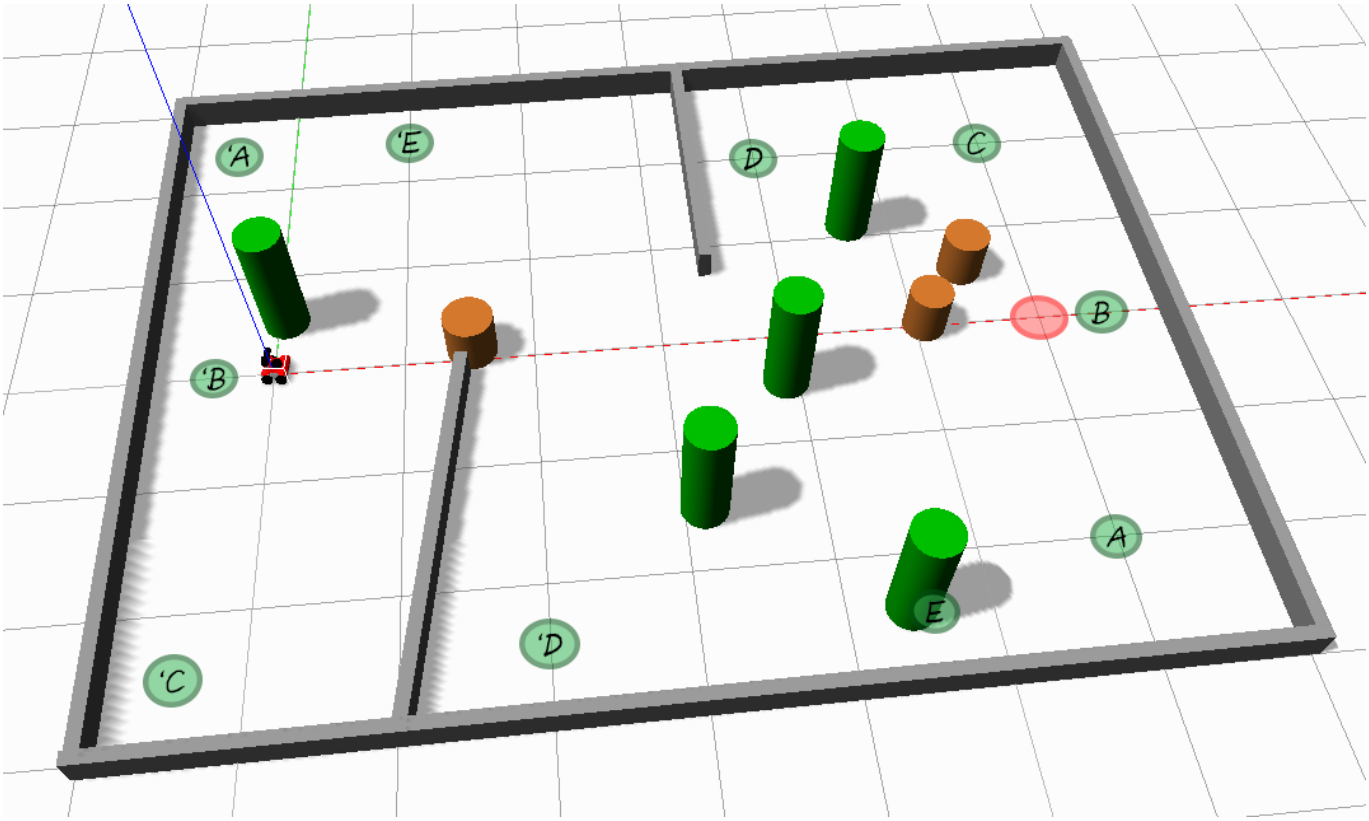


Figure 58. The arena in the virtual Gazebo world with circled indicators for the robot's and the pedestrian's start and end points.

file which contains the (shuffled) coordinates of each start position, the waypoints and end position for each pedestrian. It also requires a copy of the virtual world and the current position and angle of the virtual ROSbot. From these inputs it can then simulate the pedestrians and publish their positions to the Gazebo node.

The second node is the 'sfm_mpdm_ea' node. This node directly controls the virtual ROSbot's linear and angular velocities. Upon the start of each run, it reads out the parameter values of the current chromosome from a csv file and loads these into the robot object to use with SFM-MPDM. During the run it keeps track of the scores and saves these scores at the end of the run in the generation's csv file.

The third and most important node is the 'ea_controller' node. This node houses the evolutionary algorithm and performs most of the E.A. related work:

- It creates the initial population with their parameters and stores this into a csv file for each generation.
- It generates the waypoints for the SFM pedestrians.
- It tracks the run time for each chromosome.
- It controls whether or not the other E.A. nodes are active or standby.
- It reads out the chromosome scores and evaluates their fitness.
- It adjusts mutation probability and convergence pressure over time.
- It performs selection, crossover and mutation.
- It populates the next generation.
- And finally, it keeps track of the best chromosomes per generation and saves these in another csv file.

Limitations

Unfortunately there are also limitations to using an evolutionary algorithm for learning parameters. Some of the most important limitations come from time, parameter tuning and the lack of guarantee of finding the optimal solution. Due to the sequential setup, the time required to learn the parameter set for a single situations is equal to the time per run times the population size times the generation amount. In the case of roughly a minute per chromosome run, a population size of 60 chromosomes per generation and 30 generations, this equals 30 hours for just one situation, assuming the right behavior is learned at the first 30-hour simulation. This is a lot of time for learning one set of parameters. With this comes another time consuming task, the tuning of the parameters used by the E.A. itself, such as mutation probability, convergence pressure, population size, generation amount and fitness function weights.

Secondly, there is the lack of a guarantee of finding the optimal solution for a given problem. As mentioned in the previous phase 'Research and Exploration', an evolutionary algorithm can only give the 'best solution it could find'. This is where exploration versus exploitation is key. If the algorithm doesn't explore first, it might converge too quickly to a local optimum and thus miss the overall optimum. To make sure the algorithm first explorers and then exploits, a variable selection method probability and mutation probability were defined. See also paragraph 'Selection method' and 'Crossover and mutation'.

Finally, the functions as used in the E.A. such as fitness, selection, crossover and mutation on their own can also introduce unintended limitations by the way they are implemented. It is important to be aware of this and must be kept in the back of the mind while defining them.

Reward, penalty and the fitness function

In order for a fitness function to be able to evaluate the performance of an individual, it requires scores as in input. The scores are divided up in penalties and rewards, where each score belongs to one or more aspect(s) from the Social, Technology and Service triangle. They are defined as such:

Penalties:

- Social forces (Social aspect): *The sum of the forces in the Social Force Model which the robot exerts onto the pedestrians for the duration of the run. In the calculation of the social force score, the sum of all pedestrian forces is first multiplied with a Sigmoid function; at larger distances the social force shouldn't be taken into account too strictly, while at closer distances it should take stronger effect. The result of this is then added to the running total which at the end of the run is the social forces score.*

- Number of collisions (Social aspect): *While the robot is in collision with a pedestrian, this counter will increment, thus longer collisions while have a greater effect. A collision is defined as 'when the distance between the robots center and the pedestrians center is smaller than the radii of both combined'.*

- Runtime (Service aspect): *The amount of time the run takes, measured in seconds. Maximum runtime is defined as 55 seconds. A run can be completed before this maximum time if the robot reaches the end goal.*

- Distance to goal (Service aspect): *The distance in meters to the end goal, taken from the robots center in a direct line to the end goal center.*

- Path length (Service and Technology aspect): *The total length of the path the robot has driven during the simulation run, as measured in meters.*

- Number of stops (Technology aspect): *The number of stops is incremented with 1 each time the robot slows down to below 0.1 m/s and then speeds up again to above 0.2 m/s. This score should reflect the negative consequences from accelerating and decelerating repeatedly.*

Reward:

- Goal reached (Service aspect): *A boolean, either 0 if the robot did not reach the goal, or 1 if it did reach the end goal.*

The fitness function is a delicate function to define. The function takes in the scores of each chromosome, multiplies them by certain weights and sums them up to a total fitness value. The delicate part of this function comes from the weights and factors. Different weights will result in a different kind of fitness evaluation, which in turn means evaluating for a different problem. It thus is vital that the scores and weights together define a fitness which is evaluating the situational problem. In a situation where the social aspect of the STS-triangle is of greater importance than (product) service or technology, the setup of the fitness function should reflect this.

In order to have a baseline from which the neutral weights can be determined, 20 runs were performed with the parameter values as found in the chapter 'Porting to ROS'.

The scores of these runs were taken, averaged and rounded, see also Appendix 'J. Baseline for the fitness function weights'.

In the neutral case where each aspect is equally important, the baseline scores for each aspect should also have an equal impact. This results in the following weights, see Figure 59.

The fitness function is built up from the three aspects and their respective values:

Score	Weight
Social Forces	1.515
Collision num	14.706
Runtime	6.250
Path length	48.077
Distance to goal	83.333
Goal reached	250.000
Stops count	35.714

Figure 59. Baseline weights for the fitness function.

$$Fitness_{Social} = Importance_{Social}(-1.515\ SocialForces - 14.706\ CollisionCount)$$

$$Fitness_{Service} = Importance_{Service}(250\ GoalReached - 6.25\ Runtime - 83.333\ DistanceToGoal\ -\ 0.5 * 48.077\ PathLength)$$

$$Fitness_{Technology} = Importance_{Technology}(-35.714\ StopsCount\ -\ 0.5 * 48.077\ PathLength)$$

The overall fitness is then defined as the sum of the separate fitnesses. In the fitness formula for each aspect, one can find the 'importance' factor. The values of the three aspect importance factors are directly related to the Social, Technology and Service triangle. The sum of the three importance factors should always be 1. The position of the marker within the triangle determines the value of each importance factor.
Considering an equilateral triangle with a height of 1, the value of an aspect's factor is equal to the distance from the marked point to the opposite side of that aspect's angle in the triangle. Having the marked point in the center of the triangle results in equal importance to all aspects; each aspects is then ⅓, Figure 60 (left). Moving the marked point towards, for example, the social aspect results in shifting values for the importance factor, Figure 60 (right). As mentioned above, the definition of the fitness function is a delicate procedure, and the position of the marker within the STS-triangle is part of this procedure. Before a simulation for a specific situation is started, the marked point's position should be considered carefully, thinking about the situation at hand and the desired robot behavior.

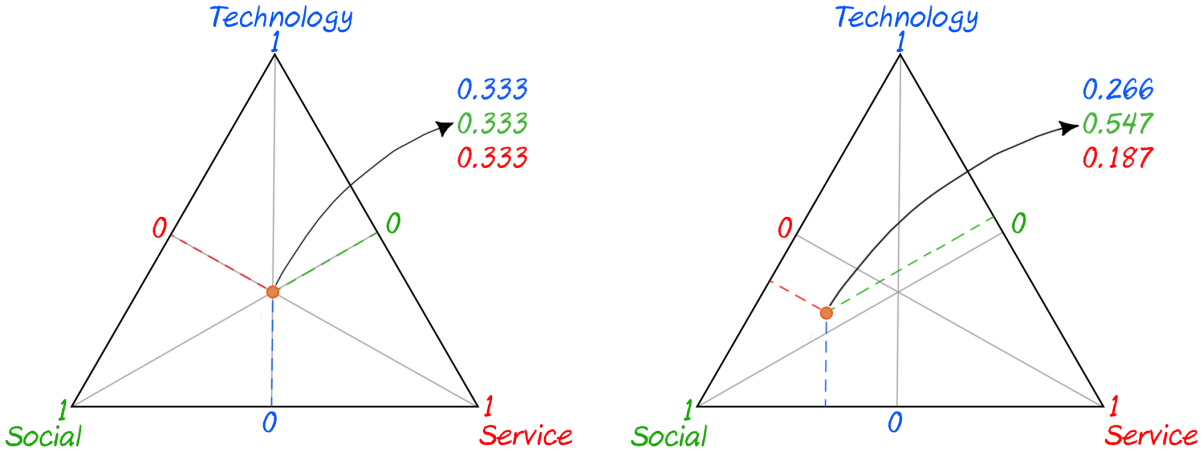


Figure 60. STS-Triangle with example marker for the aspect importance factors.

Selection method

Looking back at the chapter 'Evolutionary algorithms', four different selection methods were identified. Selection probability has a large influence on the convergence rate of an evolutionary algorithm. If the probability distribution is shifted towards favoring the fittest too much and too early, exploitation wins from exploration and instead of the global optimum, the algorithm converges to a local optimum. As mentioned before, the convergence pressure should increase over time. The traditional Roulette Wheel Selection and the Tournament Selection methods do not lent themselves well to this. Thus Linear Ranking Selection (LRS) and Exponential Ranking Selection (ERS) remain. In both Linear and Exponential Ranking Selection, the chromosomes are ranked based on their fitness; worst first, best last. Based on their rank (1 through 60), the generation size (60) and the convergence factor, they get assigned a probability.

The difference between LRS and ERS comes from the way the selection probability is calculated. LRS uses the following formula for this (Eiben, A. E., & Smith, J. E., 2003).

$$P_i = \frac{(2-s)}{N} + \frac{2i(s-1)}{N(N-1)}; \quad i \in \{1, \dots, N\}$$

Where *P* is the probability, *i* the rank of the chromosome, *s* is the convergence factor (a value between 1 and 2) and *N* is the population size. Depending upon the value of the convergence factor *s*, the distribution changes, as shown in Figure 61.

For ERS the formula is as follows (Thiele, L., & Blickle, T., 1995).

$$P_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}}; \quad i \in \{1, \dots, N\}$$

P is the probability, *i* the chromosome rank, *c* is the convergence factor (a value between 0 and 1) and *N* is the population size. Again the distribution changes depending on the convergence factor, in this case *c*, as shown in Figure 62.

While both selection methods can start out with an equal probability for all chromosomes, the exponential method can reach much higher probabilities and thus a much higher convergence pressure near the end. The linear method is limited in this by the population size. As LRS is limited, ERS is chosen, but care should be taken to not increase convergence pressure too quickly otherwise room for exploration will be limited.
After assigning the probabilities to all chromosomes, the sum of their probabilities should equal to 1. A uniform random value between 0 and 1 is then picked. Next, for each chromosome, the running sum of its and all previous chromosomes' probabilities is checked. If it is larger than the randomly picked value, that chromosomes is selected.

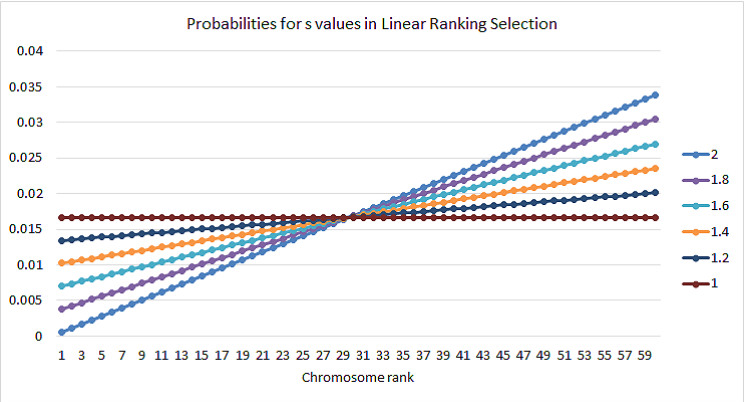


Figure 61. LRS-probabilites for 1 ≤ s ≤ 2.

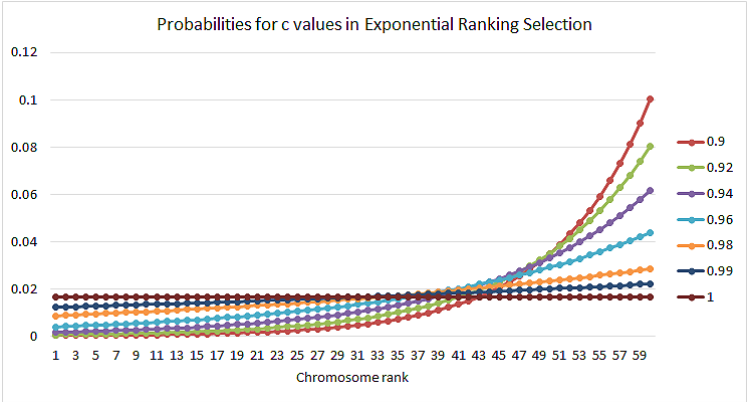


Figure 62. ERS-probabilites for 0.9 ≤ c ≤ 1.

Crossover and mutation

After the selection of the parents, crossover can take place to create the child's parameter values. Crossover is a rather simple process; for each parameter of the child, a coin is flipped and the parameter from either parent one or parent two is chosen based on that, see Figure 64. This creates a child which has traits from both parents, but without mutation new parameter values will never be introduced. Thus mutation takes places on the child's parameters. The probability that a fully new value for the parameter is chosen is based on the 'mutation probability' of the algorithm. As with the convergence pressure, this value changes over time. At first the mutation probability should start 'high', such as 3~5%, near the end it should be 0%.

There are many ways to move between these values. For example, one way is to use percentages, another way is to decrease it linearly and a third way is to use a sigmoid function. At first the mutation rate shouldn't decrease too much, then it should decrease gradually and near the end it should again not decrease too much anymore. In theory, in this way exploration is promoted for a while, then convergence can gradually take over and near the end exploitation should be nearly all there is. Looking at Figure 63, the sigmoid function corresponds best with this theory and is thus used to change the mutation probability over the course of the generations.

After crossover and mutation, the children's chromosomes for the next generation are ready and thus the next generation gets populated. At which point the simulation process starts anew.

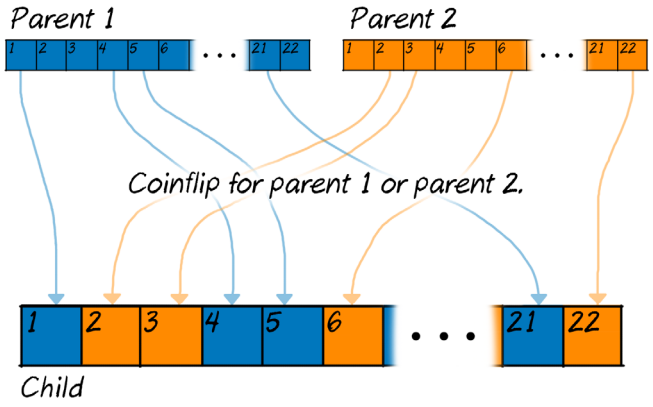


Figure 64. Crossover of two parents to create a child chromosome.

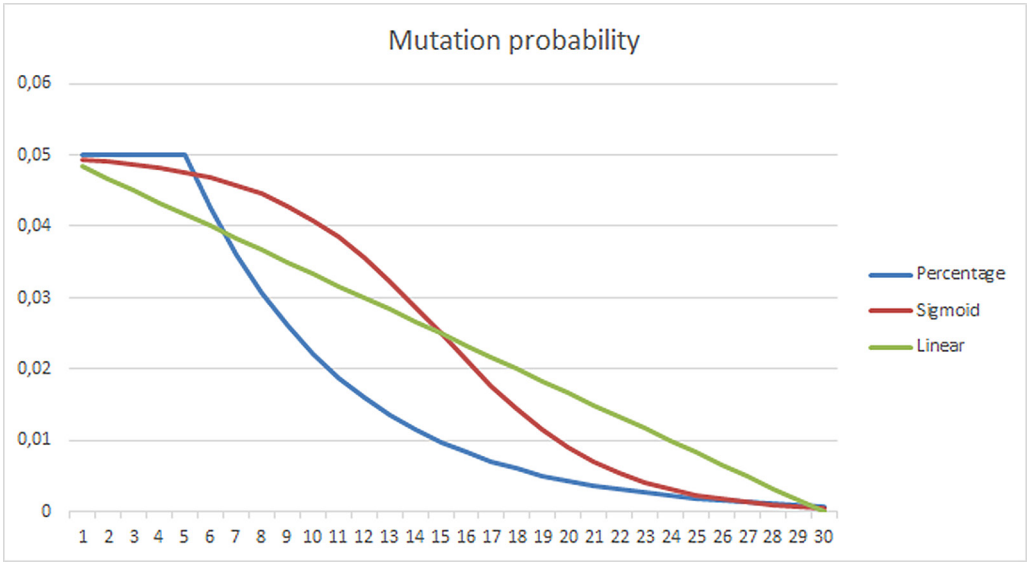


Figure 63. Different methods of moving through the mutation probabilites over the course of the genertions. Note; 'Percentage' stays level for the first 5 generations, then drops with 15% each step.

Results

Over the course of 2 weeks, the evolutionary algorithm was put to work and managed to complete six full runs on an Intel Nuc (NUC8i3BEH) with 16 GB of RAM. Each time, based on the results of the previous run, some adjustments were made on for example the E.A. weights and/or importance factors in order to explore their effects. The results of these runs will be presented in this paragraph. The implications of the results will be discussed in the next paragraph; 'Discussion'. For the full results overview of each run, including used weights and values, graphed averages, chromosome diversity and learned parameters; see Appendix 'N. Results of the Evolutionary Algorithm runs'

The E.A. weights and parameter values as used for the six runs can be found in Figure 65. Most fitness function weights remained fixed, only the goal weight was increased when it became apparent that the chromosomes were ignoring the goal altogether upon the completion of the first run. Instead of learning to reach the goal, the chromosomes learnt to avoid people and drive as little as possible to lower social forces, collision count, path length and stops; it would find a nearby spot with low chance of increasing the social forces and stay there until the end of the run, see Figure 66.

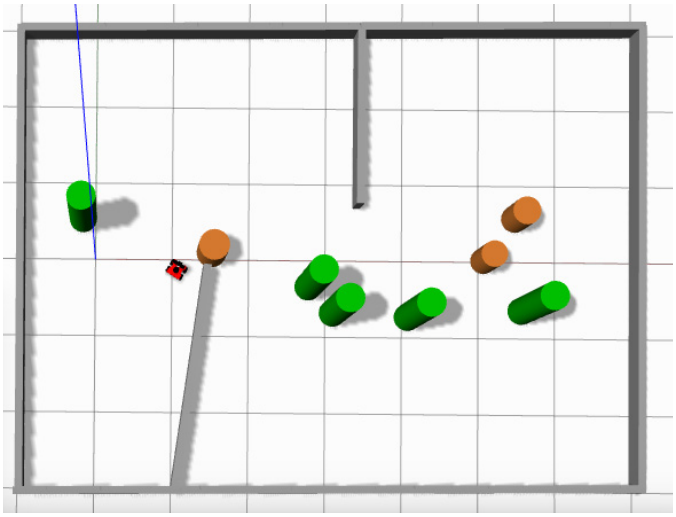


Figure 66. The spot where the chromosomes would stay to minimize the penalties.

In between runs, the values of the importance factors for each of the three aspects of the STS-triangle was adjusted. The exact values can be found in the figure below.

Weight / parameter name	Run 1	Run 2	Run 3*	Run 4	Run 5	Run 6
social_importance	0.333333	0.333333	0	0.42	0	0.6
service_importance	0.333333	0.333333	0.5	0.29	0.5	0.2
technology_importance	0.333333	0.333333	0.5	0.29	0.5	0.2
w_social_forces	1.5151515	1.5151515	1.5151515	1.5151515	1.5151515	1.5151515
w_collision_num	14.705882	14.705882	14.705882	14.705882	14.705882	14.705882
w_runtime	6.25	6.25	6.25	6.25	6.25	6.25
w_path_length	48.076923	48.076923	48.076923	48.076923	48.076923	48.076923
w_distance_to_goal	83.333333	83.333333	83.333333	83.333333	83.333333	83.333333
w_goal_reached	250	1000	1000	1000	1000	1000
w_stops_count	35.714286	35.714286	35.714286	35.714286	35.714286	35.714286
generation_size	30	30	30	30	30	30
population_size	60	60	60	60	60	60
max_mutation_probability	0.05	0.05	0.05	0.05	0.05	0.05
start_convergence_pressure	0.99	0.99	0.99	0.99	0.99	0.99
end_convergence_pressure	0.9	0.9	0.9	0.9	0.9	0.9
run_maxtime	55	55	55	55	55	55

*Figure 65. Table with the evolutionary algorithm's weights and parameter values as used for the six runs. *Run 3 did not have any pedestrians included during the simulations.*

In general the runs converge, but do so with fluctuations in the fitness values. These fluctuations can be observed when pedestrians were included in the runs. Part of this is caused by the random nature of the SFM navigation used by the pedestrians. Small deviations near the beginning result in large changes over time, comparable to the phenomenon of a double pendulum. These large changes also determine whether the robot can drive smoothly through the test environment or if it is often blocked by the pedestrians. The starting position, which is randomized at the beginning of each generation, also influences the generations starting 'luck'. If the pedestrians happen to start close to the robot or always meet the robot at the narrow middle area, the social forces penalty of that whole generation will be increased. These factors combined can explain the fluctuations in the fitness values when pedestrians were included in the runs.

In order to judge the learnt behavior of the runs, each top scoring chromosome of the final generation was employed multiple times (n=10) in their situation and observed. Based on these observations a marker can be placed in the STS-triangle, representing the observed behavior which can in turn be compared to the original marker which governs the importance factors of each aspect, see Figure 67.

Note, as the observations are interpretations of behavior, they are influenced by the subjectivity of the observer.

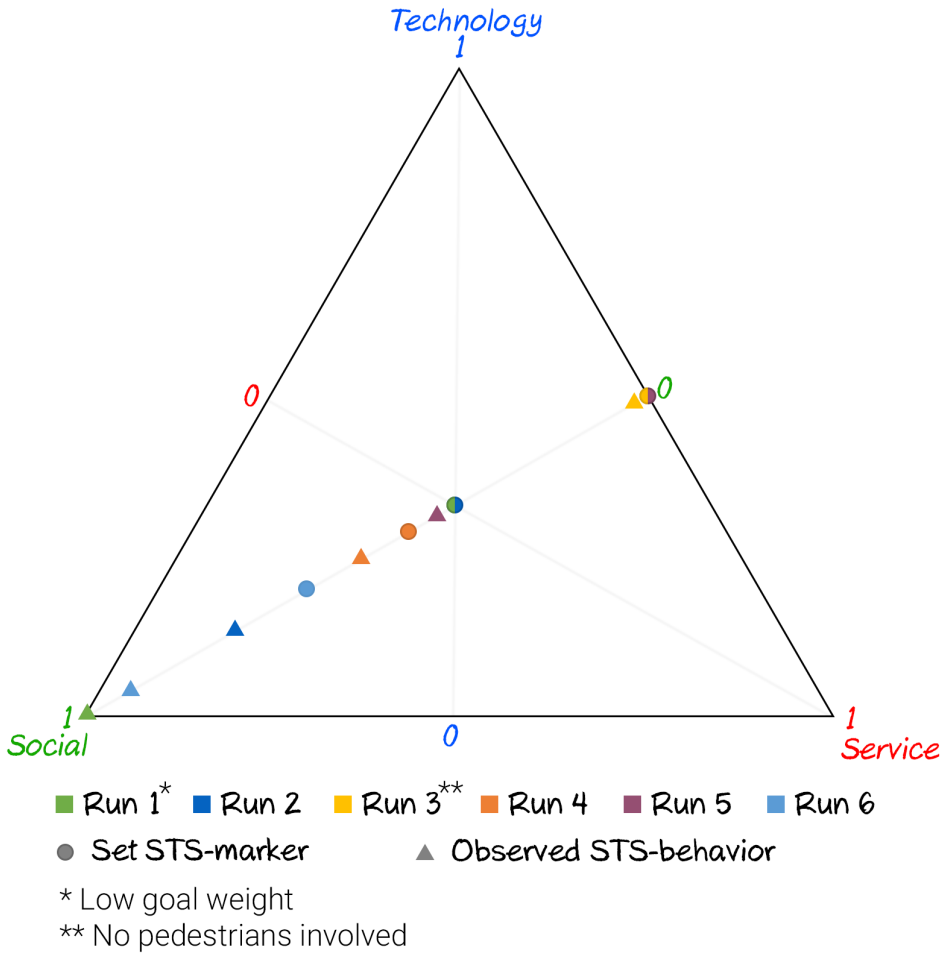


Figure 67. STS-triangle with the set STS-markers and the observed STS-behavior markers for all 6 runs.

Excluding the third run where no pedestrians were present, all runs are judged as more social than where their marker was placed. A logical explanation for this could be the built in safety features. As the time of flight sensors detect anything closer than 20 cm, the robot will slow down, anything closer than 10 cm and the robot will stop, independent of the learnt parameters. A robot slowing and/or stopping to avoid a collision can be seen as behaving more social than its learnt parameters would suggest.

There is one run which is difficult to position in the STS-triangle; run 6. Due to the exaggerated behavior of the robot; making wide turns, keeping an illogically large distance to pedestrians at one moment while nearly colliding with one the next. It also does not always reach the goal, and when it does its path is inefficient and it took almost twice the amount of time compared to other runs. Because of this, run 6 scores very poorly on the technology and service aspects, and dependent upon the interpretation, it is either asocial or too social. The behavior of run 6 can best be described as socially chaotic.

Discussion

All STS marker positions and thus the importance factors of the three aspects were placed along the 'social line'; the line where the social value is picked and as such governs the other two aspects. On this line the technology and service aspects are equally important.
As such, there have been no runs where the marker was placed in a position where technology and service weren't equal. The user test with the physical ROSbot is focused around evaluating the behavioral performance, and it thus made sense to stick to the social line in order to see the effect of mainly the social aspect upon the behavior of the robot. This however leaves large portions of the STS-triangle unexplored and still open for investigation in future projects.

A decision had to be made on how to determine when an E.A. run would be finished. One way is to look at the rate of change of the convergence, if the convergence flattens out, this means that most likely an optimum has been reached and the E.A. is done. The other option, which was the one applied in this project, is to give a maximum number of generations. At the end of the last generation, the best scoring chromosome is then the result of the E.A. run. This might seem like the less logical option of the two, but convergence isn't the only factor to consider. As time is limited and a number of learnt parameter sets (completed runs) is required for the user test, the decision was made to limit the amount of generations and thus the time per run. A possible issue which was unknowingly dodged by this choice was that of the fitness fluctuations. The fluctuations caused by the SFM pedestrians and their random starting positions over the generations would have complicated the accurate detection of a converged run.

With the six runs completed (of which five in the 'normal street' situation) a selection of three parameter sets was made to use in the user test. By picking run 2, 5 and 6, the placed STS-markers are spread furthest apart from one another in the STS triangle, giving the best chance to identify the effect of the social marker on the behavioral performance through the user test. The marker position of run 2 is neutral for all importance aspects. The marker of run 5 has a value of 0 for the social importance factor and the marker for run 6 has the highest value for the social importance factor (0.6) out of the three.

User test setup with the physical ROSbot

With the newly learned parameters and the addition of passing and crossing forces, the developed computational SFM-MPDM model can now be tested with users in pedestrian rich environments. During the tests, the physical robot will have to navigate said environment and will have the Go-Solo, Follow and Stop policies available.

The goal of the user test is to evaluate the physical robot's behavioral performance on the aspects of comfort (intuitiveness) and predictability (ease of use) and to evaluate the communication of intent through core vehicular movements.

During the study, the robot will be tested in the 'normal street' situation, which includes passing, following and crossing. For this, multiple participants are required at once. Based on the results a discussion and conclusion with recommendations for improvements of the robot's computational model will be written. These will also include reflections on the performance in the technological and product service aspects. For this study, the following research questions have been defined:

- *How comfortable is the behavior of the robot?*
- *How predictable is the behavior of the robot?*
- *How do participants perceive the communicated intent?*
- *What improvements are to be made to the computational model of the robot?*

User test setup

From prior experiments it became apparent that a walled in test area is required to keep noise levels and clutter to a minimum. The walls of the test environment can be low due to the low viewpoint of the LiDAR. Additionally, some obstacles are needed inside the test environment for the robot to drive around and to obscure direct vision of all pedestrians. The test environment should be of a size and layout similar to the EA digital learning environment (roughly 6 meters wide and 8 meters long) to facilitate longer ranged interactions, see Figure 68. Similarly, participants need this space for them to familiarize themselves with 'walking down the test environment' as immediate contact with the robot could influence the perceived comfort and predictability.

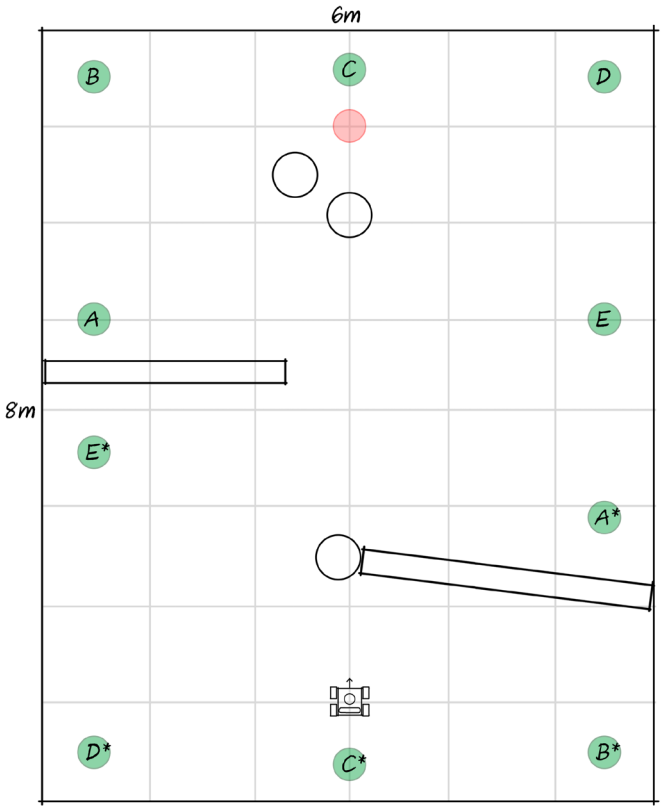


Figure 68. Floorplan for the user test, 'normal street' situation.

Before participation in the study, the participant will be asked to read and sign the 'HREC Informed consent' form, see Appendix 'L. HREC informed consent form'. No names, age or other personal information will be stored of the participants, only the written responses to the questions. Video-recordings will be anonymized through facial blurring before publication.

The participants will be asked to operate in a group of roughly 5 persons (minimum of 3, maximum of 6, preferred 5). Each group is asked to peform 6 runs. During those runs, three sets of learnt parameters will be employed by the robot, thus each set is tested twice per group of participant. The order of the parameter sets is switched around between groups and the groups are not told that the robot will be using different sets during the 6 runs.

The participants will get assigned a number or color, this will help with linking their responses in the form with the video-recordings. In the digital form the participants will first be asked to state the current run number, their assigned number/color and starting position. Next they are asked to rank the robot's behavior on comfort and predictability. Following this, they are asked how they perceived the communication of intent and to rank their experience of the robot's behavior in that run on a scale from negative to positive. In case of their final run, the participant is asked to supply any improvements for the robot and its behavior. For an example of the form, see Appendix 'M. User test digital form'.

During each run, the robot will log data. The logged data includes: active policy, run time, amount of stops, path length and the sum of the social forces for each interval. The data will be logged at the policy election cycle.

After all data is gathered, both from the robot's data logging as from the participants' responses; the robot behavior can be evaluated on comfort (intuitiveness), predictability (ease of use), and the communication of intent through core vehicular movements. The results will be processed by linking the participants' responses to the events in the runs, by studying the participants' experiences and by reflecting on the points of improvements as directly suggested by the participants. From these results, new guidelines and pointers can be set up and recommendations can be made for future work.

User test results

Over the course of several days the learnt parameter sets were tested with participants in the described environment. The results of these tests are presented in this chapter; first the parameter sets and their rated behavior as a whole are presented. After which a closer look is taken at the individual parameters and their impact.

The participants (n=42), divided over nine groups, all performed six runs totaling 54 completed user test runs with the ROSbot 2.0 Pro. After each run the participants filled in the digital form, totaling 252 digital forms and resulting in 84 forms per parameter set. An important note before presenting the results; many participants stated that the question about 'positive influence' was difficult to interpret and answer. Interpretations between participants varied and on occasion were completely opposite to one another. Therefore the results for this particular question cannot be used. The full results can be found in Appendix 'O. User test results'.

The results for parameter set 1 and set 2 are not significantly different from one another for comfort (p=.181), predictability (p=.251) or communication of intent (p=.646). Participants mostly agree with the statement *"The behavior of the robot feels comfortable to me"* for both set 1 and 2. Predictability is scored neutral, albeit slightly leaning towards to positive side. Intent communication shows similar results, mostly neutral but leaning slightly towards the positive side. This is surprising when one considers the importance factors for both runs as used during the E.A. parameter learning. For set 1, the social, technology and service importance factors were all neutral and equal to one another (0.333). For set 2, the social importance factor was set to 0, while technology and service were both set to 0.5. For these sets to show the same behavior with such different factors (and thus different fitness functions) is unexpected and suggests taking a look at the parameter sets themselves.

Set 3 clearly scores lower; there is a significant difference between set 3 and set 1 and 2 on comfort (p=.000 and .007), predictability (p=.000 and .001) and communication of intent (p=.000 and .004). Its behavior was described as chaotic and confusing by observers and participants alike. This is again unexpected when looking at the importance factors for this set. During parameter learning, the social importance was 0.6, which is the most social of the three sets. Technology and service were both set to 0.2. When looking at just the technology and service aspects, the behavior seems to make sense; the robot did not always reach its goal, it was slow and inefficient to get there and it frequently decelerated and accelerated. This all corresponds with a low importance value for technology and service. The chaotic social behavior on the other hand is surprising, scoring low on comfort, predictability and intent communication. This again suggest taking a look at the parameter set itself.

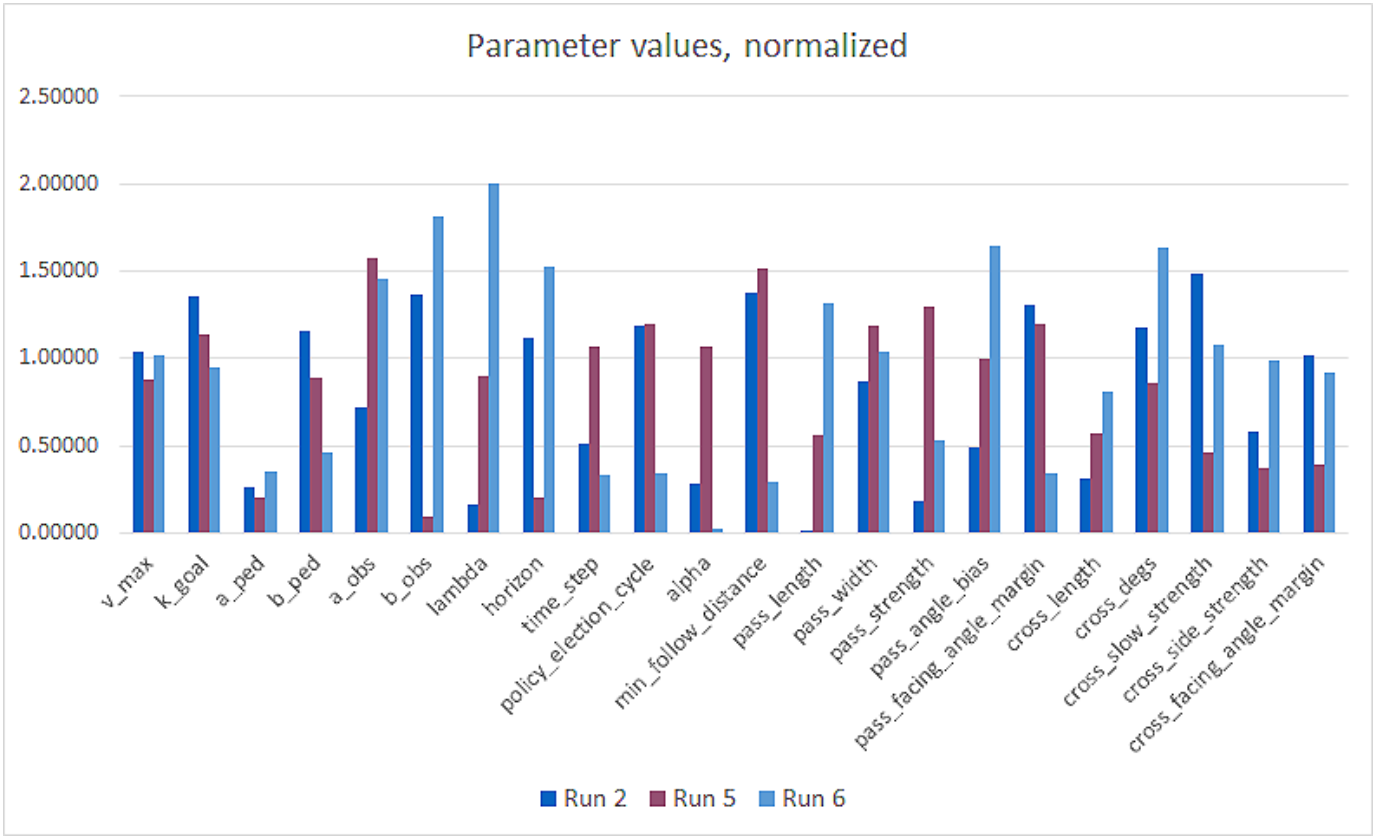


Figure 69. Learnt parameter values for run 2 (set 1), run 5 (set 2) and run 6 (set 3).

Figure 69 shows the parameters for run 2 (set 1), run 5 (set 2) and run 6 (set 3). Looking at the parameter sets shows some interesting values. Parameter set 1 has a passing length of near 0 meters. This results in the robot never employing the passing behavior and renders any other passing parameters useless for this set. Parameter set 2 shows a similar though less extremely short pass length of 1.2 meters. This can be considered as rather short sighted for a passing situation. Parameter set 3 shows an acceptable passing length of 2.65 meters. Similar parameter values can be found for the crossing lengths of all three sets; set 1, 2 and 3 have a crossing length of 0.45, 0.82 and 1.15 meters respectively. This can again be considered short sighted. Moreover, set 1 and 2 show a low crossing side strength, meaning that even if the crossing is employed at these short ranges the sideways avoidance is weak. Another notable parameter is the alpha value used during the MPDM. There is a large difference between the alpha values of set 1, 2 and 3 which corresponds to what one would expect from the importance factors. A higher social importance and thus lower technology and service importance should result in the alpha parameter (which signifies the importance factor of progress in the MPDM cost function) to be lower. The high level of importance for progress in set 2 is expected as social importance was set to 0. What is interesting here is the extremely low alpha value for set 3. Although a low alpha value is expected for a social importance value of 0.6, the learnt value is too low; 0.0037. This means the robot nearly does not care about making progress during the policy election. Combining this with the short time step, low policy election cycle and short minimum follow distance values it can explain the chaotic behavior of the robot.

And then there are the values which govern the core SFM itself. The maximum velocity, goal and pedestrian strengths are similar for all sets. The first difference can be found in the pedestrian range. Set 1 and 2 show similar

values, but the range for set 3 is clearly shorter. This means the presence of pedestrians won't have any effect until very close (<1m) to the robot. The obstacle strengths and ranges for set 1 and 2 are rather different. Set 1 has a below average strength but above average range, which balances each other out. In set 2, however, the obstacle strength is far above average and its range is far below average (<0.2m), this results in the robot not caring about obstacles until almost colliding with them, at this point, the general safety measures have already taken over. The most notable is again set 3. The obstacle strength and range values are well above average for both (range of ~3.4m). This results in the robot trying to keep an illogically large distance towards objects. Combining this with the large lambda value, causing the robot to consider objects and pedestrians behind it to be only slightly less important, adds onto the chaotic behavior.

Finally there is the communication of intent. When asked to what extend the participants agreed with the statements *"The behavior of the robot feels predictable to me"* and *"It was clear to me what the robot's intentions were"*, they scored them rather neutral. This leaves room for improvement on the communication of intent through core vehicular movements. Especially the passing and crossing, which should communicate intent, were lacking due to the short ranges. Additionally, about 29% of participants stated they would have liked to see other methods of intent communication, e.g. *"Some visual indication, like an LED strip indicating the current direction of motion can help the pedestrians understand the intent of the robot".*

User test discussion and conclusion

In this chapter the user test results of the previous chapter are discussed and conclusions are presented. Suggestions are given for alterations in the parameter learning method.

The results show that no significant difference could be found between parameter set 1 and 2. Their behaviors during the user test were similar and so are their rated comfort, predictability and communication of intent. Between set 3 and set 1 and 2 there is a significant difference. The behavior was described as socially chaotic or confusing. This is reflected in the ratings for comfort, predictability and communication of intent. Different behavior does result in different ratings on these aspects. Some notes should be made however on possible external influences on the user test. First of all, the ROSbot is rather small. The difference in size between the ROSbot and an actual autonomous delivery vehicle could impact the way the AGV is perceived. Secondly, the user test environment was set up in such a way as to promote both crossing and passing as well as obstacle avoidance. The setup included a bottleneck near the center of the environment to promote interaction. However, due to the nature of the bottleneck and the relatively short distances before and after this bottleneck, the effect of passing could not be tested on longer ranges. Thirdly, there is the possible demographic impact of the participants. Nearly all participants were Industrial Design Engineering students from the TU Delft, aged 18~28. It is unclear if and how this limited demographic spread influences the results, but for future tests a wider spread demographic sample set is advisable.

Looking back at the research questions on comfort, predictability and communication of intent the robots behavior isn't perceived negatively, but it does leave room for improvement. The passing and crossing wasn't employed optimally. The robot collided with participants on several occasions (n=9, 16.7%) and the intent was not communicated clearly.

The analyzed learnt parameter values of the sets gave some insights into why performance was not optimal. To solve the behavioral issues caused by the parameters, first the parameter learning must be improved. As a first suggestion for improvement the amount of parameters should be decreased and default values should be used for those which are not learnt. For example, in passing and crossing, if any of the length, width, strength or margin values was low, the method was very likely to not be employed. As such these 5 passing and 5 crossing parameters should be excluded from parameter learning and instead should be replaced with just two; one parameter for each method which determines the effective strength of the resulting forces through a value between 0 and 1. Similarly, parameters which govern the policy election and horizon length of the MPDM should also be set to a default value. This should prevent excessive forward simulation on one hand and infrequent (<2 Hz) policy changes on the other.

A second alteration, which can be applied if time is not an issue, is to increase the population size. In this way a larger solution space can be explored, increasing the chances of finding the overall optimum. In addition to this, one might include 'elitism' to the E.A. With elitism you ensure that the best chromosome of the generation always survives into the next. This is especially useful in cases where a solution close to the optimum is found in early generations, when mutation rate is still high and selection pressure is low. Finally, there is the possibility to change from stopping at a fixed maximum number of generations to stopping by looking at the rate of change of the convergence. If the convergence flattens out, this means the run has found its optimum and as such the run can be stopped.

Looking back at the user test runs, we can also reflect on the performance of the robot in the technology and (product) service aspects. Service wise, set 1 and 2 performed very well; the runtime was short, the goal was always reached and the path length was close to optimal. Technology wise, set 1 and 2 could improve by decreasing the amount of stops; the robot could sometimes be seen to stop in order to rotate towards its (sub)goal on the spot. Set 3 doesn't score well on technology nor service. It often did not reach the end goal, drove inefficiently, stopped often and sometimes ended at a large distance away from the goal. These results for the 3rd set were caused by its socially chaotic behavior as discussed before.

To conclude

In these final chapters the results of the project are discussed, conclusions are drawn and recommendations are given for future projects.

Discussion

Over the course of this graduation project a computational mechanism was designed for an autonomous guided vehicle in pedestrian rich environments. The original design goal was stated as such:

"Research and design a computational mechanism for an autonomous guided vehicle based on the social, technology and product service needs by utilizing a machine learning method to deal with a variety of situations while expressing its (change of) intention."

As such, the context of pedestrian behavior and mobile (delivery) robots was researched on three aspects; social, technology and (product) service. These three aspects combined formed the STS-triangle, which served as the overarching framework for the research and exploration phase. In this phase the foundation for the social navigational model was presented; the Social Force Model. The SFM was expanded upon by Multi-Policy Decision Making, which adds forward simulation and the option to choose between three policies: go-solo, follow and stop. By utilizing MPDM, the (virtual) robot can predict and decide which policy will bring the most progress and the least social disruption. And although this addition helped in making the robot behave more polite, it was still found to be lacking in passing and crossing situations.

Passing and crossing

From pedestrian observations it became clear that pedestrians, in the Dutch context, follow the right hand rule during passing. They look ahead and adjust their paths to their right, if possible, to timely communicate their intent of passing and to prevent collisions. During crossing pedestrians judge the situation and, if so required, adjust their speed and heading. Usually the one which alters the most is the one which crosses behind the other. The robot, while utilizing SFM-MPDM, did not show these anticipating and correcting behaviors in passing or crossing situations. As such, methods were designed and added to the social navigational model which checked for possible passing and crossing pedestrians. If found, the methods would add sideways and/or slowing forces to the robots core SFM, which in turn made the robot anticipate the situation and adjust its trajectory accordingly.

The SFM-MPDM with passing and crossing was tested with users in a digital environment. From the user test it became apparent that the robots behavioral performance was judged as quite human like, comfortable and predictable, but the social navigational model still required some adjustments and tuning. The viewing cone for passing was changed to a rectangular area which rotates with the agents heading. This change helped promote a more consistent distance keeping during passing. The viewing cone for crossing was also adjusted. Its range was increased by 50% to better promote looking ahead. Participants also mentioned that the robot should maintain a larger minimal distance towards people, as it was sometimes found to be following and driving too closely.

Learnt behaviors

During the research and exploration phase eight core situations were identified. The behavior of the robot should be adapted based on the situation it finds itself in and as such, behaviors for different situations had to be found. The behaviors of the robot are governed by their parameter sets. To find these parameter sets an evolutionary algorithm was employed. The evolutionary algorithm emulated Darwinian evolution through fitness, selection, crossover and mutation in the Gazebo virtual physics world. The learnt behavior sets were put to the test with participants (n=42) and a physical robot in an environment which represented a normal street situation with obstacles, pedestrians, passing and crossing. The robot showed

some unexpected behavior however. The parameter set which used a neutral position for all three aspects in the STS-triangle was judged as no different from a parameter set which had no social aspect during parameter learning. Moreover, the parameter set which should have produced the most social behavior towards pedestrians was found to be socially chaotic and was scored lowest on comfort, predictability and communication of intent. The suspected underlying reasons for these surprising results were the underlying parameters of each set. Some counterproductive parameter values were learnt and linking (groups of) parameters to exact parts of the behaviors was found to be difficult. It is thus apparent that the evolutionary algorithm requires adjusting in order to learn parameter sets which better match the intended behavior as marked in the STS-triangle. More on this can be found in the chapter 'Recommendations' on page 88.

Requirements and wishes

At the end of the research and exploration phase a program of requirements and wishes was set up. These requirements and wishes were stated in a way such that they apply to the autonomous delivery robot as if fully implemented and operational within the pedestrian context. A short-list was made on which the end result was to be reflected. Even though the state of the project is far from 'fully implemented and operational in the pedestrian context' same valuable comments can be made.

To a certain extend the robot adjusts its behavior to the situation through the use of MPDM. The on the fly parameter set changing however, was not yet implemented in this project (R. S1). A good start was made on finding the right behavior which is seen as intuitive (predictable) and comfortable (ease of use). Room for improvement is clearly present, but parameter set 1 and 2 weren't negatively perceived (R. S2).

The robot tries to maintain a comfortable distance towards pedestrians, but at times the pedestrian is not detected and a collision could occur. Similarly, if the pedestrian comes too close on purpose, the robot does not have an option to drive backwards and maintain the distance (R. S4 & R. S5). The virtual robot can be seen to exhibit the Dutch pedestrian conventions of passing on the right, crossing behind and slowing and waiting for others. The learnt parameters for the physical robot however, upon closer inspection, were found to be lacking in passing and crossing. As such these are to be tested again with more profound parameter values (R. S6 & R. S7).

Through the use of MPDM, the robot can look ahead, predict and decide upon the policy which is both progressive and which results in the least social disruption (R. S8, W. S5 & R. T7). During the development and testing with the ROSbot 2.0 Pro, the intensive calculations were performed on a ROS connected laptop (Ubuntu 16.04 virtual machine with quad core and 8GB RAM). The robot itself only served as the sensor and actuator. It was not tested if the ROSbot could run the calculations on its own hardware (W. T1).

The robot can be seen following a straight path if no other agents or obstacles are nearby (W. T4). For smaller angular adjustments the robot exhibits a turning radius which is proportional to its translational velocity, but for larger adjustments the robot prefers to stop and turn on the spot instead (W. S2).

The communication of intent through core vehicular movements alone was judged to be neutral. A number of participants (n=12) mentioned they would have liked to see other methods of intent communication such as visual indicators and LED lights (R. S11).

Conclusion

Looking back at the design goal, it can be said that during this project, a large part of the computational mechanism was achieved. A social navigational framework was developed based on the Social Force Model with the addition of Multi-Policy Decision Making. Walls, static obstacles and dynamic obstacles (pedestrians) could be detected through the use of LiDAR and an adjusted obstacle detector structure of ROS nodes. New additions were the Social, Technology and Service triangle method, the passing and crossing to the SFM and the use of an evolutionary algorithm to learn parameters for the SFM-MPDM, see Figure 70.

Through the user test it became clear that the robots behavior influences the experience of the pedestrians, but it is unclear which (group of) parameter(s) exactly influences which part of the behavior. There are certain expectations to what impact a parameter or group of parameters has, but at which times and to what effect this controls parts of the behavior is difficult to identify. Nevertheless a good foundation has been laid down for future (graduation) projects through the development of the social navigational model and a way to learn parameters for the many different situations an autonomous delivery robot can encounter.

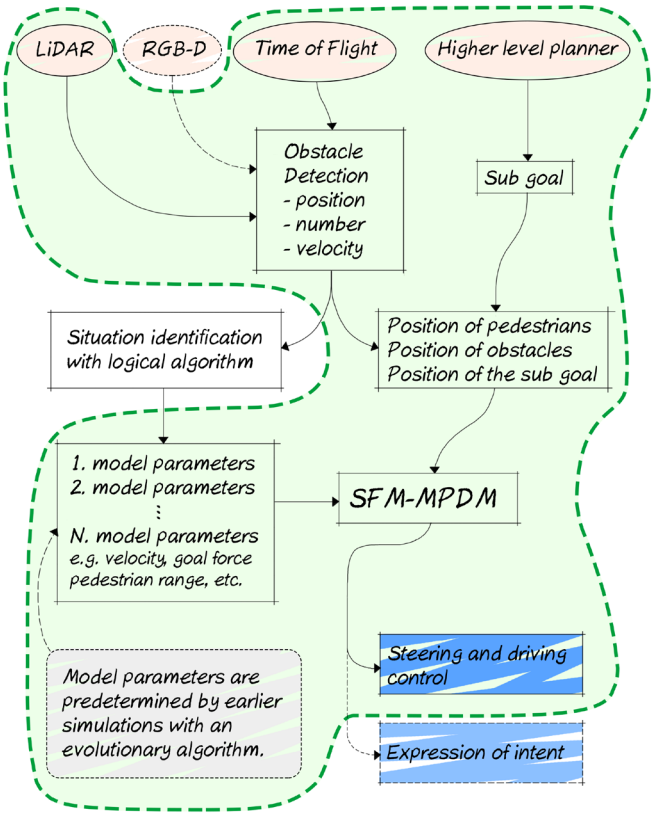


Figure 70. The computational mechanism, implemented parts are marked green.

Recommendations

Within the time that was available for this graduation project, choices had to be made on which parts should be left out and which parts should be focused on. As such, recommendations are in order for those parts which were left out of scope, but also for those parts which were addressed but which can be improved upon.

First of all, in the chapter 'Disparity issues between virtual and physical worlds' on page 66 the issue of drift was discussed. As the robot moves around, it tries to keep track of where it is through odometry. This odometry isn't perfect however and as such a positional and angular drift occurs. Another issue which occurs when the robot is moving is the invalid detection of static obstacles as dynamic.
For both these issues it is advisable to include Simultaneous Localization and Mapping (SLAM), which can help with correcting the odometry and excluding static obstacles from being marked as dynamic.

Computational mechanism
As mentioned in the discussion, the robot does not identify the situation nor alters its parameter set to the situation. The situation identification is an important next step in the further development of the computational mechanism. Interesting options for the development of the situation identification are a classical algorithm of conditionals and effects, or instead to build and train a neural network which takes in the sensor data and outputs a likelihood score for all of the situations, upon which a decision can then be made.

Other improvements to the computational mechanism are the inclusion of the RGB-D camera. Through image recognition a pedestrian standing in front of the robot can be detected, which can augment the detection through LiDAR. The depth sensing of the camera can also augment the mapping of the environment and detect obstacles in front which the LiDAR might have missed.

Social navigational model
The choice of switching from a narrow viewing cone for passing to a rectangular detection area was made without first exploring other options and shapes. The change had a positive effect on the detection and behavior of agents during passing, but it could be interesting to explore other shapes, such as an elongated trapezoid with a base positioned slightly behind the agent.

Another interesting area of improvement are the policies of the Multi-Policy Decision Making. The core policies of MPDM (go-solo, follow and stop) were used, but no additional policies were developed. One such additional policy could be to allow the robot to drive backwards. These could be valuable in situations where a, the robot is stuck because of an obstacle and b, a stalemate with a pedestrian was reached where the robot cannot go forward but the pedestrian does not make way.

Size of the robot
The user tests with the physical robot were performed with the ROSbot 2.0 Pro. This robot served as an ideal platform for the development of the computational mechanism, but its size might have had an influence on the user test results, especially in regard to the rated comfort. It can be hypothesized that an actual, more sizeable delivery robot will not be seen as equally comfortable with the same behavior, but this is yet to be tested.

Evolutionary algorithm and the STS-triangle
As mentioned in the discussion, the resulting learnt parameter sets do not always match the intended behavior as set by the marker in

the STS-triangle. As such it is recommended to first improve the EA framework. One such recommendation is to decrease the number of parameters which can be influenced in an attempt to decrease complexity and help with better linking parameter values to learnt behavior. Another recommendation would be to increase the population size of the generations. This should help with exploring a larger solution space which increases the chances of finding the overall optimum. In addition to this, elitism could be included to the evolutionary algorithm. Through elitism one can ensure that the best performing chromosome always survives into the next generation. This should help with keeping those chromosomes which are performing well in the early stages, where selection pressure is low and mutation rates are high.

A final recommendation is to explore outside of the 'social line' in the STS-triangle. During parameter learning, markers were positioned inside the triangle which always balanced the technology and service aspects. As such it was not investigated what the effects are on the learnt behavior when one shifts the balance to increase the importance of the technology or service aspect.

References

Amazon. (2019, January 23). Amazon Scout [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=peaKnkNX4vc

Battery University. (2019). Discharging at High and Low Temperatures. Retrieved October 25, 2019, from https://batteryuniversity.com/learn/article/discharging_at_high_and_low_temperatures

Besselink, I. J. M., Wang, J., & Nijmeijer, H. (2013). Evaluating the TU/e Lupo EL BEV performance. In 2013 World Electric Vehicle Symposium and Exhibition (EVS27) (pp. 1-12). IEEE.

Chen, Y. F., Everett, M., Liu, M., & How, J. P. (2017b). Socially aware motion planning with deep reinforcement learning. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1343-1350). IEEE.

Chen, Y. F., Liu, M., Everett, M., & How, J. P. (2017a). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In 2017 IEEE international conference on robotics and automation (ICRA) (pp. 285-292). IEEE.

de Groot, S. (2019). Pedestrian acceptance of delivery robots. (Master Thesis, Delft University of Technology, Delft, The Netherlands). Retrieved from http://resolver.tudelft.nl/uuid:f9e8c003-c8fc-4075-bff3-0d54e0f0fecb

Deutsche Post. (2017, October 29). Kollege “Postbot”: Postbotin bekommt Hilfe von Roboter [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=Fqf7PwNVYFQ

DHL. (2019). Alle specificaties op een rij | DHL Parcel. Retrieved October 30, 2019, from https://www.dhlparcel.nl/nl/zakelijk/support/verzenden/afmetingen-en-gewicht

Domino’s. (2016, March 21). Domino’s Pizza Delivery Robot Is The Future - Louder News [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=mCntGKp-JM0

Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. IEEE Transactions on evolutionary computation, 3(2), 124-141.

Eiben, A. E., & Smith, J. E. (2003). Introduction to evolutionary computing (Vol. 53, p. 18). Berlin: springer. Retrieved January 21st, 2020, from https://www.slideshare.net/guest9938738/genetic-algorithms

Federatie Nederlandse Vakbeweging. (2011). Bereken maximaal tilgewicht online-light versie. Retrieved October 30, 2019, from https://www.arbobondgenoten.nl/arbothem/lichblst/lift.htm

FedEx. (2019a). RoxoTM | The Future of FedEx. Retrieved September 17, 2019, from https://thefuturefedex.com/

FedEx. (2019b, February 26). Meet the FedEx SameDay Bot [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=N0rt_HB7vd4

Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. Physical Review E, 51(5), 4282–4286. https://doi.org/10.1103/PhysRevE.51.4282

Jebari, K., & Madiafi, M. (2013). Selection methods for genetic algorithms. International Journal of Emerging Sciences, 3(4), 333-344.

Kiwi Campus. (2018, May 25). Kiwi's robots deliver food to hungry Berkeley students [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=eGNpqvHAydE

Laffey, J. M., & Amelung, C. J. (2010). Using Notification Systems to Create Social Places for Online Learning. Handbook of Research on Social Interaction Technologies and Collaboration Software, 170–180. https://doi.org/10.4018/978-1-60566-368-5.ch016

Mahadevan, K., Somanath, S., & Sharlin, E. (2018, April). Communicating awareness and intent in autonomous vehicle-pedestrian interaction. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (p. 429). ACM.

Marble. (2018, August 20). Marble Robotic Delivery Vehicle Comes to Arlington [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=O8vjCc3FhsA

Mehta, D., Ferrer, G., & Olson, E. (2016). Autonomous navigation in dynamic social environments using Multi-Policy Decision Making. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). https://doi.org/10.1109/IROS.2016.7759200

ministerie van Sociale Zaken en Werkgelegenheid. (2019, March 14). Tillen en dragen. Retrieved October 30, 2019, from https://www.arboportaal.nl/onderwerpen/tillen-en-dragen

Postmates. (2019, March 21). Postmates Presents New NVIDIA Jetson AGX Xavier Equipped Delivery Robot at GTC [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=pckZFC_hs50

PostNL. (2018). Een doos of verpakking kiezen. Retrieved October 30, 2019, from https://www.postnl.nl/versturen/pakket-versturen/hoe-verstuur-ik-een-pakket/doos-of-verpakking-kiezen/

Premium Disposables. (2019). Pizzadozen » Premium Disposables. Retrieved October 30, 2019, from https://premiumdisposables.nl/product-categorie/pizzadozen/?gclid=EAIaIQobChMIkrTm7rfE5QIVFc13Ch1-zwDkEAAYASAAEgL3GPD_BwE

Przybyla, M. (2018). obstacle_detector (Software). Poznan. Retrieved December 2nd, 2019, from https://github.com/tysik/obstacle_detector

Risto, M., Emmenegger, C., Vinkhuyzen, E., Cefkin, M., & Hollan, J. (2017). Human-vehicle interfaces: the power of vehicle movement gestures in human road user coordination.

Saini, N. (2017). Review of selection methods in genetic algorithms. International Journal Of Engineering And Computer Science, 6(12), 22261-22263.

Shiffman, D. (2015, August 11). 9.x: Genetic Algorithms and Evolutionary Computing - The Nature of Code [YouTube]. Retrieved September 20, 2019, from https://www.youtube.com/watch?v=6l6b78Y4V7Y

Starship Technologies. (2018a). Starship Deliveries - Apps on Google Play [Mobile app]. Retrieved October 30, 2019, from https://play.google.com/store/apps/details?id=xyz.starship.deliveries

Starship Technologies. (2018b, April 30). Starship Campus Delivery Service with Robots [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=P_zRwq9c8LY

Swiss Post. (2017). FACTSHEET STARSHIP DELIVERY ROBOT. Retrieved from https://www.post.ch/-/media/post/ueber-uns/medienmitteilungen/2017/factsheet-lieferroboter.pdf?la=en

Szafir, D., Mutlu, B., & Fong, T. (2014, March). Communication of intent in assistive free flyers. In 2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI) (pp. 358-365). IEEE.

The Spoon. (2019, March 18). ArticulATE Q&A: Why Starship's Delivery Robots are as Wide as Your Shoulders. Retrieved October 30, 2019, from https://thespoon.tech/articulate-qa-why-starships-delivery-robots-are-as-wide-as-your-shoulders/

Thiele, L., & Blickle, T. (1995). A comparison of selection schemes used in genetic algorithms. TIK Report of Swiss Federal Insititute of Technology.

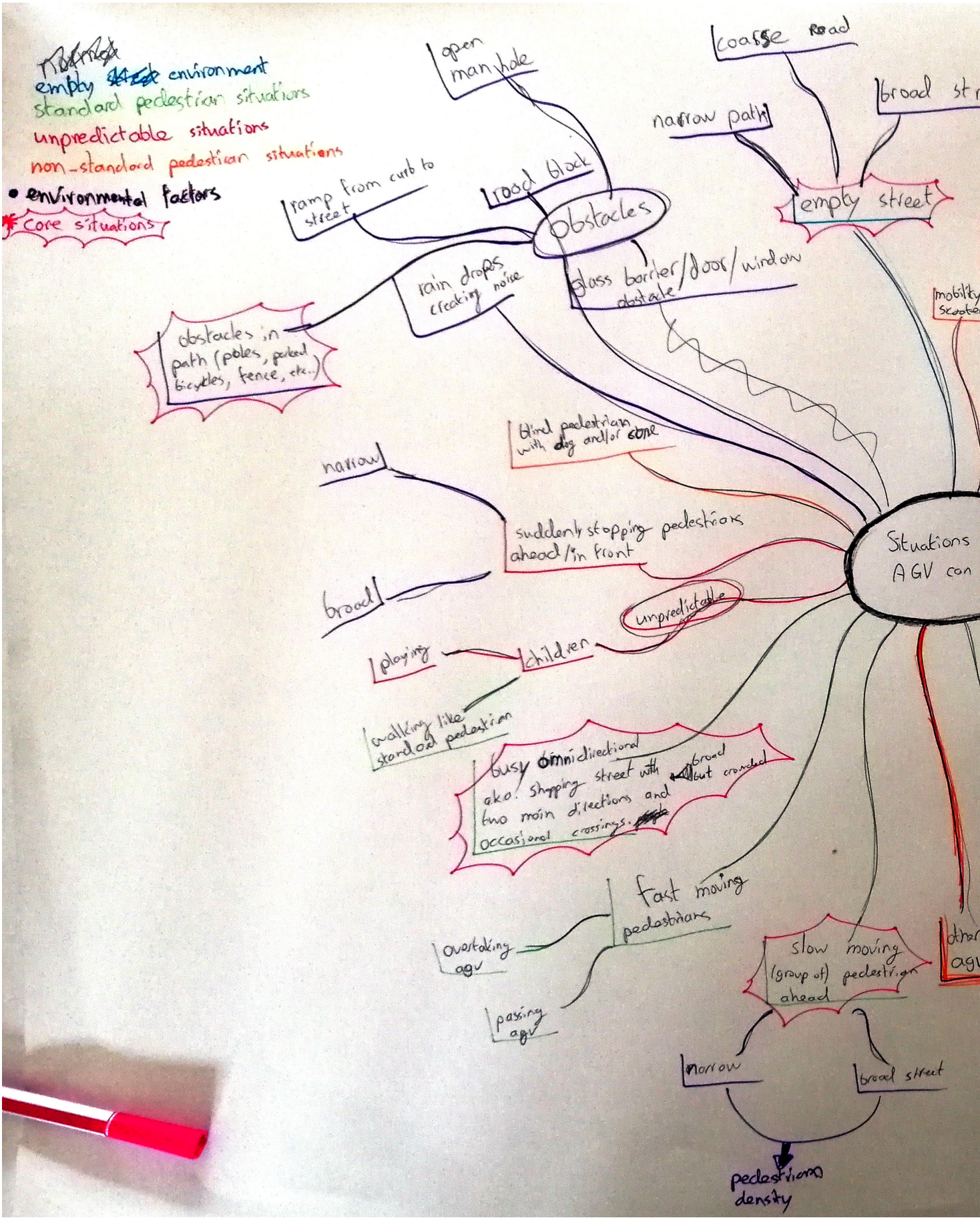
Wang, J. (2016). Battery electric vehicle energy consumption modelling, testing and prediction: a practical case study Eindhoven: Technische Universiteit Eindhoven

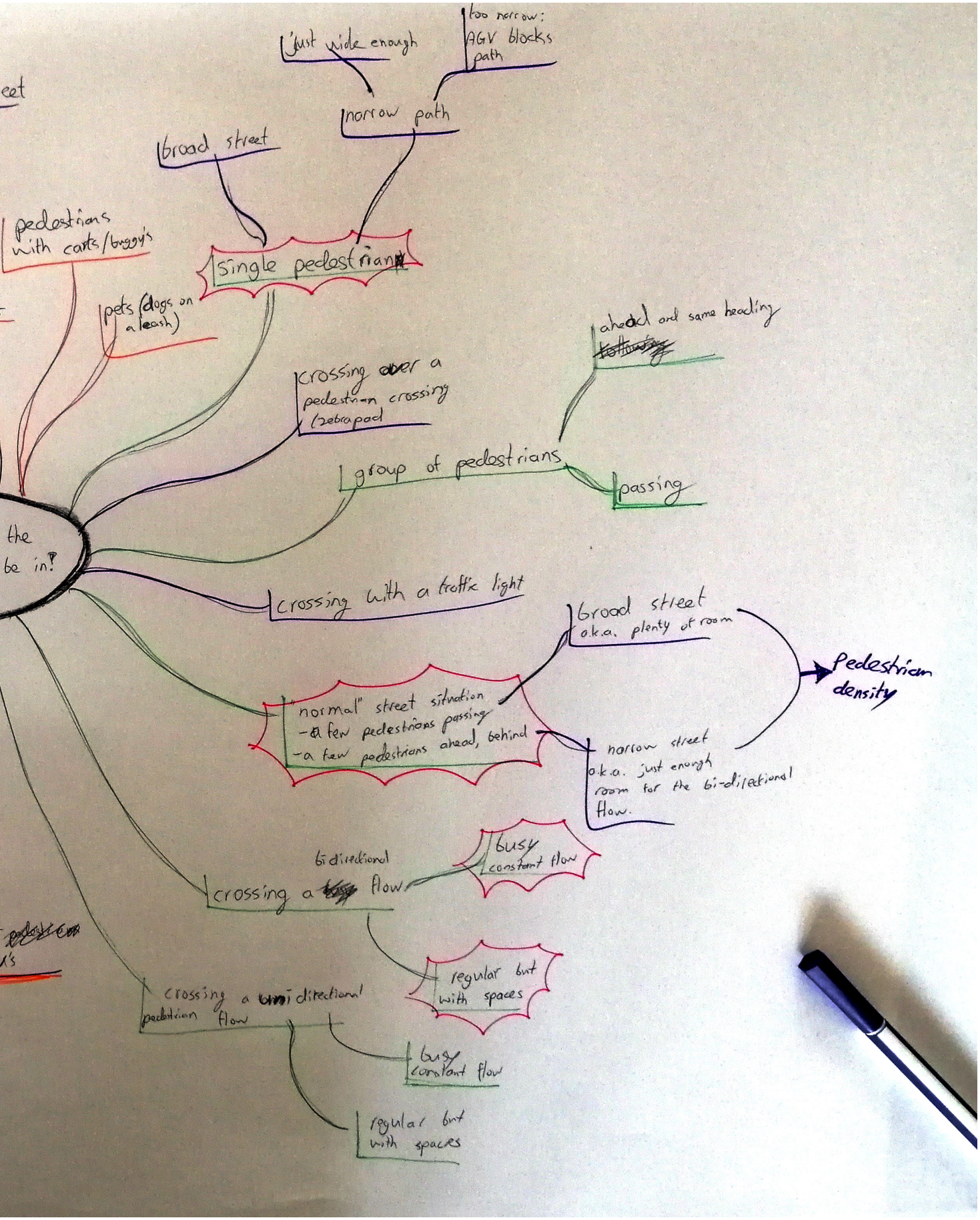
zeropointmoment. (2019, March 13). CarriRo Deli [YouTube]. Retrieved September 17, 2019, from https://www.youtube.com/watch?v=AI0l3ySnAiA

Appendices

Appendices to the report such as the original Design Brief form, raw interview data, photos of mindmaps, code, user test setups and results.

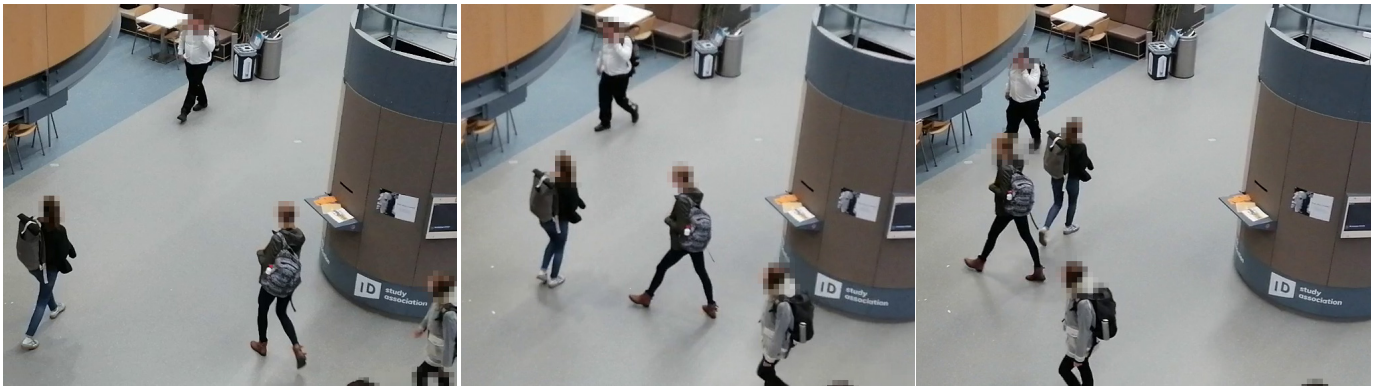
B. Mindmap of AGV situations




eet
just wide enough
too narrow: AGV blocks path
narrow path
broad street
pedestrians with carts/buggy's
single pedestrian
pets (dogs on a leash)
crossing over a pedestrian crossing (zebra pad
ahead and same heading
group of pedestrians
passing
the
be in?
crossing with a traffic light
broad street o.k.a. plenty of room
"normal" street situation
- a few pedestrians passing
- a few pedestrians ahead, behind
narrow street o.k.a. just enough room for the bi-directional flow.
Pedestrian density
bi directional
crossing a flow
busy constant flow
regular but with spaces
crossing a unidirectional pedestrian flow
busy constant flow
regular but with spaces

C. Pedestrian observations

Adjust and cross behind

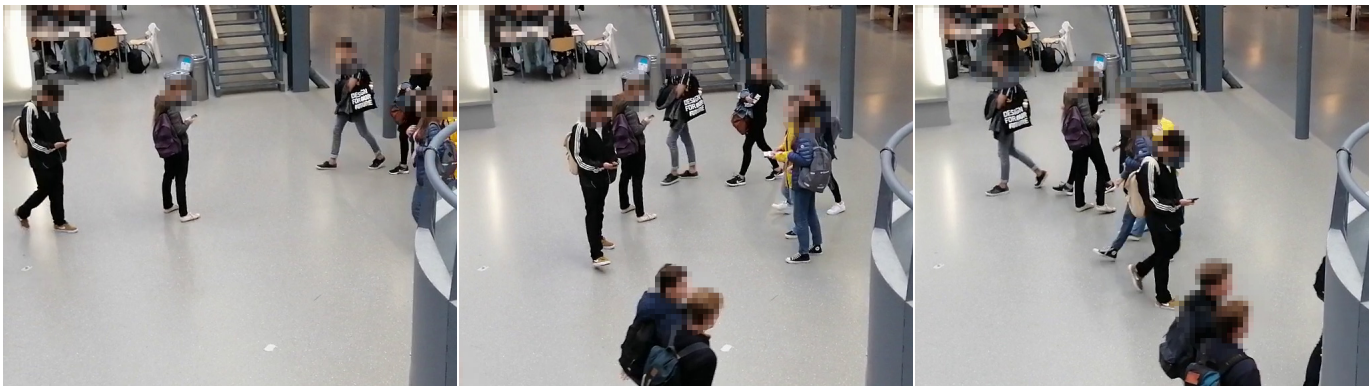


In this crossing situation, the female with the backpack in the middle of the picture can be seen to want to cross in front (left), notice this will result in a collision and adjust (middle) and cross behind instead (right).

Right hand rule during passing #1

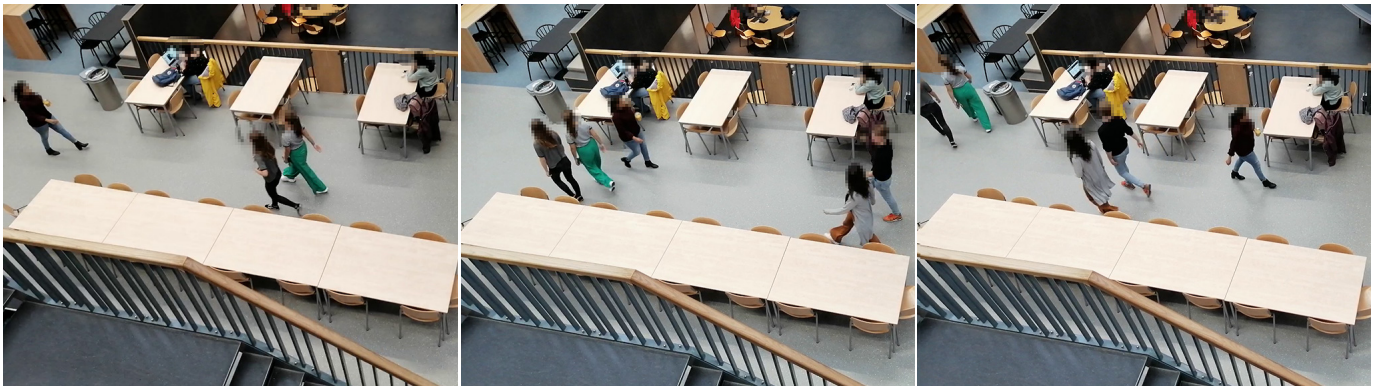
In this passing situation, the male in the blue t-shirt (left) keeps to the right while passing (middle and right), despite his his initial left biased position (left).

Adjust while passing



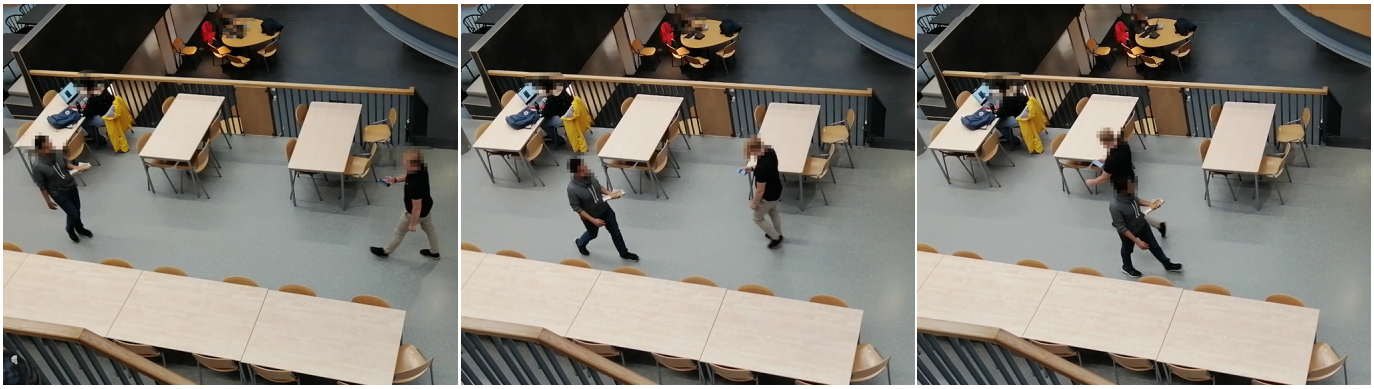
In this passing situation, the male in black and white is not paying attention and looking at his phone (left), he notices the oncoming group (middle) and passes the group on the right (right).

Passing on the left side



In this passing situation, the female in black sees the oncoming duo and decides to neglect the right hand rule in (left) in favor of passing 2 groups on the left (middle and right).

Right hand rule during passing #2



In this passing situation, the two males are faing eachother mostly head-on. The male on the left notices the other is looking at his phone (left) and decides to adjust a bit more and pass on the right (middle and right).

D. Pedestrian interview and data

Goal of the interview
Discover pedestrian behavior during passing and crossing for implementing in MPDM-SFM.
Discover pedestrian expectations of the robot, identify differences.
Gain insights and pointers for AGV communication and investigate leaning.

Before interview
Ask:
If the participant understands all data will be handled anonymously.
Permission to record with a voice recorder
If the participant wants their voice recording deleted after project finishes.
Permission to publish the processed and anonymous data in my graduation thesis.
To fill in their age category (0-14, 15-24, 25-39, 40-64, 65+).
To fill in male, female or other.

Pedestrian natural behavior
Passeren
- Stel u loopt in een voetgangersgebied (bijv. een winkelstraat, voetpad of een stoep) en er komt een voetganger u tegemoet, wat doet u?
- Welke kant gaat u op om de ander te laten passeren?
- Wanneer zou u de ander aan de andere kant passeren?
- Zit er een verschil tussen uw gedrag en het type voetgangersgebied?

Kruisen
- Stel u loopt weer over de stoep en er komt een voetganger vanaf de zijkant aanlopen, jullie paden kruisen, wat doet u? (Eventueel voordoen)
- (Welke manier van aanpassen doet u om een botsing te voorkomen? Inhouden, versneller of bijsturen?)
- Maakt het uit van welke kant de persoon u kruist?
- Bij het besturen van een voertuig gaat rechts voor, past u dit ook toe als voetganger?
- Waarom wel of niet?
- Hoe bepaald u anders wie er voorlangs gaat en wie achterlangs kruist?
- Wat voor invloed heeft de hoek waarmee jullie elkaar kruisen? (Bijv. loodrecht, kleine hoek in dezelfde richting, kleine hoek tegenover gestelde richting)
- Expected autonomous robot vehicle behavior.

Algemeen
- Wat voor gedrag verwacht u van een autonoom voertuig, zoals hierboven zichtbaar, die zich door voetgangersgebieden verplaatst?
- Waarom?

Passeren
- Wat voor gedrag zou u willen zien van een autonoom voertuig als hij u gaat passeren?
- Moet hij standaard rechts aanhouden?

- Hoe is dit vergelijkbaar met wat u van een tegemoetkomende voetganger verwacht? En hoe is het verschillend?

Kruisen
- Wat voor gedrag zou u willen zien van een autonoom voertuig als hij u gaat kruisen?
- Zijn er regels waar het voertuig zich aan moet houden?
- Hoe is dit vergelijkbaar met wat u van een kruisende voetganger verwacht? En hoe is het verschillend?

Robot intent communication
- Communicatie van verandering in snelheid en richting
- Zou u het prettig vinden als de robot u laat zien wat hij doet en/of van plan is om te doen? Denk aan van richting veranderen, vertragen, versnellen, stoppen.
- Hoe zou willen dat de robot dit aan u communiceert?
- Wat vindt u van verlichting als communicatiemiddel? (remlichten aan achterkant, koplampen voorop, oranje knipperlicht aan zijkanten)
- Wat vindt u van geluid als communicatiemiddel? (spreken / nabootsen van een elektrische motor waarin toonhoogte snelheid aangeeft / signalen)
- Zou u het zien van een kleine verandering in snelheid of richting als hint gebruiken dat de robot daarna een grotere verandering zal gaan doen?
- Wat vindt u ervan als de robot kort van te voren leunt in de richting van sturen (zoals een motorrijder of fietser) als communicatiemiddel van sturen?
- Zou dit ook werken bij versnellen of vertragen? Hoe moet de robot dan leunen?
- Welk middel of combinatie van middelen zou voor u het prettigst in gebruik zijn?
- Heeft u zelf nog een communicatiemiddel die niet genoemd is maar welke u toe zou willen voegen?

Thank the interviewee for their participation.

De gegevens die bij of door dit interview naar boven komen worden anoniem verwerkt en gebruikt. Deze anonieme gegevens kunnen direct of indirect worden gepubliceerd in de vorm van een citaat, afstudeerscriptie, presentatie en/of video.

Ik verklaar hiermee dat ik het bovenstaande gelezen en begrepen heb: Ja / Nee

Ik geef toestemming voor opnames met een voice recorder: Ja / Nee

Ik wil dat mijn opnames verwijderd worden na afloop van het afstudeerproject: Ja / Nee

Leeftijd:	0-14	15-24	25-40	41-64	65+
Geslacht:	Man		Vrouw		Anders

Pedestrian
Passing
on the right; *"You're supposed to stay on the right."*
Only left when: *"The other is very much on my right, doesn't pay attention and I cannot go further right then the other."*
Overtaking is by default in the left.
A city (shopping street) is very chaotic, then I would just zigzag to overtake, but most certainly not touch / cause a collision. I think the robot should avoid these busy streets.

Crossing
"I always try to quickly pass in front of the other, whether they come from the right or the left. I might let somebody cross in front of me (and thus I go behind them) when they are older, less mobile, a woman/man with a child e.g. if they ask for empathy."

Pedestrians do not have priority rules like cars, bicycles and other vehicles have. Those don't apply to people on foot among other people on foot.

I will also cross behind when the other is just slightly earlier than I am or if something is sticking out in front (like a baby buggy).

When you're exactly equal you engage in a short, mostly non-verbal communication with the other.

Crossing with shallow angles (heading similar), I will always try to go behind then.
Crossing with shallow angles (heading opposite), I will pass in front. Left or right doesn't matter.

Robot
"A robot should respect the people around him, he is not a person with feelings thus he should move around the people and slowdown(stop) if he cannot. The people shouldn't need to deviate for him, but he needs to deviate for the people."

He shouldn't be a nuisance / annoying thus he shouldn't:
- Drive on your toes
- Cause people to trip
- Shock people / surprise people
- Irritate (such as cause to much noise, and most definitely not honk all the time.)
- Lights are fine, especially with eyes. It's ok to have interactions.

As the robot is smaller and faster than a human, the robot should be the one to deviate from its path the most. The robot should keep to his right when driving on the sidewalks / streets, he should not 'look for the challenge'

"When it's a busy street, he should zigzag between the crowd and honk/beep. If he can only go 5 km/h (~1.4 m/s) then what is the point of having this delivery robot?
He should go 15 km/h and instead use the bicycle lanes."

I expect the robot to deviate well ahead when passing (5 meters preferred, minimally 3 meters).
I don't want him to have orange indicator lights, those aren't required on the sidewalk.

Crossing
Independent of which side the robot crosses, he must give priority to the other pedestrians.
"With a pedestrian you have a small 'negotiation' and 'feel' who should go first through eye contact and empathy. This happens really quickly. Less than a second. A robot should adapt to my behavior."

Robot communication
He should have a brake light. He should have brake lights and go as fast as a mobility scooter.

"You notice a small deviation, the change of angle gives enough information when he is weaving/passing/overtaking on the sidewalk, it doesn't require indicator lights to signal to where he is turning then."
"With big corners (taking a side-street) he should slow down a bit and check before going."
"He should adjust his speed to go with the flow of the pedestrians."
"It's ok if he follows me, just not too long (maximum 20 meters or the like) and with an appropriate following distance, thus not too closely behind (2-3 meters distance)."
"I do want to hear something, it's very annoying when it's very silent, then it can negatively surprise you like these e-bikes can."

The amplitude of the sound should adjust to its surrounding noise level.
The pitch of the sound should adjust to its speed, faster means higher pitched, slower is lower pitched.

"He is allowed to beep but definitely no talking, it's not a social robot and people should treat it as such, I don't want people stopping and starting a conversation with it. Using beeps makes it international too."
"The robot should, when taking a corner; slow down, turn, adjust/respond when situation asks for it."
"I don't want it to lean, this makes it weird and too lively, it's a robot not an animal."
"Just keep it uniform and don't add any strange movements, don't make it too lively."

As minimal communication as possible, headlights, brake lights, one sound for everything (beep beep, same tune), don't add a display.

All other communication should be through the application, not through a display on the robot.

"It should drive and behave like a pedestrian, but shouldn't start acting social."

Appearance
It should have a sleek design, very rectangular in essence but rounded off so no sharp corner/pointy edges. It should be two tone: wheels and bottom black, top a color (white, yellow, blue, red, orange etc. as long as it very noticable).

The headlights should be higher up (not at the wheel level).
Wheels shouldn't protrude from the body.

height: 70 ~ 100 cm
width: 50 ~ 60 cm
depth: 70 ~ 100 cm (like a coffee trolley or aircraft trolley)

E. SFM-MPDM Passing and Crossing in Python

Python code for calculating passing forces in SFM-MPDM

```
def passing_force(self, agents):
    # Calculate passing force which makes agent slightly alter their heading to prevent head on collisions while passing
    self.f_pass = []                    # Clear passing force list
    for a in agents:                    # Loop through all agents objects
        if a.index != self.index:   # If it isn't the agent itself
            self.to_agent = a.pos - self.pos            # Vector from self to agent
            dist = np.linalg.norm(self.to_agent)        # Distance to the agent (length of self.to_agent)
            if self.robot:
                if dist < PASSING_CONE_LENGTH_R:            # if the agent is within active passing range
                    self.to_goal = self.goal - self.pos             # vector from self to goal
                    u_to_goal = normalize(self.to_goal)             # unit vector to goal
                    ra = rotate_vector(u_to_goal, -90) * PASSING_CONE_WIDTH_R  # rotated rectangle vertex A
                    rb = u_to_goal * PASSING_CONE_LENGTH_R + ra                # rotated rectangle vertex B
                    rc = rb - 2 * ra                                           # rotated rectangle vertex C

                    # Check if the other agent is within the active passing rectangle
                    if point_in_rotated_rectangle(self.pos + ra, self.pos + rb, self.pos + rc, a.pos):
                        to_self = self.pos - a.pos  # Vector from agent towards self
                        other_angle = a.vel  # Other agent's velocity
                        if angle_difference(to_self, other_angle) < PASSING_CONE_FACING_ANGLE_MARGIN_R:
                            # looking at first agent
                            # if the other agent is 'looking' towards the first (self) agent within margins,
                            # passing is likely. offset vector angle clockwise with bias, SLIGHTLY to left and right
                            # for determining relative position

                            self.to_goal_biased_l = normalize(rotate_vector(self.to_goal, PASSING_CONE_ANGLE_BIAS_R - .5))
                            self.to_goal_biased_r = normalize(rotate_vector(self.to_goal, PASSING_CONE_ANGLE_BIAS_R + .5))

                            d_angle_to_left = angle_difference(self.to_goal_biased_l, self.to_agent)
                            d_angle_to_right = angle_difference(self.to_goal_biased_r, self.to_agent)
                            if d_angle_to_left < d_angle_to_right:
                                # agent is on the left of the bias, thus move right to avoid

                                to_right = np.array([-self.to_goal[1], self.to_goal[0]])            # 90 deg CW (right): -y, x
                                self.f_pass.append(normalize(to_right) * PASSING_CONE_STRENGTH_R)
                            else:
                                # agent is on the right of the bias, thus move left to avoid

                                to_left = np.array([self.to_goal[1], -self.to_goal[0]])             # 90 deg CCW (left): y, -x
                                self.f_pass.append(normalize(to_left) * PASSING_CONE_STRENGTH_R)
            else:
                if dist < PASSING_CONE_LENGTH:          # if the agent is within active passing range
                    self.to_goal = self.goal - self.pos             # vector from self to goal
                    u_to_goal = normalize(self.to_goal)             # unit vector to goal
                    ra = rotate_vector(u_to_goal, -90) * PASSING_CONE_WIDTH     # rotated rectangle vertex A
                    rb = u_to_goal * PASSING_CONE_LENGTH + ra                   # rotated rectangle vertex B
                    rc = rb - 2 * ra                                            # rotated rectangle vertex C

                    # Check if the other agent is within the active passing rectangle
                    if point_in_rotated_rectangle(self.pos + ra, self.pos + rb, self.pos + rc, a.pos):
                        to_self = self.pos - a.pos  # Vector from agent towards self
                        other_angle = a.vel  # Other agent's velocity
                        if angle_difference(to_self, other_angle) < PASSING_CONE_FACING_ANGLE_MARGIN:
                            # looking at first agent
                            # if the other agent is 'looking' towards the first (self) agent within margins,
                            # passing is likely. offset vector angle clockwise with bias, SLIGHTLY to left and right
                            # for determining relative position

                            self.to_goal_biased_l = normalize(rotate_vector(self.to_goal, PASSING_CONE_ANGLE_BIAS - .5))
                            self.to_goal_biased_r = normalize(rotate_vector(self.to_goal, PASSING_CONE_ANGLE_BIAS + .5))

                            d_angle_to_left = angle_difference(self.to_goal_biased_l, self.to_agent)
                            d_angle_to_right = angle_difference(self.to_goal_biased_r, self.to_agent)
                            if d_angle_to_left < d_angle_to_right:
                                # agent is on the left of the bias, thus move right to avoid

                                to_right = np.array([-self.to_goal[1], self.to_goal[0]])            # 90 deg CW (right): -y, x
                                self.f_pass.append(normalize(to_right) * PASSING_CONE_STRENGTH)
                            else:
                                # agent is on the right of the bias, thus move left to avoid

                                to_left = np.array([self.to_goal[1], -self.to_goal[0]])             # 90 deg CCW (left): y, -x
                                self.f_pass.append(normalize(to_left) * PASSING_CONE_STRENGTH)
```

Python code for calculating crossing forces in SFM-MPDM

```
def crossing_force(self, agents):
    # Calculate crossing force which makes agent slow and alter their heading to prevent sideways
    # collisions while crossing
    self.f_cross = []               # Clear the crossing force list
    for a in agents:                # Loop through all agent objects
        if a.index != self.index:           # If it isn't the agent itself
            to_agent = a.pos - self.pos             # Vector towards other agent from self
            dist = np.linalg.norm(to_agent)         # Distance towards other agent (magnitude of to_agent)
            if self.robot:
                if dist < CROSSING_CONE_LENGTH_R:  # If the other is within active crossing range
                    to_goal = self.goal - self.pos                  # Vector towards the goal
                    angle = angle_difference(to_agent, to_goal)     # Angle difference between to_agent and to_goal
                    if angle < CROSSING_CONE_DEGS_R:                    # If the other agent is within the crossing viewing cone
                        other_angle = a.vel                                     # The other's heading
                        cross_angle = angle_difference(to_goal, other_angle)        # Calculate the angle of crossing
                        if 90 - CROSSING_CONE_FACING_ANGLE_MARGIN_R < cross_angle < 90 + CROSSING_CONE_FACING_ANGLE_MARGIN_R:
                            # crossing while other is moving perpendicular within margin
                            # check if already crossed
                            # angle difference between to_agent and right offsetted by
                            # crossing_cone_deg // 2 < crossing_cone_deg //2
                            # and the other is facing right then passed.

                            # Offset to goal vector clockwise with half of half (quarter) of the crossing cone total angle.
                            to_goal_offsetted = rotate_vector(to_goal, CROSSING_CONE_DEGS_R // 2)

                            if angle_difference(to_agent, to_goal_offsetted) <= CROSSING_CONE_DEGS_R // 2:
                                # if the other agent is on the right (relatively to the to_goal vector 'line').
                                if angle_difference(other_angle, np.array([self.to_goal[1], -self.to_goal[0]])) < CROSSING_CONE_FACING_ANGLE_MARGIN_R:
                                    # If the angle difference between the own goal vector rotated 90 deg CCW and the
                                    # other agent's heading is within margins
                                    # The other agent is moving left, (and is on the right, thus not passed).
                                    # slow down and deviate (negative unit vector of own heading and negative unit
                                    # vector of the other agent's heading multiplied by respective strength factors)

                                    slowing_force = -CROSSING_CONE_SLOW_STRENGTH_R * normalize(self.state[1]) - CROSSING_CONE_SIDE_STRENGTH_R * normalize(a.state[1])
                                    self.f_cross.append(slowing_force)  # Add force to crossing force list
                            else:
                                # other agent is on the left
                                if angle_difference(other_angle, np.array([-self.to_goal[1], self.to_goal[0]])) < CROSSING_CONE_FACING_ANGLE_MARGIN_R:
                                    # The other agent is moving right, (and is on the left, thus not passed).
                                    # slow down and deviate

                                    slowing_force = -CROSSING_CONE_SLOW_STRENGTH_R * normalize(self.state[1]) - CROSSING_CONE_SIDE_STRENGTH_R * normalize(a.state[1])
                                    self.f_cross.append(slowing_force)  # Add force to crossing force list
            else:
                if dist < CROSSING_CONE_LENGTH:  # If the other is within active crossing range
                    to_goal = self.goal - self.pos                  # Vector towards the goal
                    angle = angle_difference(to_agent, to_goal)     # Angle difference between to_agent and to_goal
                    if angle < CROSSING_CONE_DEGS:                  # If the other agent is within the crossing viewing cone
                        other_angle = a.vel                                     # The other's heading
                        cross_angle = angle_difference(to_goal, other_angle)        # Calculate the angle of crossing
                        if 90 - CROSSING_CONE_FACING_ANGLE_MARGIN < cross_angle < 90 + CROSSING_CONE_FACING_ANGLE_MARGIN:
                            # crossing while other is moving perpendicular within margin
                            # check if already crossed
                            # angle difference between to_agent and right offsetted by
                            # crossing_cone_deg // 2 < crossing_cone_deg //2
                            # and the other is facing right then passed.

                            # Offset to goal vector clockwise with half of half (quarter) of the crossing cone total angle.
                            to_goal_offsetted = rotate_vector(to_goal, CROSSING_CONE_DEGS // 2)

                            if angle_difference(to_agent, to_goal_offsetted) <= CROSSING_CONE_DEGS // 2:
                                # if the other agent is on the right (relatively to the to_goal vector 'line').
                                if angle_difference(other_angle, np.array([self.to_goal[1], -self.to_goal[0]])) < CROSSING_CONE_FACING_ANGLE_MARGIN:
                                    # If the angle difference between the own goal vector rotated 90 deg CCW and the
                                    # other agent's heading is within margins
                                    # The other agent is moving left, (and is on the right, thus not passed).
                                    # slow down and deviate (negative unit vector of own heading and negative unit
                                    # vector of the other agent's heading multiplied by respective strength factors)

                                    slowing_force = -CROSSING_CONE_SLOW_STRENGTH * normalize(self.state[1]) - CROSSING_CONE_SIDE_STRENGTH * normalize(a.state[1])
                                    self.f_cross.append(slowing_force)  # Add force to crossing force list
                            else:
                                # other agent is on the left
                                if angle_difference(other_angle, np.array([-self.to_goal[1], self.to_goal[0]])) < CROSSING_CONE_FACING_ANGLE_MARGIN:
                                    # The other agent is moving right, (and is on the left, thus not passed).
                                    # slow down and deviate

                                    slowing_force = -CROSSING_CONE_SLOW_STRENGTH * normalize(self.state[1]) - CROSSING_CONE_SIDE_STRENGTH * normalize(a.state[1])
                                    self.f_cross.append(slowing_force)  # Add force to crossing force list
```

F. User test setup of Python SFM-MPDM

Goal of the user test
During the Research and Exploration phase a python simulation model has been built based on the Social Force Model with Multi-Policy Decision Making. Some additions have been made to this model after identification of possible shortcomings. These additions are: longer distance/earlier communication in passing, improved behavior on short distance crossing and the prevention of the 'instant 180 degree turn' by introducing a rotational velocity for the agents. The model is supposed to simulate pedestrian agents navigating amongst each other in an environment representing a small square with connected alleys. A robot agent is also present in the environment and tries to move through this pedestrian rich environment with socially acceptable behavior. The goal of the user test is to evaluate the model on any shortcomings in pedestrian behavior and to evaluate the behavior of the robot by ranking the shown behavior on intuitiveness, comfortableness and predictability, as indicated by the participants. The insights resulting from the user test are to be used as an input for adjustments to the model before transfer to the ROS development environment, which in turn is to be implemented in a physical robot that can be tested in real world environments.

Research questions
What shortcomings are present in the pedestrian simulation model?
How human like is the behavior of the robot?
How comfortable is the behavior of the robot?
How predictable is the behavior of the robot?

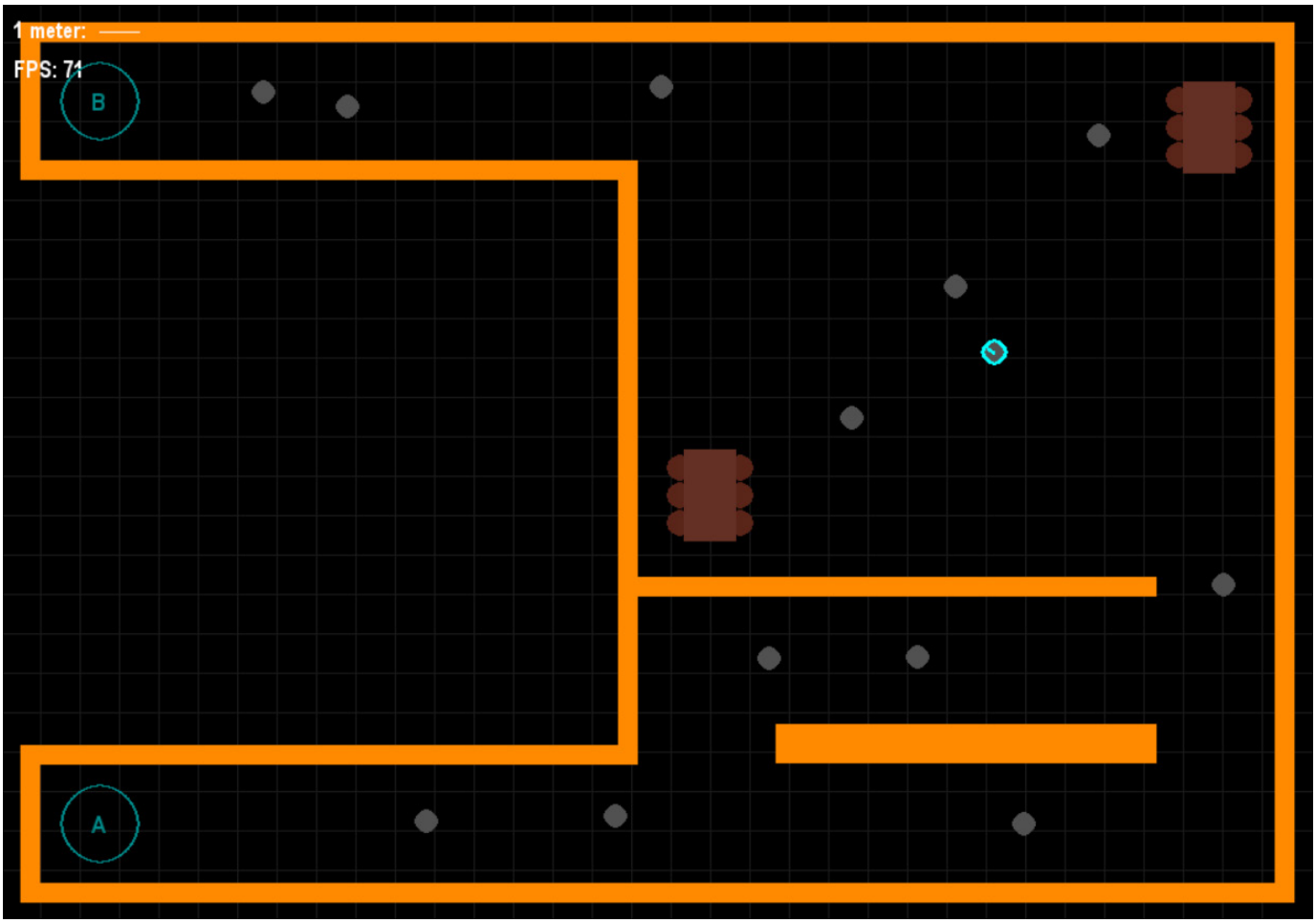
Test setup
There are two scenarios which are to be tested:

1. Scenario with just pedestrians walking about in the environment.
2. Scenario with both pedestrians and the robot in the environment.

The first is to evaluate the quality / shortcomings in the behavior of the simulated pedestrians. The second is to evaluate the behavioral performance of the robot on intuitiveness, comfortableness and predictability.

All participants will be shown both scenarios, but half of the participants will be shown scenario 1 first, the other half will be shown scenario 2 first.

The environment in which the scenarios are tested looks as such:



This environment includes a large open space, several corridors and a decision on two paths for the same destination. A note here, crossing will most likely not happen, but can be included in the large open space.

The force vectors acting upon the pedestrians and robots will not be shown. All pedestrians will be of the same color; light gray.

The goal of the robot and the active policy will not be shown, the robot will be light gray with a cyan ring to distinguish it from pedestrians.

The participants will be asked to think out loud while watching both scenarios and to comment on anything they might find noteworthy.

After the questions have been answered I will go through the questions with the participant to further detail the reasons for their answers.

Questions
Please think out loud while watching the simulation and comment on anything you find relevant.

G. User test form of Python SFM-MPDM

Scenario A
Please state to what extent you agree with the following statement:
The behavior of the pedestrians is similar to what I expect from real people.

Strongly disagree | Disagree | Undecided | Agree | Strongly agree

Please explain your answer below.
……
……
……

Scenario B
Please state to what extent you agree with the following statement:
*The behavior of the robot seems **human like** to me.*

Note: think about how it deals with passing, crossing and overtaking and compare this with the way humans would act.

Strongly disagree | Disagree | Undecided | Agree | Strongly agree

Please explain your answer below.
……
……
……

Please state to what extent you agree with the following statement:
*The behavior of the robot feels **comfortable** to me.*

Note: think about distance, speed, and the way it follows or drives alongside pedestrians.

Strongly disagree | Disagree | Undecided | Agree | Strongly agree

Please explain your answer below.
……
……
……

Please state to what extent you agree with the following statement:
*The behavior of the robot feels **predictable** to me.*

Note: can you predict what the robot will do in the immediate future?

Strongly disagree | Disagree | Undecided | Agree | Strongly agree

Please explain your answer below.
……
……
……

Were there moments in the simulation that surprised you?

Yes | No

If applicable; Please describe the moment(s) as clearly as possible below.
……
……
……

H. User test results of Python SFM-MPDM

Scenario A

The behavior of the pedestrians is similar to what I expect from real people.

1. (Agree) The dots seem to follow the social 'rules' of navigating, hold back when necessary and pass each other in acceptable ways.
2. (Agree) I would not walk so straight in the middle as the dots do. Take a longer distance to others in some situations.
3. (Agree) I would change my direction a bit earlier to not get too close to anyone. Also I would probably stay more to the right side.
4. (Agree) People would avoid each other sooner and walk in groups more often.
5. (Disagree) I expect more groups of walking pedestrians such as 2 or 3 walking side by side. I expect people to stand still sometimes.
6. (Undecided) People cut corners. The will anticipate more when they see someone coming.
7. (Disagree) In real life people would cut corners more. The 'dance' if you're facing each other head on instead of passing each other so easily.
8. (Agree) The think the 'randomness' is good, but the predictive capabilities of the people in the simulation can be improved, with regards to the predicting / anticipating how other people would walk.
9. (Agree) Passing goes well in most cases, people don't bump into each other. Chosen 'routes' look random which matches reality. Do take pedestrian density into account.
10. (Agree) In the middle of the room it seems natural, but at the turning points it becomes weird (at the dead ends people are walking while turning around instead of stopping and turning). I found it difficult to imagine the dots as people.
11. (Undecided) Human: the people don't touch each other and walk in different directions which is human. Not human: The people do not stop (for example to talk with one another).
12. (Agree) There should also be people who are talking and standing still in the scenario. The difference between fast and slow people wasn't that clear.
13. (Agree) Sometimes I would observe in advance and turn a bit earlier than what the people do on the screen.

Scenario B

The behavior of the robot seems human like to me.

1. (Undecided) In general the robot moves human like, but there are some minor things noticeable in giving priority in a socially acceptable way. (Does not hold back sometimes).
2. (Agree) Feels like a bit too close to the humans
3. (Agree) Soft turns, waits before corner. But feels like people move away from the robot more than the robot moves away from people.
4. (Agree) The behavior is comparable to the behaviors of the humans in this scenario, too much so. A robot should have a lower priority in overtaking and shouldn't be in the way of humans.
5. (Strongly agree) As far as the model is concerned the robot acts like the humans.

6. (Agree) The robot maneuvers through the people naturally. It is good that he stops sometimes to let people pass.
7. (Undecided) It avoid people and finds the shortest path.
8. (Undecided) People look ahead more, but this robot seems to make the decision at the last moment.
9. (Agree) Robot can sense accurately when someone is in his routes but might be better if he keeps a bit more distance from the people near him. Also think about the behavior of the robot when it is being overtaken.
10. (Disagree, positive in some areas) Nice how it lets people first. But it can follow behind someone for too long -> weird / uncomfortable. The robot needs 'knowledge' of how uncomfortable following feels. Too long behind someone -> slow down or pass.
11. (Agree) If it's from the supermarket towards a home; yes very human. The robot moves out of the way of people and politely waits. It also goes straight for its target.
12. (Strongly agree) He stops at places where required and 'walks' just about the same speed as others do. In busy places he moves like the others do.
13. (Agree) It looks very similar to the human beings in the simulation, except for appearance.

The behavior of the robot feels comfortable to me.

1. (Agree) It has a speed comparable to the humans and keeps enough distance while passing.
2. (Agree) Feels like it needs longer distance, not following me. Speeds seems good as long as a human walks faster if it follows you.
3. (Undecided) I think the robot should be the one to take responsibility of not hitting anyone. And not walk straight behind anyone.
4. (Disagree) See answer above.
5. (Disagree) I think the robot should not be treated as a equal to humans and should give way. Not force humans to alter or slow their movement.
6. (Strongly agree) It is respecting the people in his way and stops when he needs to. He goes with the stream.
7. (Agree) Sufficient distance, same speed as the humans, stops at critical moments. Deviates from the route to let people pass.
8. (Disagree) The space between people, speeds and decisions at the last moment can come across as undesired.
9. (Agree) It knows its route well, the speed is good and adjusts itself to the situation. Do think about a minimum distance between the robot and the nearby people.
10. (Strongly agree) The robot moves out of the way for the people
11. (Agree) He waits politely, doesn't do anything unexpected. But a bit more space between him and the humans would be more comfortable.
12. (Strongly agree) He blends in with the others speed wise, which is good. He also lets others pass.
13. (Undecided) I would prefer it to stop for a while and let me go first.

The behavior of the robot feels predictable to me.

1. (Agree) It holds its line and doesn't make sudden turns where not expected. As I told earlier it does sometimes not hold in where a human possibly would.

2. (Undecided) It feels safe but a bit unpredictable.
3. (Disagree) I think it's difficult to predict behaviors both for people and the robot. Maybe short term but not for the future.
4. (Strongly agree) Just as predictable as humans.
5. (Agree) The pathing is relatively simple, where I know all entities continue straight until they encounter something in the way that is close.
6. (Agree) It has a clear route and only deviates if someone is close.
7. (Agree) Follows a straight line as much as possible, only deviates when there is an obstacle, nearly no collisions.
8. (Undecided) The ideal line is predictable, but the way it passes someone is not.
9. (Agree) On basis of the simulation I can predict that the robot will deviate when a person is nearby.
10. (Strongly agree) Doesn't turn harshly or randomly. So would easily avoid him.
11. (Strongly agree) Always waits in the same way, usually the same distance between robot and human, except when it's narrow due to walls.
12. (Agree) Sometimes you see the robot doubt a bit (which also happens with humans in reality) but this does not make it predictable. But I think that that doesn't matter.
13. (Undecided) I am not really sure about its routine in a bigger picture.

Were there moments in the simulation that surprised you?

1. No
2. Yes; Robot went through a human once.
3. Yes; People walking through the wall, Robot hitting a person.
4. No (but actually yes); Sometimes when a human and the robot meet, the robot takes the priority, which you wouldn't expect from a robot.
5. No
6. No
7. No
8. Yes; When the robot nearly pushed someone into a wall.
9. No
10. Yes; people got locked out of the simulation (went through the walls)
11. Yes; It wasn't completely clear what kind of scenario I was looking at. Also, people sometimes went through walls.
12. Yes; But in a good way, because he stopped for others.
13. Yes; I expect a fixed routine for the robot.

Common themes

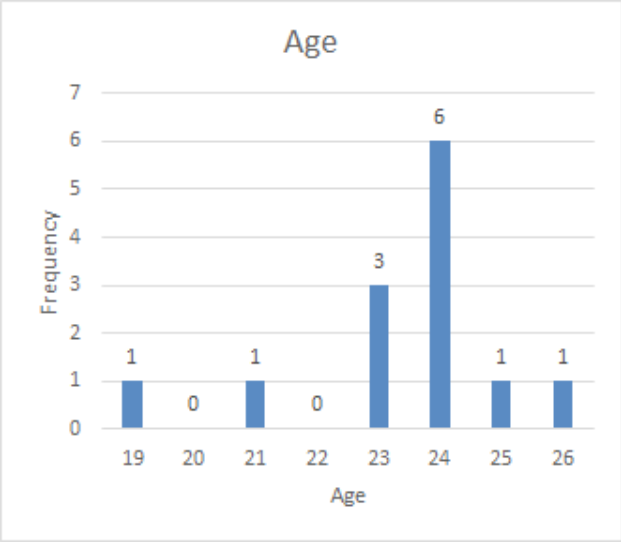
Scenario A (just people):

- **People would anticipate earlier, the reaction is late (crossing and passing)**
- **People would sometimes just stop or slow down.**
- People can walk in groups which they don't in this case.
- People went through walls
- Stay to the right a bit more
- People would cut corners (bottom center in environment)
- People are quite random, which is realistic
- People don't collide, which is good
- I sometimes find it difficult to imagine the dots as people.

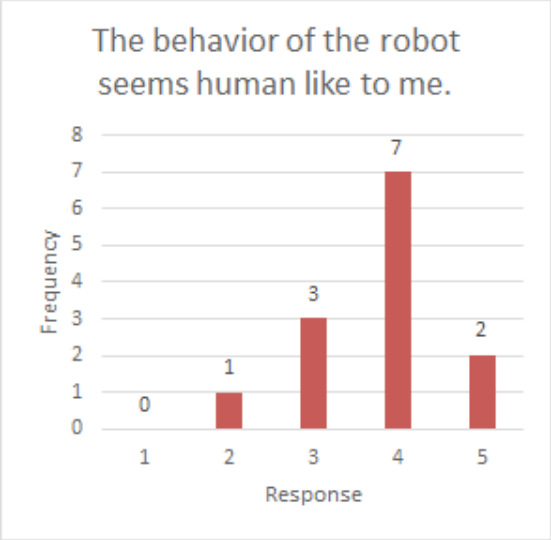
Scenario B (with robot):

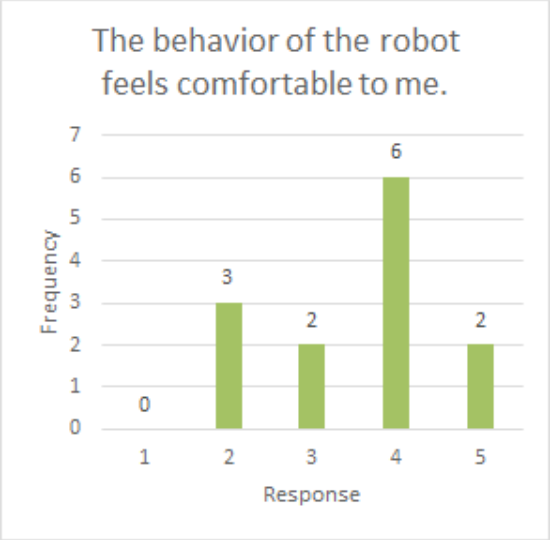
- **More distance between people and robot.**
- **Robot sometimes takes priority which you wouldn't expect or want, robot should be better behaved.**
- Is it good that the robot sometimes stops to let people pass or cross (especially near the right most corner after the open space).
- Robot follows straight lines when it can, this makes it more predictable; it doesn't turn suddenly or randomly.
- **The robot can follow people for too long / too closely, this isn't comfortable.**
- The speed is good compared to the other people (similar speed).
- The robot (and people) stay quite far from the walls.
- The robot doesn't necessarily shows human behavior, but this can be good as humans can sometimes be egoistic or uncomfortable, less human makes it more predictable.

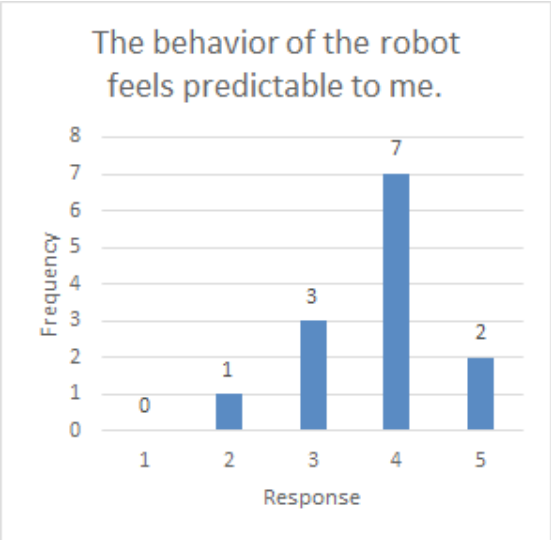
	Demographic information		**Scenario A**	**Scenario B**				
Participant number	*Male / Female*	*Age*	*The behavior of the pedestrians is similar to what I expect from real people.*	*The behavior of the robot seems human like to me.*	*The behavior of the robot feels comfortable to me.*	*The behavior of the robot feels predictable to me.*	*Where there moments that surprised you?*	
1	M	24	4	3	4	4	N	
2	F	24	4	4	4	3	Y	
3	F	24	4	4	3	2	Y	
4	F	23	4	4	2	5	N	
5	M	24	2	5	2	4	N	
6	F	26	3	4	5	4	N	
7	F	25	2	3	4	4	N	
8	M	21	4	3	2	3	Y	
9	F	23	4	4	4	4	N	
10	F	23	4	2	4	4	Y	
11	F	24	3	4	4	5	Y	
12	F	19	4	5	5	4	Y	
13	F	24	4	4	3	3	Y	
Median	-	24	4	4	4	4	-	
Average	-	23.38	3.54	3.77	3.54	3.77	-	
M	23.1%						46.2%	N
F	76.9%						53.8%	Y


Age
Frequency
7
6
5
4
3
2
1
0
1
0
1
0
3
6
1
1
19
20
21
22
23
24
25
26
Age


The behavior of the pedestrians is similar to what I expect from real people.
Frequency
10
8
6
4
2
0
0
2
2
9
0
1
2
3
4
5
Response


The behavior of the robot seems human like to me.
Frequency
8
7
6
5
4
3
2
1
0
0
1
3
7
2
1
2
3
4
5
Response


The behavior of the robot feels comfortable to me.
Frequency
7
6
5
4
3
2
1
0
0
3
2
6
2
1
2
3
4
5
Response


The behavior of the robot feels predictable to me.
Frequency
8
7
6
5
4
3
2
1
0
0
1
3
7
2
1
2
3
4
5
Response

I. Evolutionary Algorithm parameters and their range and impact

	V_MAX_R	K_GOAL_R	A_PED_R	B_PED_R	A_OBS_R	B_OBS_R	L_R	HORIZON	TIME_STEP	POLICY_ELECTION_CYCLE	ALPHA	MIN_FOLLOW_DISTANCE	PASS_LENGTH_R	PASS_WIDTH_R	PASS_STRENGTH_R	PASS_ANGLE_BIAS_R	PASS_FACING_ANGLE_MARGIN_R	CROSS_LENGTH_R	CROSS_DEGS_R	CROSS_SLOW_STRENGTH_R	CROSS_SIDE_STRENGTH_R	CROSS_FACING_ANGLE_MARGIN_R
Social	-	+	+	+	+	+	+	+	-	+	-	+	+	+	+	+	+	+	+	+	+	+
Technology	-	+	-	-	-	-	+	-	+	-	+	-	-	-	-	-	-	-	-	-	-	-
(Product) Service	+	+	-	-	-	-	+	-	+	-	+	-	-	-	-	-	-	-	-	-	-	-

Note: The + and - symbols represent the expected impact of 'increasing the parameter' on the Social, Technology or (Product) Service aspect. Note that increasing values beyond sensible ranges will negate any positive effects due to extreme behavior created by the parameter. *Most parameters will have a 'sweet spot' for a certain aspect. Any value higher / lower than that sweet spot will negatively impact that aspect relatively to the impact of the sweet spot.*

Parameter description		Default	Unit	Min	Max
V_MAX_R	Maximum velocity of the robot	1.3	m/s	0.5	2
K_GOAL_R	Goal force strength factor	0.4	-	0	4
A_PED_R	Pedestrian force strength factor	0.3	-	0	4
B_PED_R	Pedestrian force range factor	0.8	-	0	4
A_OBS_R	Obstacle force strength factor	0.2	-	0	4
B_OBS_R	Obstacle force range factor	1	-	0	4
L_R	Lambda; strength of the anisotropic factor	0.1	-	0	1
HORIZON	A forward simulation horizon of default 1 seconds	1	s	0	3
TIME_STEP	Amount of seconds during each iteration in the horizon	0.1	s	0.1	0.5
POLICY_ELECTION_CYCLE	Select new policy every 3 TIME_STEPS (default every 0.3 simulated seconds)	3	-	1	10
ALPHA	The 'progress' weight factor for the MPDM cost function	0.2	-	0	0.5
MIN_FOLLOW_DISTANCE	Minimal distance to leader during follow policy	1.5	m	0	3
PASS_LENGTH_R	Length of the sensing area for frontal passing	3	m	0	5
PASS_WIDTH_R	Width of the sensing area for frontal passing	0.6	m	0	2
PASS_STRENGTH_R	Passing force strength factor	0.3	-	0	1
PASS_ANGLE_BIAS_R	Right hand rule bias in passing (increase to keep the right even though on more the left)	3	deg	0	10
PASS_FACING_ANGLE_MARGIN_R	Angle the other agent's 'look' can deviate from directly 'looking' at the first agent.	30	deg	0	50
CROSS_LENGTH_R	Length of the sensing range for the crossing cone	2	m	0	4
CROSS_DEGS_R	Active viewing area within DEG of current viewing angle, both ways so 2 * DEG total	40	deg	0	80
CROSS_SLOW_STRENGTH_R	Speed reduction force factor per crossing pedestrians	0.1	-	0	1
CROSS_SIDE_STRENGTH_R	Side stepping force factor per crossing pedestrian	0.3	-	0	1
CROSS_FACING_ANGLE_MARGIN_R	Angle the other agent's 'look' can deviate from directly perpendicular to the own heading	40	deg	0	60

Note: The above table shows all the chromosome parameters in the evolutionary algorithm. It also lists their description, default value, unit and (mutation) range. The color grouping is based on the way the parameters are handled in the computational model. Parameters in the same colour group are often used together.

J. Baseline for the fitness function weights

	SLOW SPEED						
	Soc	Soc	Ser	Tec + Ser	Ser	Ser	Tec
	Social Forces	**Collision num**	**Runtime**	**Path length**	**Distance to goal**	**Goal reached**	**Stops count**
	236.301	22	33.18	10.54	0	1	10
	609.284	104	48.78	17.71	0	1	21
	232.955	0	30.46	16.92	0	1	10
	631.011	94	55.1	18.16	0.537	0	22
	291.654	66	30.94	14.01	0	1	10
	282.602	44	41.64	16.53	0	1	19
	489.804	68	55.09	16.42	2.456	0	22
	114.388	11	24.01	12.06	0	1	6
	260.414	0	44.81	17.64	0	1	16
	405.124	13	47.68	15.1	0	1	14
Average	355.3537	42.2	41.169	15.509	0.2993	0.8	15

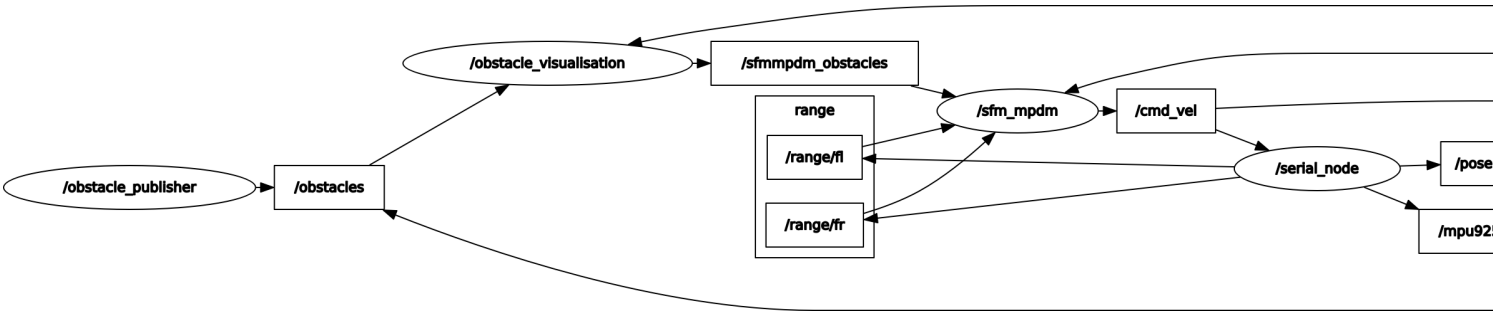
	Social Forces	**Collision num**	**Runtime**	**Path length**	**Distance to goal**	
	Social	*Social*	*Service*	*Tech + Serv*	*Service*	
Values for weights	**330**	**34**	**40**	**15.6**	**3**	
Social	500	500				
Service			250	250	250	
Technology				500		
Importance factor (Social, Service or Technology, 0,333 = neutral)	0.33333	0.33333	0.33333	0.33333	0.33333	
WEIGHTS:	**1.515151515**	**14.70588235**	**6.25**	**48.0769231**	**83.33333333**	
Social total	333.33					
Service total	333.33					
Technology total	333.33					
Fitness function	166.665	166.665	83.3325	249.9975	83.3325	
	-833.325					

	STD SPEED						
	Soc	Soc	Ser	Tec + Ser	Ser	Ser	Tec
	Social Forces	**Collision num**	**Runtime**	**Path length**	**Distance to goal**	**Goal reached**	**Stops count**
	136.454	0	27.67	10.93	0	1	7
	360.309	34	33.57	15.21	0	1	11
	248.714	9	51.87	21.08	0	1	24
	229.541	75	24.05	14.53	0	1	6
	332.421	84	53.85	18.19	0	1	18
	204.11	0	29.05	14.8	0	1	8
	604.212	14	55.1	14.32	3.09	0	18
	312.841	20	33.85	14.24	0	1	13
	207.671	4	30.73	15.09	0	1	10
	433.814	15	47.71	18.54	0	1	13
Average	307.0087	25.5	38.745	15.693	0.309	0.9	12.8
Avg of avg:	331.1812	33.85	39.957	15.601	0.30415	0.85	13.9

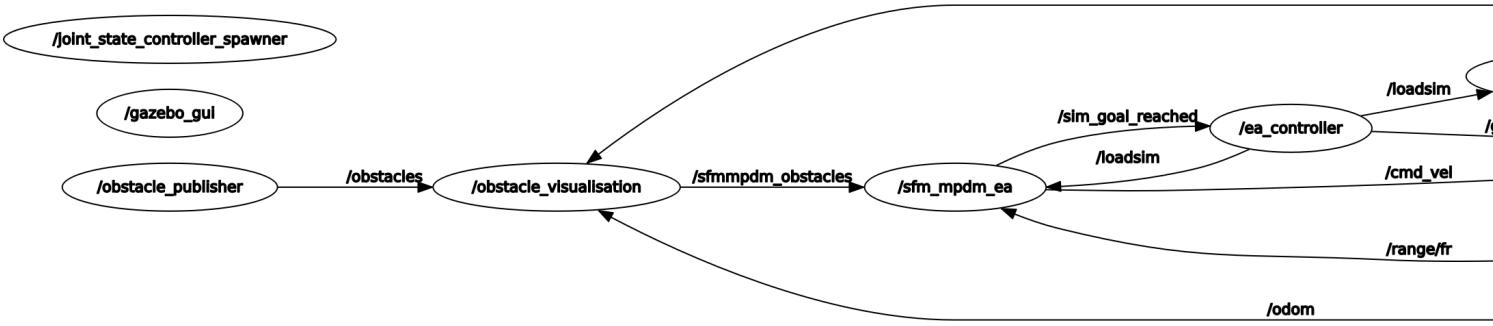
Goal reached	**Stops count**					
Service	*Technology*					
1	**14**			**Importance**		
		1000		0.33333		
250		1000		0.33333		
	500	1000		0.33333		
0.33333	0.33333			0.99999		
250	**35.7142857**					
		< If the chromosome scores average on all scores.				
		And each aspect from the triangle is weighted neutral (0.3333) then				
		each aspects summed score in the fitness function equals a 333.33				
83.3325	166.665					
		< The average chromosome will then have a fitness value of -833.25				

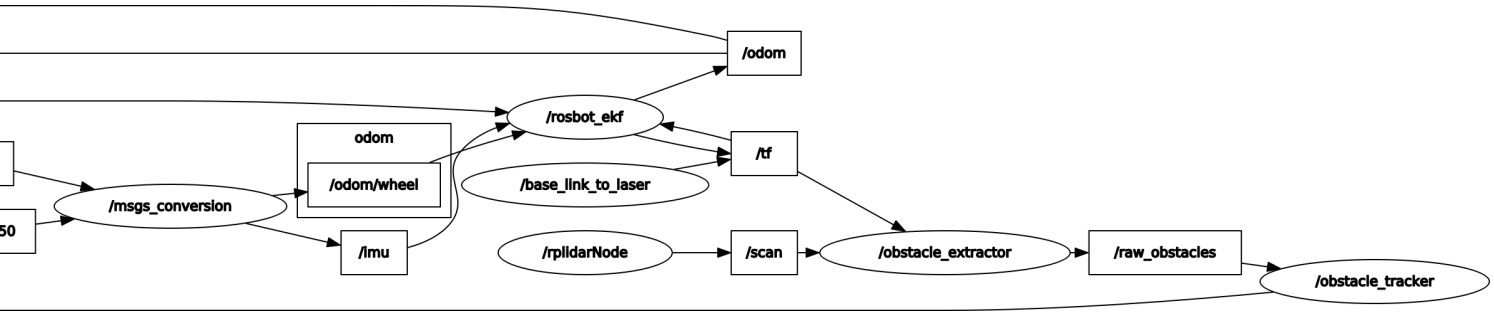
K. RQT Graphs of the ROS node structures

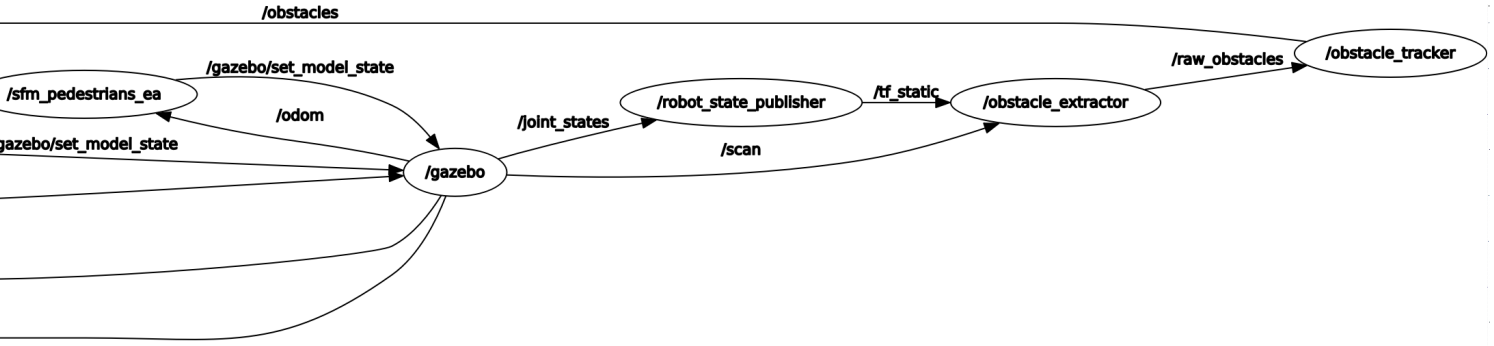
RQT Graph of the node structure for the driving of the physical ROSbot 2.0 Pro.



RQT Graph of the node structure for the parameter learning with the evolutionary algorithm.




/odom
/rosbot_ekf
odom
/odom/wheel
/tf
/base_link_to_laser
/msgs_conversion
/imu
/rplidarNode
/scan
/obstacle_extractor
/raw_obstacles
/obstacle_tracker


/obstacles
/obstacle_tracker
/raw_obstacles
/gazebo/set_model_state
/sfm_pedestrians_ea
/odom
/robot_state_publisher
/tf_static
/obstacle_extractor
/joint_states
gazebo/set_model_state
/gazebo
/scan

L. HREC informed consent form

Information Sheet for 'ROSbot behavior test'
Date: 04/02/2020

The goal of the study is to evaluate the behavioral performance of an autonomous guided vehicle (AGV) called the ROSbot 2.0 Pro.

In this study, the computational model controlling the AGV has been developed for autonomous driving in pedestrian rich environments. The user test involves performing multiple runs with other participants and the AGV inside the closed off test environment. The AGV will be driving based on its perception of its surroundings. After each run, you are asked to fill in a short digital form reflecting your experiences and opinions of that run.

Any information provided will be handled anonymously, the digital form requires no name, age or gender. Any video-recordings will be anonymised post-study through facial blur in the case of a publication.

Gathered data will be part of the graduation work; the master thesis and public presentation, and thus will be deposited, after anonymisation, in the TU Delft Education repository.

Consent Form for 'ROSbot behavior test'

Please tick the appropriate boxes	Yes	No
Taking part in the study		
I have read and understood the study information dated 04/02/2020, or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves filling out a digital form anonymously as part of each run. I also understand that taking part in the study involves the use of video-recordings which will be anonymised in the case of a publication.	○	○
Risks associated with participating in the study		
I understand that taking part in the study involves the following risks: Possible collision of my feet with a small (20 x 24 x 22 cm) and lightweight (2.84 kg) autonomous guided vehicle; the ROSbot 2.0 Pro.	○	○
Use of the information in the study		
I understand that information I provide will be used for the purpose of a graduation project including the graduation thesis, public presentation and a possible publication.	○	○
I understand that personal information collected about me that can identify me, such as the visual materials recorded, will not be shared beyond the study team.	○	○
I agree that my information can be quoted in research outputs.	○	○
Future use and reuse of the information by others		
I give permission for the anonymous responses in the digital form and the anonymised video material that I provide to be archived in the TU Delft Education Repository so it can be used for future research and learning.	○	○

Signatures

_____	_____	_____
Name of participant	Signature	Date

I have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

Patrick Keesmaat	_____	_____
Researcher name	Signature	Date

Study contact details for further information:

Name: Patrick Keesmaat, Dr. Zoltán Rusák
E-mail: P.Keesmaat@student.tudelft.nl, Z.Rusak@tudelft.nl

M. User test digital form

ROSbot behavior test

*Vereist

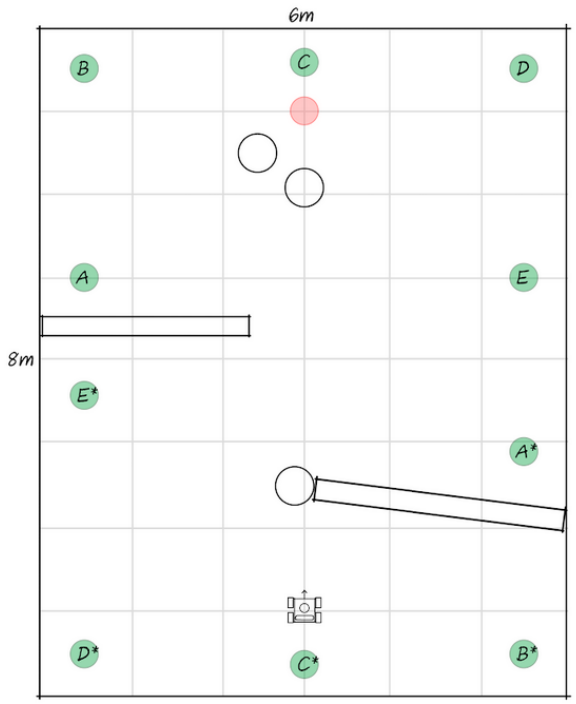
Number of the run you just completed: *

Jouw antwoord

Your assigned number / color: *

Kiezen

Looking at the following floor-plan, please mark your starting position. *



Kiezen

Please state to what extent you agree with the following statement: "The behavior of the robot feels comfortable to me." Note: think about distance, speed, and the way it follows or drives alongside pedestrians. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Please state to what extent you agree with the following statement: "The behavior of the robot feels predictable to me." Note: can you predict what the robot will do in the immediate future? *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Please state to what extent you agree with the following statement: "It was clear to me what the robot's intentions were." Note: could you tell which way it was going to drive, did you feel like the robot noticed you? *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Please state to what extent you agree with the following statement: "My run in the test environment was positively influenced by the robot's behavior." Note: were you blocked by the robot, did the robot try to avoid you, did the robot collide with you? *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

If this is your final run, please take a short moment to fill in any improvements and/or suggestions you have for the robot and its behavior.

Jouw antwoord

N. Results of the Evolutionary Algorithm runs

Run 1. 'normal street' situation.

Evolutionary Algorithm parameters

Weight / parameter name	value
social_importance	0.333333
service_importance	0.333333
technology_importance	0.333333
w_social_forces	1.5151515
w_collision_num	14.7058824
w_runtime	6.25
w_path_length	48.0769231
w_distance_to_goal	83.3333333
w_goal_reached	250
w_stops_count	35.7142857
generation_size	30
population_size	60
max_mutation_probability	0.05
start_convergence_pressure	0.99
end_convergence_pressure	0.9
run_maxtime	55

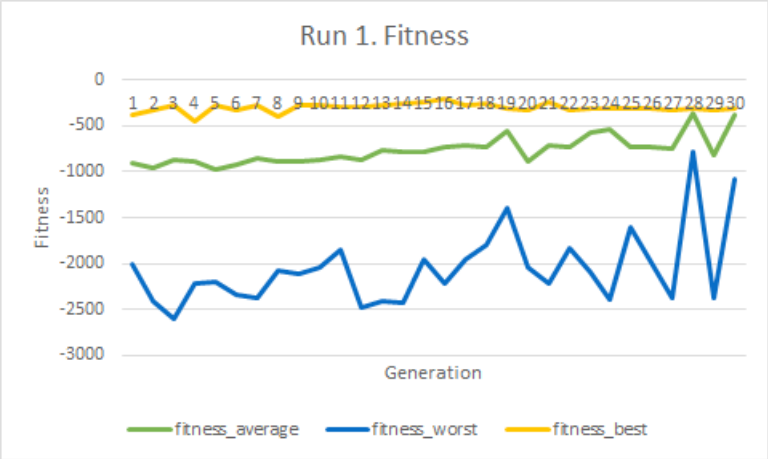
Learned SFM-MPDM parameters

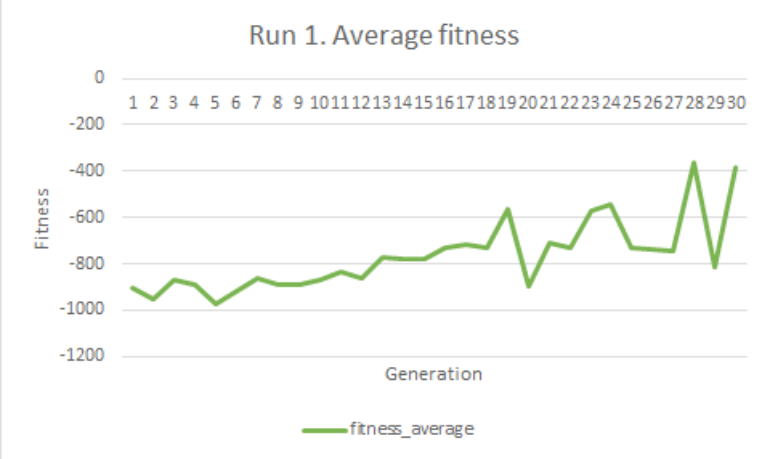
Parameter name	#1	#2	#3
fitness	-319.85847	-325.96620	-331.66434
v_max	0.81439	0.81439	0.81439
k_goal	0.40858	0.40858	0.40858
a_ped	3.81392	3.78187	3.81392
b_ped	1.38193	3.47652	3.47652
a_obs	1.56175	3.16356	3.16356
b_obs	3.81488	3.81488	3.81488
lambda	0.52259	0.12389	0.12389
horizon	0.49681	1.40686	0.49681
time_step	0.49561	0.49561	0.49561
policy_election_cycle	4.71916	5.28918	5.28918
alpha	0.35185	0.4351	0.35185
min_follow_distance	0.45683	0.45683	0.45683
pass_length	3.33588	3.33588	3.33588
pass_width	1.00089	1.20267	1.20267
pass_strength	0.51287	0.91384	0.38683
pass_angle_bias	1.84987	3.51409	3.51409
pass_facing_angle_margin	40.35922	40.35922	44.62168
cross_length	1.15957	1.29794	1.15957
cross_degs	19.24577	19.24577	19.24577
cross_slow_strength	0.88401	0.88401	0.88401
cross_side_strength	0.98041	0.98041	0.98041
cross_facing_angle_margin	59.63725	59.63725	59.63725

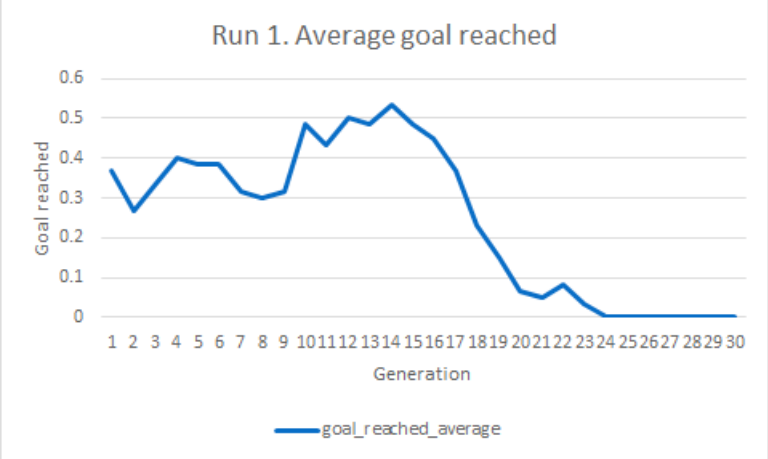
Notes

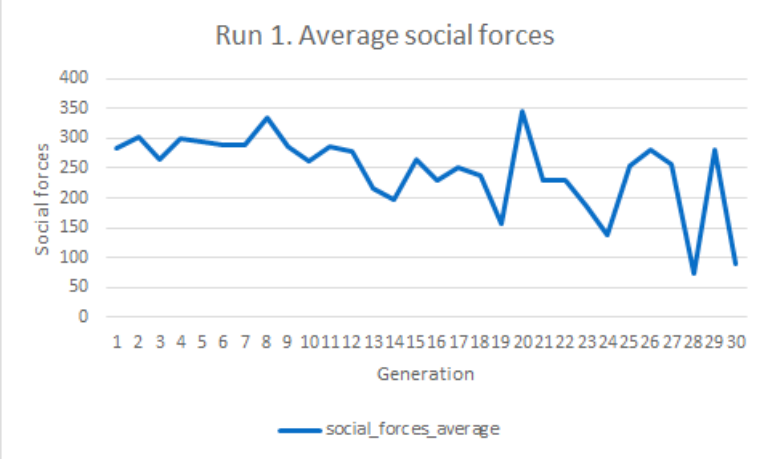
Behavior
Did not learn to reach the goal. Instead it learnt to avoid people and drive as little as possible to lower social forces, collision count, path length and stops.

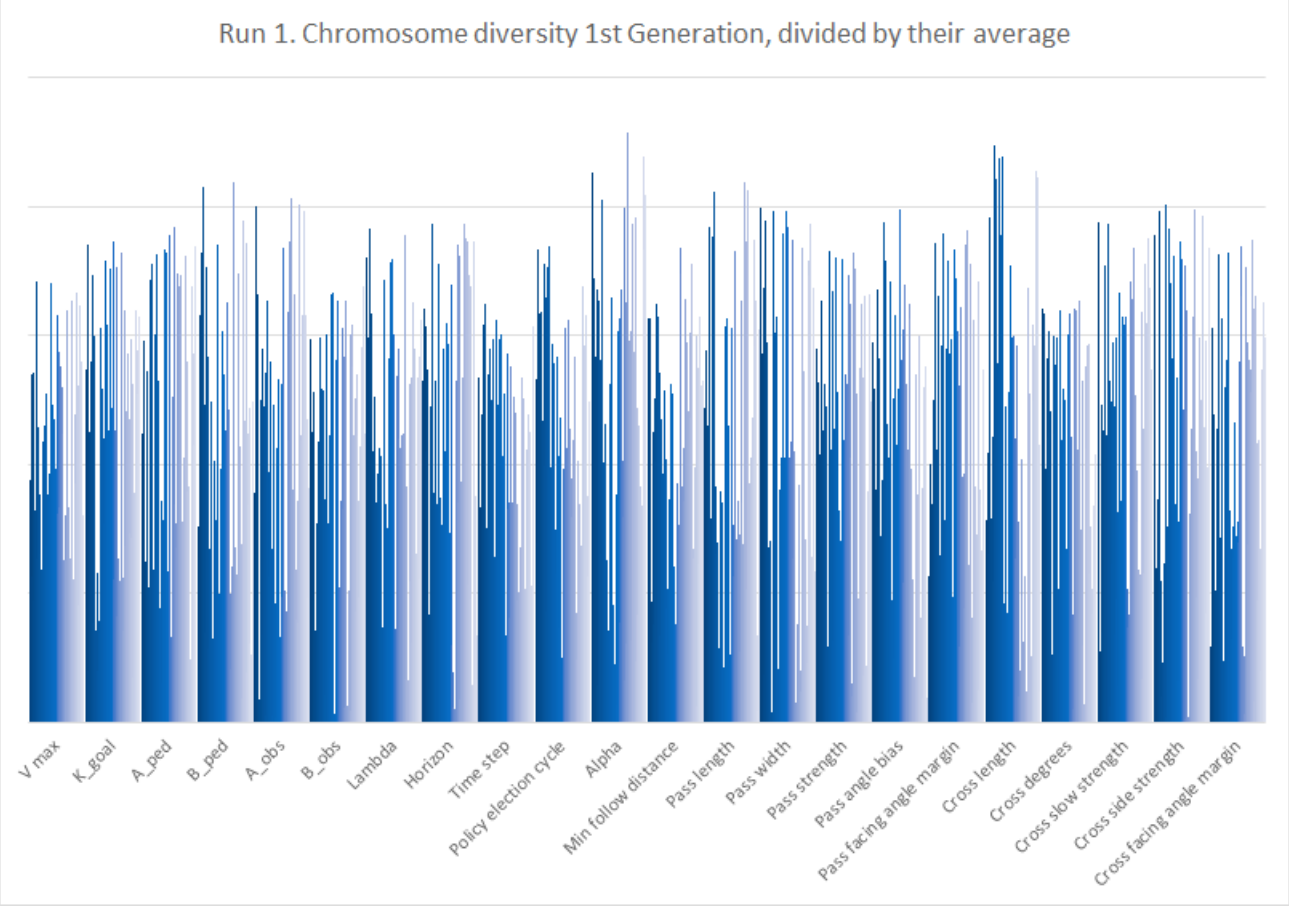
Adjustment for next time.
Increase w_goal_reached from 250 to 1000 to compensate/counteract the other scores combined effect of making the reaching of the goal undesirable.

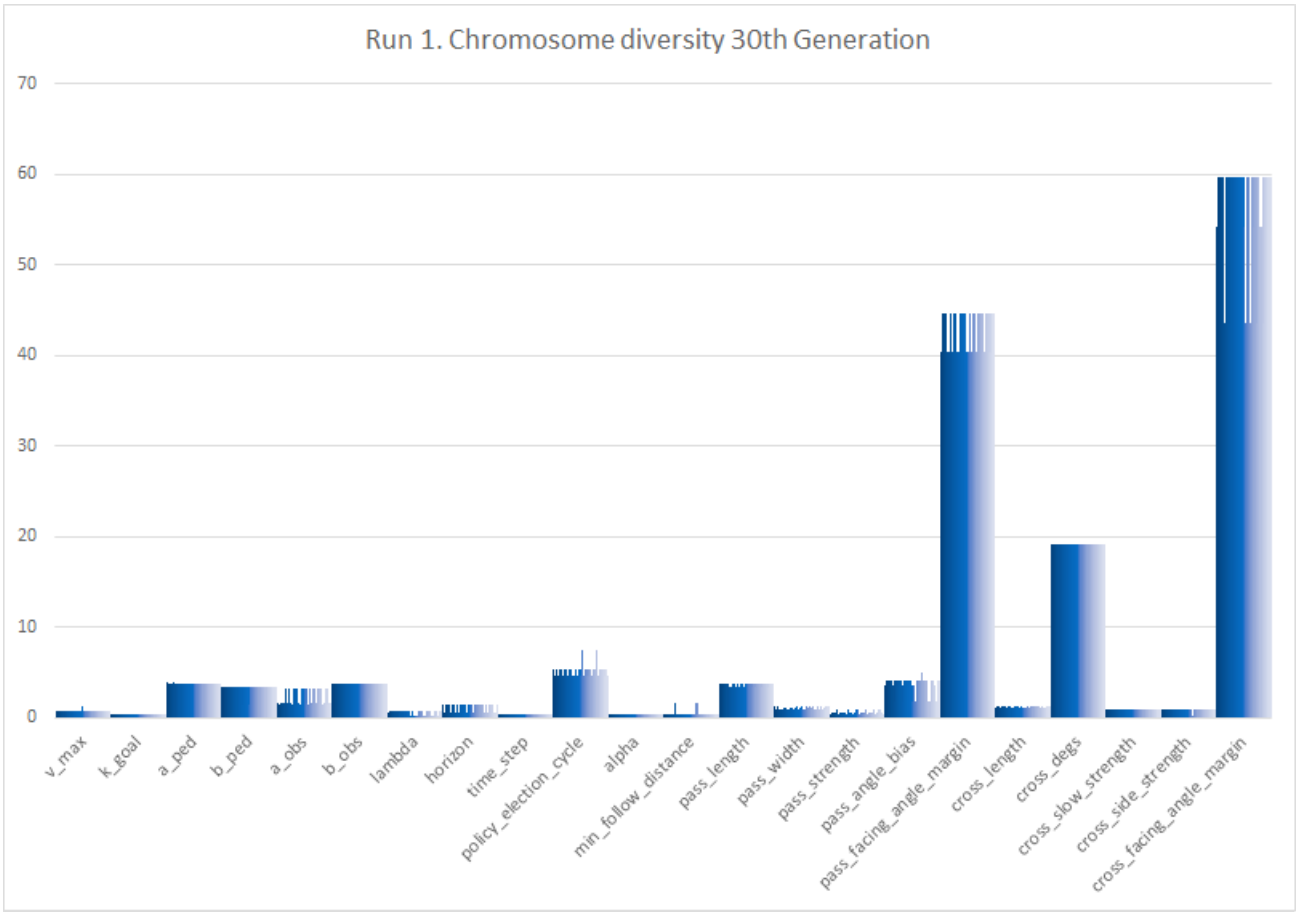

Run 1. Fitness
Fitness
Generation
fitness_average
fitness_worst
fitness_best

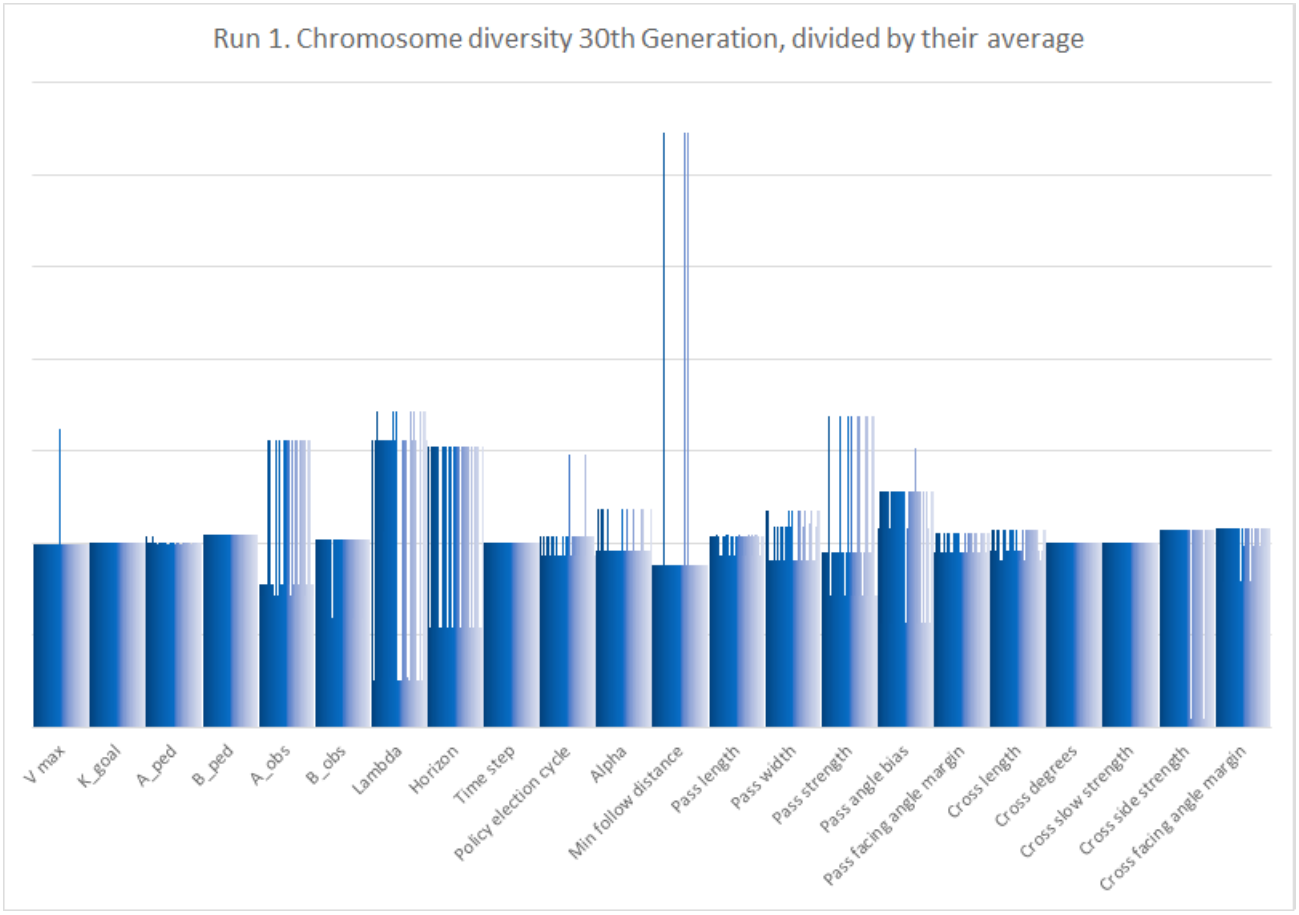

Run 1. Average fitness
Fitness
Generation
fitness_average


Run 1. Average goal reached
Goal reached
Generation
goal_reached_average


Run 1. Average social forces
Social forces
Generation
social_forces_average


Run 1. Chromosome diversity 1st Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin


Run 1. Chromosome diversity 30th Generation
70
60
50
40
30
20
10
0
v_max
k_goal
a_ped
b_ped
a_obs
b_obs
lambda
horizon
time_step
policy_election_cycle
alpha
min_follow_distance
pass_length
pass_width
pass_strength
pass_angle_bias
pass_facing_angle_margin
cross_length
cross_degs
cross_slow_strength
cross_side_strength
cross_facing_angle_margin


Run 1. Chromosome diversity 30th Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin

Run 2. 'normal street' situation.

Evolutionary Algorithm parameters

Weight / parameter name	value
social_importance	0.333333
service_importance	0.333333
technology_importance	0.333333
w_social_forces	1.5151515
w_collision_num	14.7058824
w_runtime	6.25
w_path_length	48.0769231
w_distance_to_goal	83.3333333
w_goal_reached	1000
w_stops_count	35.7142857
generation_size	30
population_size	60
max_mutation_probability	0.05
start_convergence_pressure	0.99
end_convergence_pressure	0.9
run_maxtime	55

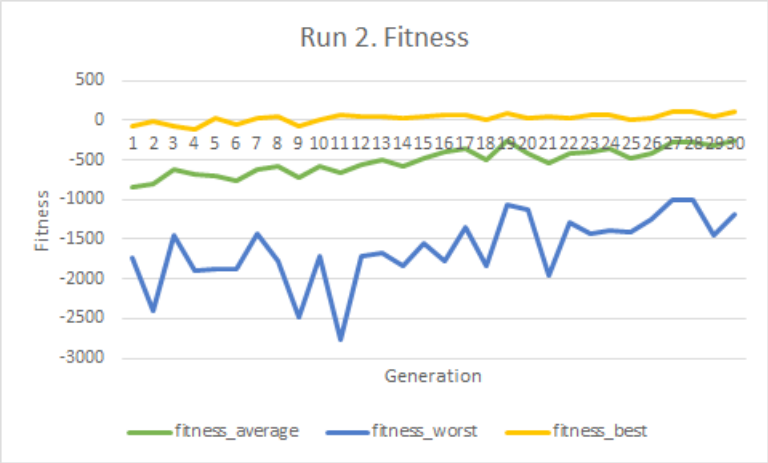
Learned SFM-MPDM parameters

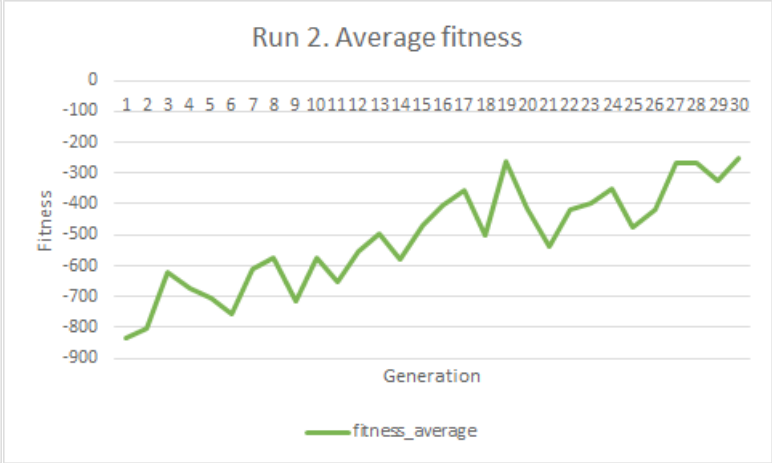
Parameter name	#1	#2	#3
fitness	97.16447	60.65379	58.20966
v_max	0.68953	0.68953	0.68953
k_goal	3.81815	3.81815	3.81815
a_ped	0.32826	0.68995	0.32826
b_ped	1.91954	1.91954	2.3103
a_obs	1.06282	1.02229	1.06282
b_obs	2.57409	0.10161	1.18271
lambda	0.05727	0.05727	0.05727
horizon	2.03993	2.03993	2.03993
time_step	0.15215	0.18817	0.15215
policy_election_cycle	7.45464	2.97136	7.45464
alpha	0.05259	0.39497	0.05259
min_follow_distance	2.46069	2.46069	2.46069
pass_length	0.00534	3.80389	0.00534
pass_width	1.3634	1.3634	1.3634
pass_strength	0.10407	0.10407	0.10407
pass_angle_bias	2.41257	7.87735	7.87735
pass_facing_angle_margin	49.8576	43.31647	49.8576
cross_length	0.44945	2.86691	2.86691
cross_degs	50.9467	50.9467	27.34357
cross_slow_strength	0.77474	0.35598	0.35598
cross_side_strength	0.24257	0.07256	0.07256
cross_facing_angle_margin	40.66473	40.66473	40.66473

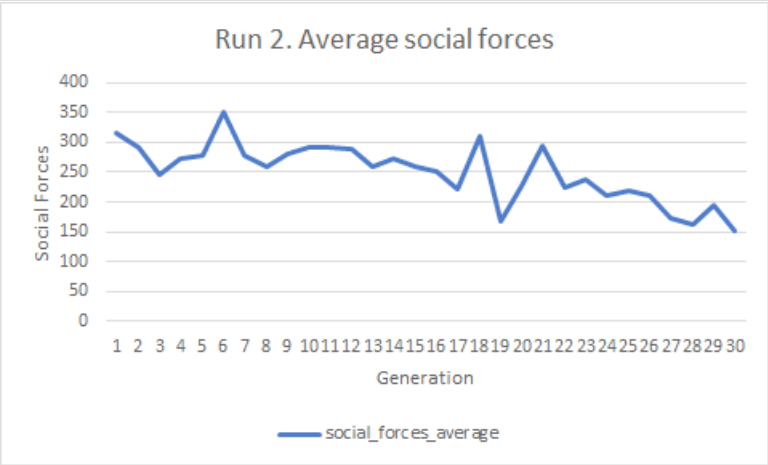
Notes

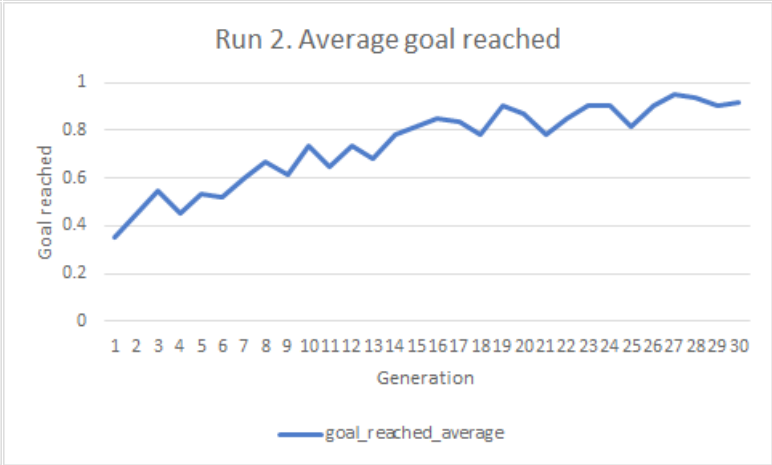
Behavior
Learned how to reach the goal quite consistently. Even managed to get a number of fitness scores above 0. Seems to have converged nicely towards 1 kind of behavior. Theory is that the random pedestrians greatly influence the fluctuations of the fitness scores.

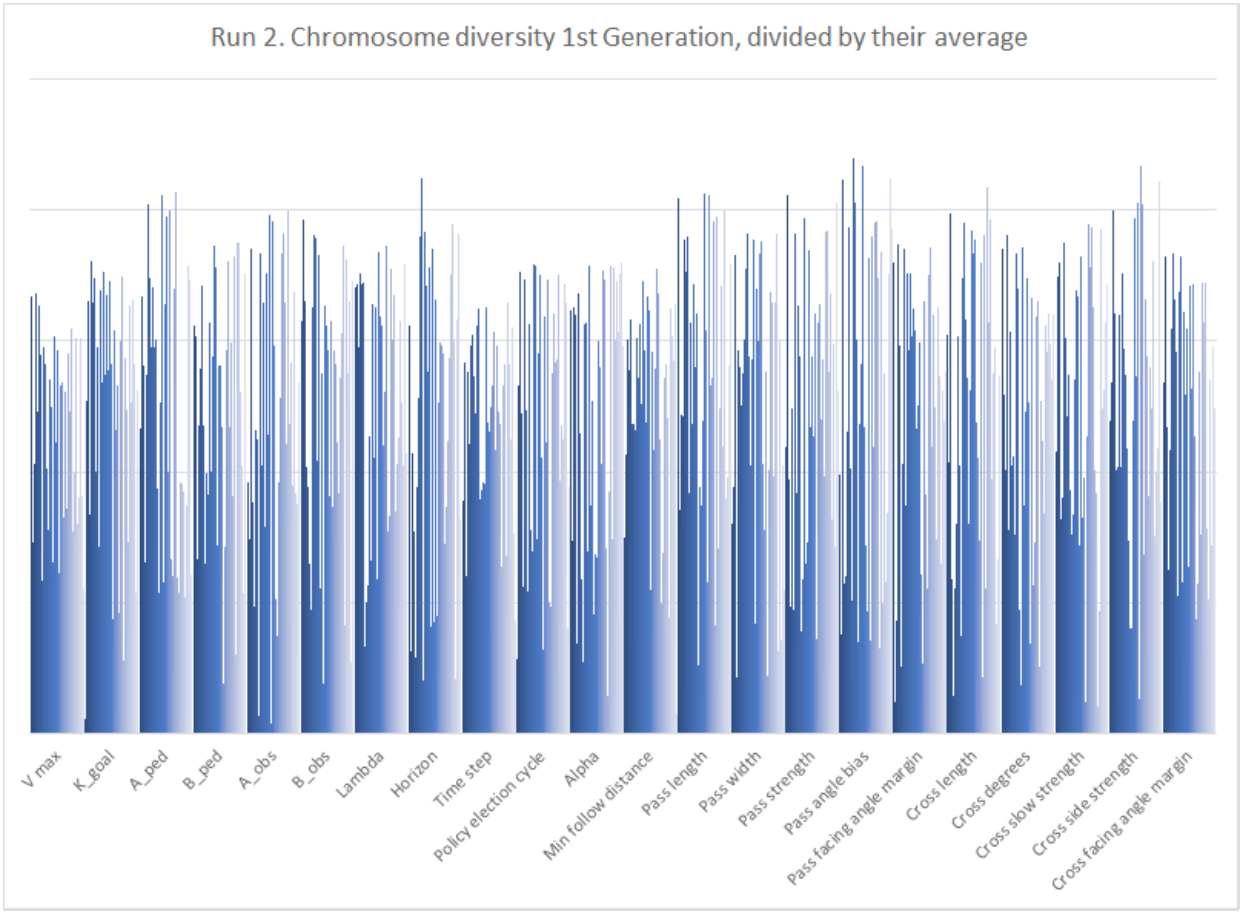
Adjustment for next time
Get rid of the sfm pedestrians and perform a 'empty street' run.

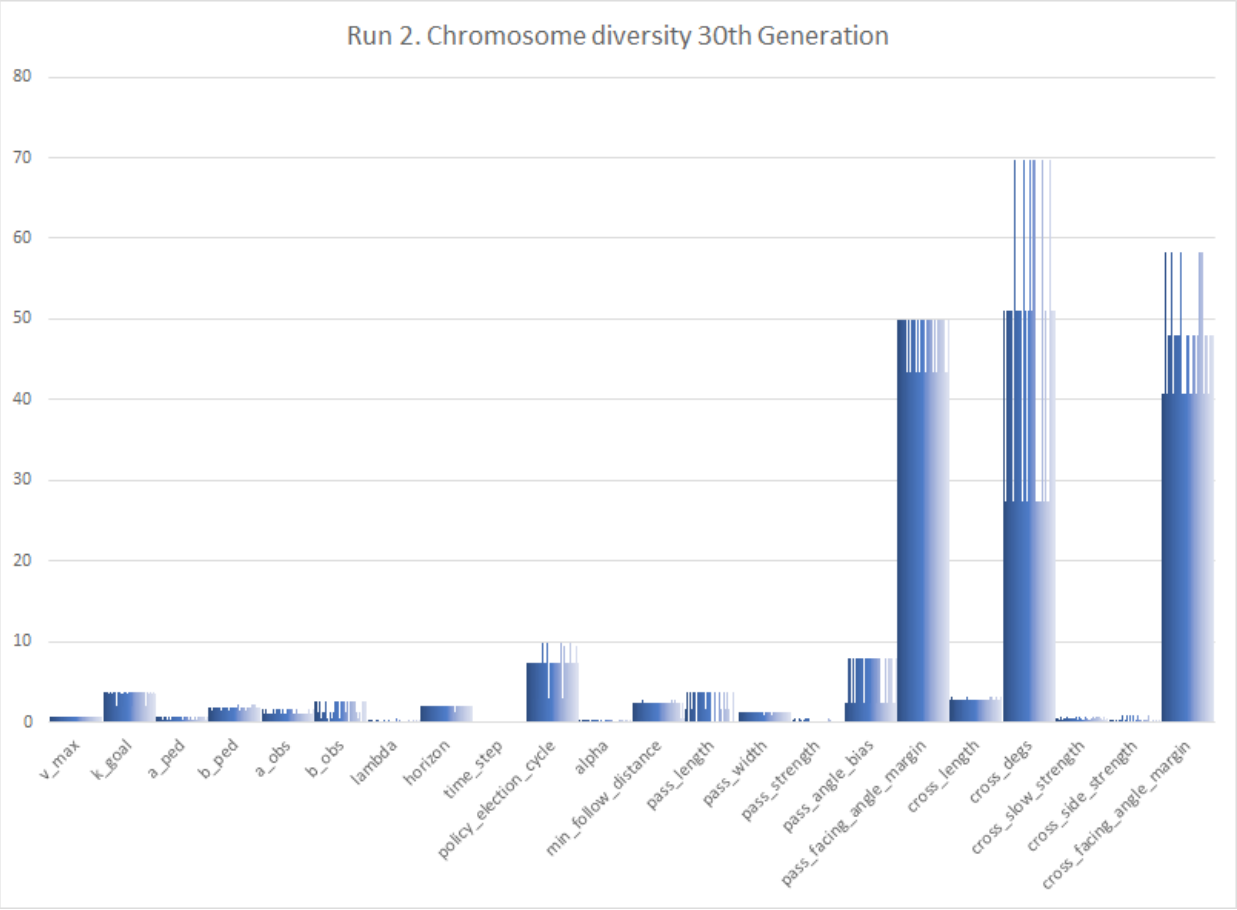

Run 2. Fitness
Fitness
Generation
fitness_average
fitness_worst
fitness_best

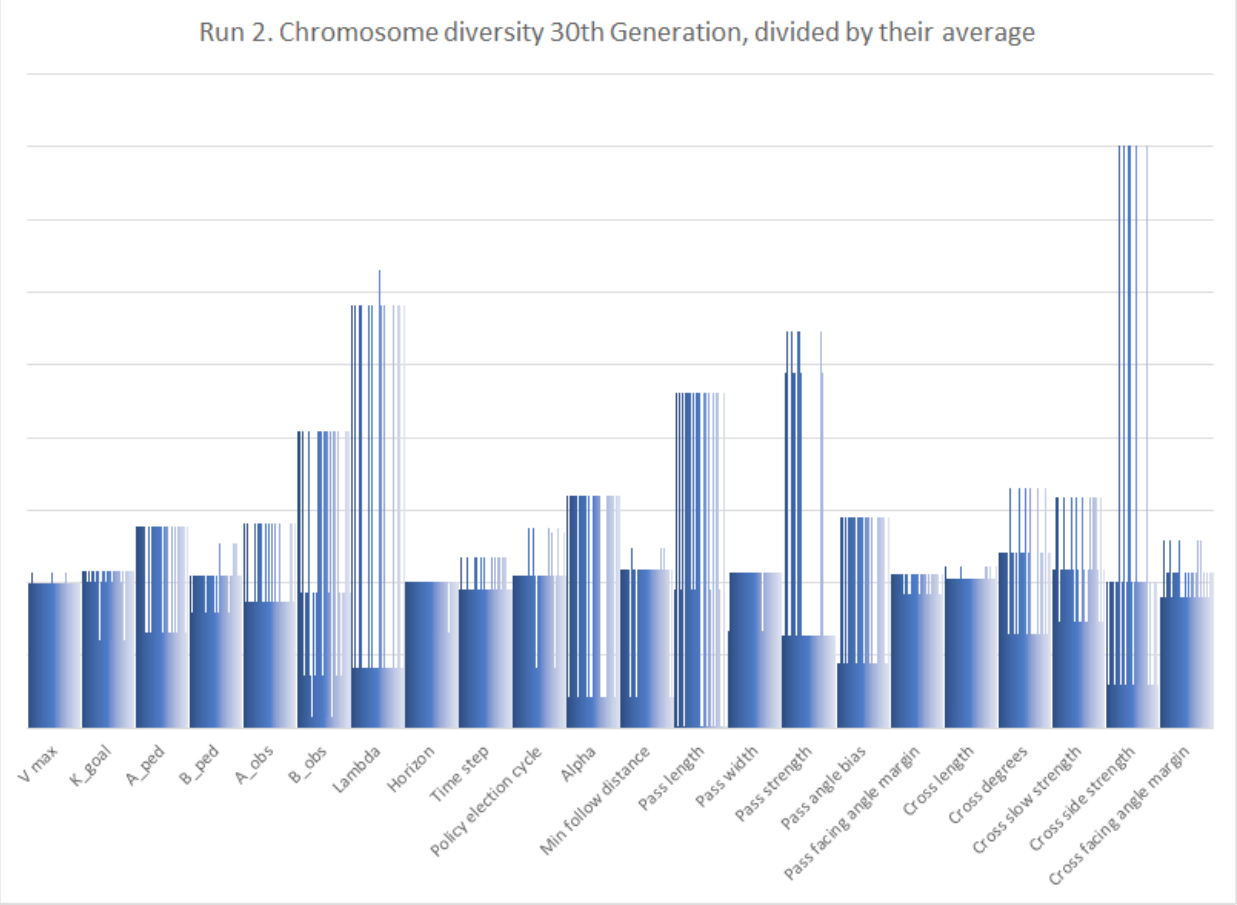

Run 2. Average fitness
Fitness
Generation
fitness_average


Run 2. Average social forces
Social Forces
Generation
social_forces_average


Run 2. Average goal reached
Goal reached
Generation
goal_reached_average


Run 2. Chromosome diversity 1st Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin


Run 2. Chromosome diversity 30th Generation
80
70
60
50
40
30
20
10
0
v_max
k_goal
a_ped
b_ped
a_obs
b_obs
lambda
horizon
time_step
policy_election_cycle
alpha
min_follow_distance
pass_length
pass_width
pass_strength
pass_angle_bias
pass_facing_angle_margin
cross_length
cross_degs
cross_slow_strength
cross_side_strength
cross_facing_angle_margin


Run 2. Chromosome diversity 30th Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin

Run 3. 'empty street with obstacles' situation.

Evolutionary Algorithm parameters

Weight / parameter name	value
social_importance	0
service_importance	0.5
technology_importance	0.5
w_social_forces	1.5151515
w_collision_num	14.7058824
w_runtime	6.25
w_path_length	48.0769231
w_distance_to_goal	83.3333333
w_goal_reached	1000
w_stops_count	35.7142857
generation_size	30
population_size	60
max_mutation_probability	0.05
start_convergence_pressure	0.99
end_convergence_pressure	0.9
run_maxtime	55

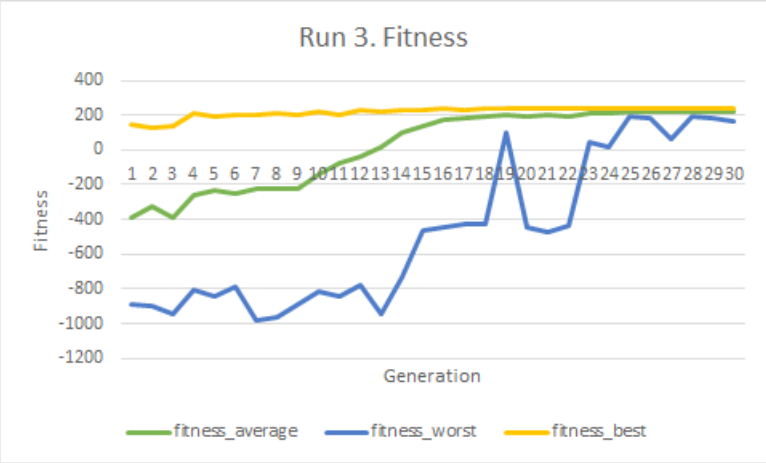
Learned SFM-MPDM parameters

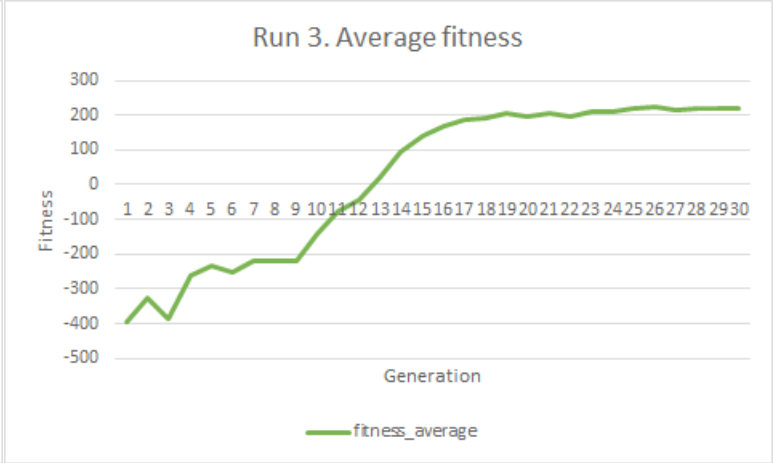
Parameter name	#1	#2	#3
fitness	238.333981	236.269321	235.914148
v_max	0.51806	0.51806	0.51806
k_goal	3.12012	3.12012	3.12012
a_ped	2.47873	2.47873	2.47873
b_ped	2.08274	1.19896	0.25004
a_obs	0.88646	0.88646	0.88646
b_obs	0.07556	0.07556	0.07556
lambda	0.05517	0.05517	0.05517
horizon	2.60464	2.60464	2.64717
time_step	0.36602	0.36602	0.36602
policy_election_cycle	7.79192	7.79192	7.79192
alpha	0.15467	0.15467	0.15467
min_follow_distance	2.70968	2.70968	2.70968
pass_length	4.47429	2.26296	4.47429
pass_width	1.59647	1.59647	1.59647
pass_strength	0.84704	0.9825	0.32012
pass_angle_bias	4.11071	4.11071	5.34978
pass_facing_angle_margin	35.51194	35.51194	35.51194
cross_length	2.32787	2.32787	3.1791
cross_degs	36.76327	62.17514	62.17514
cross_slow_strength	0.02307	0.02307	0.02307
cross_side_strength	0.6669	0.6669	0.6669
cross_facing_angle_margin	50.79877	50.79877	50.79877

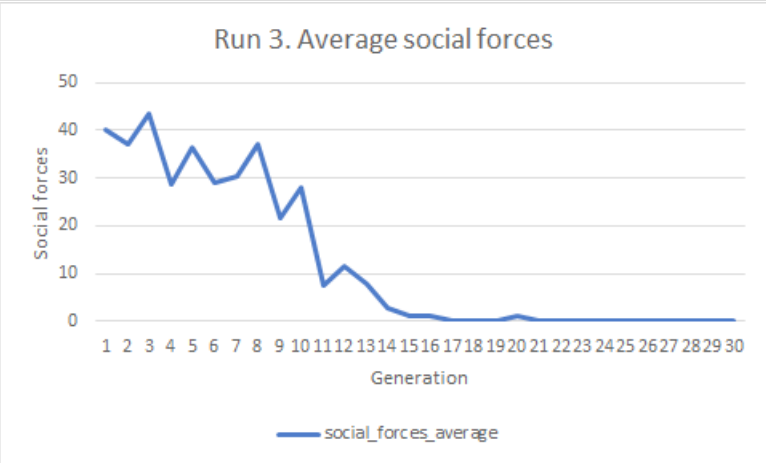
Notes

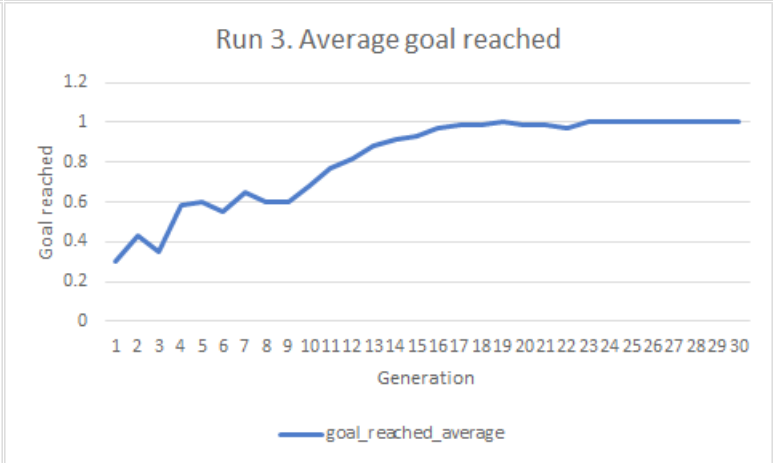
Behavior
No pedestrians, empty street with obstacles.
Learned how to reach the goal very consistently. Final fitness scores are very close, all around 200. It has converged the most compared to the other runs. The theory that the random pedestrians greatly influences the fluctuations of the fitness scores is supported by the lack of large fluctuations in the fitness scores with the absence of the pedestrians.

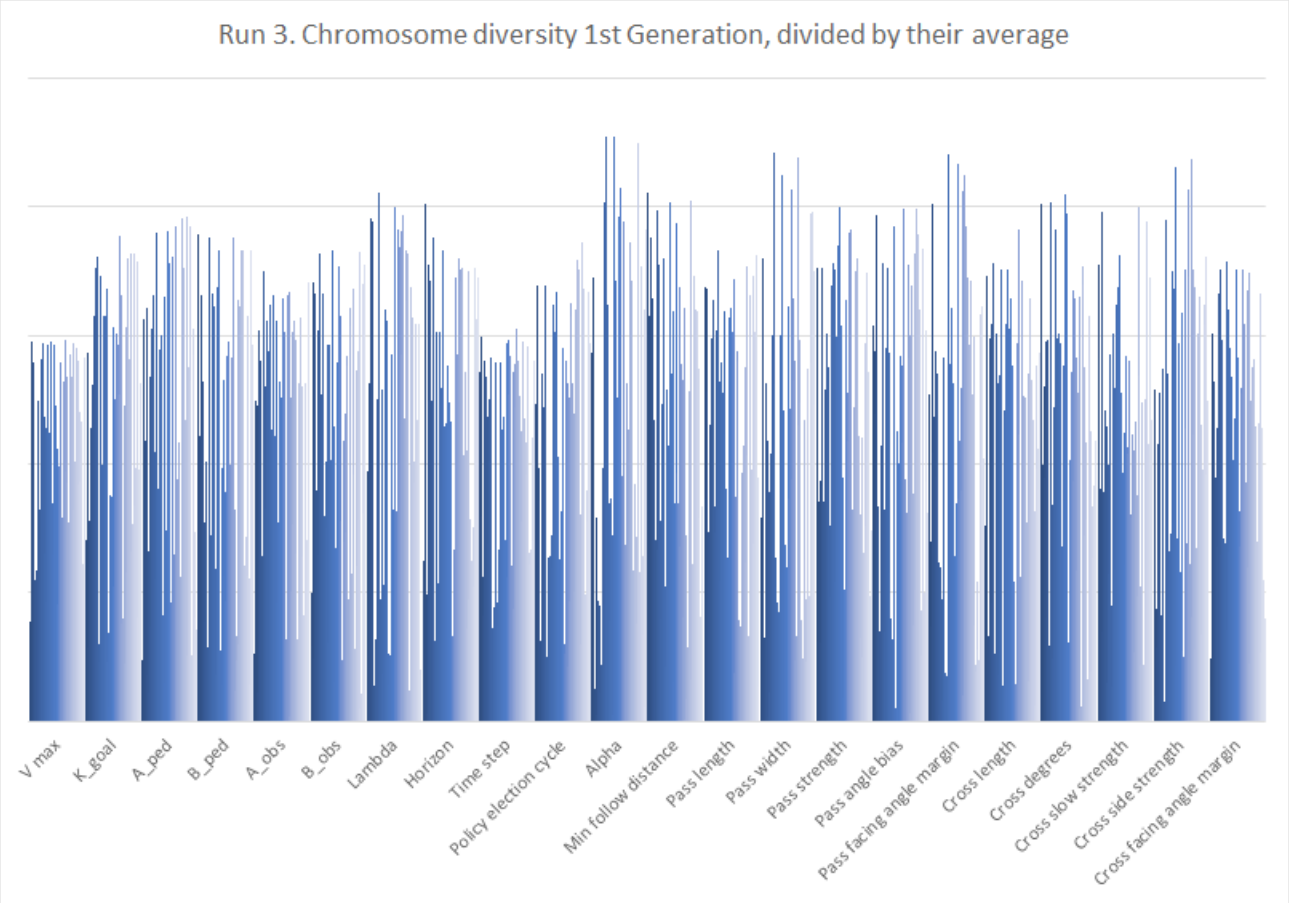
Adjustment for next time
Try with pedestrians, different importance values.

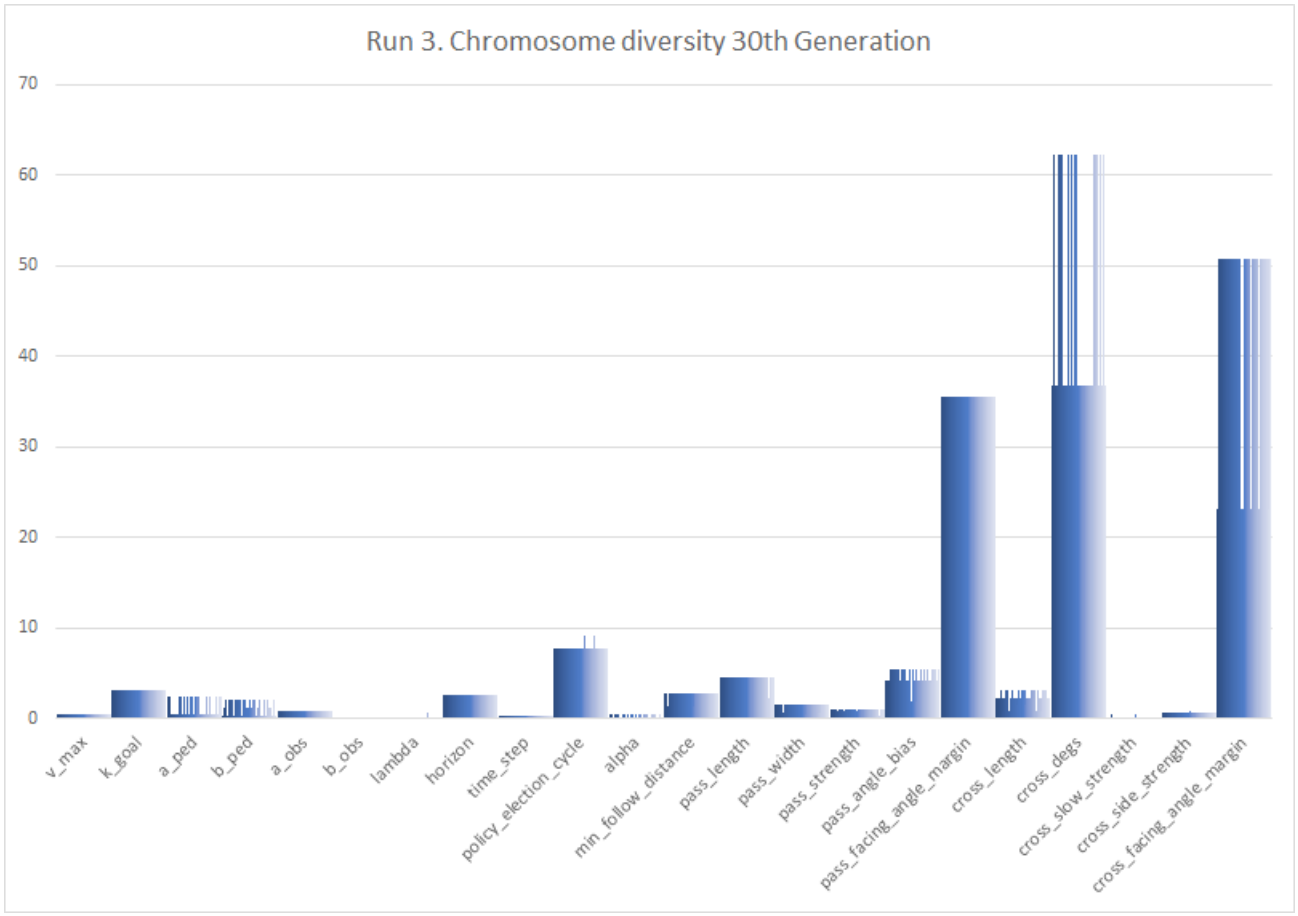

Run 3. Fitness
Fitness
Generation
fitness_average
fitness_worst
fitness_best

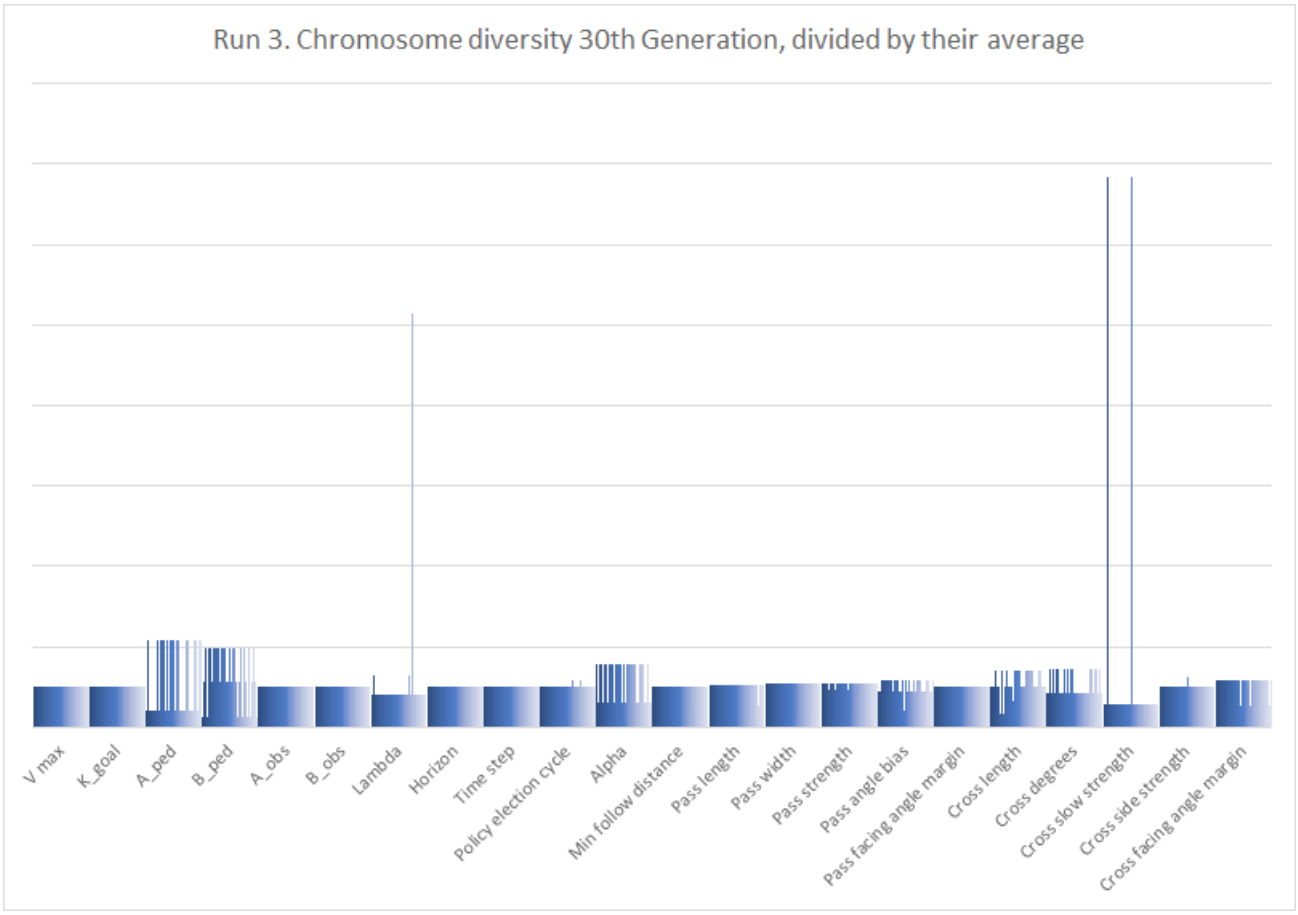

Run 3. Average fitness
Fitness
Generation
fitness_average


Run 3. Average social forces
Social forces
Generation
social_forces_average


Run 3. Average goal reached
Goal reached
Generation
goal_reached_average


Run 3. Chromosome diversity 1st Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin


Run 3. Chromosome diversity 30th Generation
70
60
50
40
30
20
10
0
v_max
k_goal
a_ped
b_ped
a_obs
b_obs
lambda
horizon
time_step
policy_election_cycle
alpha
min_follow_distance
pass_length
pass_width
pass_strength
pass_angle_bias
pass_facing_angle_margin
cross_length
cross_degs
cross_slow_strength
cross_side_strength
cross_facing_angle_margin


Run 3. Chromosome diversity 30th Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin

Run 4. 'normal street' situation.

Evolutionary Algorithm parameters

Weight / parameter name	value
social_importance	0.42
service_importance	0.29
technology_importance	0.29
w_social_forces	1.5151515
w_collision_num	14.7058824
w_runtime	6.25
w_path_length	48.0769231
w_distance_to_goal	83.3333333
w_goal_reached	1000
w_stops_count	35.7142857
generation_size	30
population_size	60
max_mutation_probability	0.05
start_convergence_pressure	0.99
end_convergence_pressure	0.9
run_maxtime	55

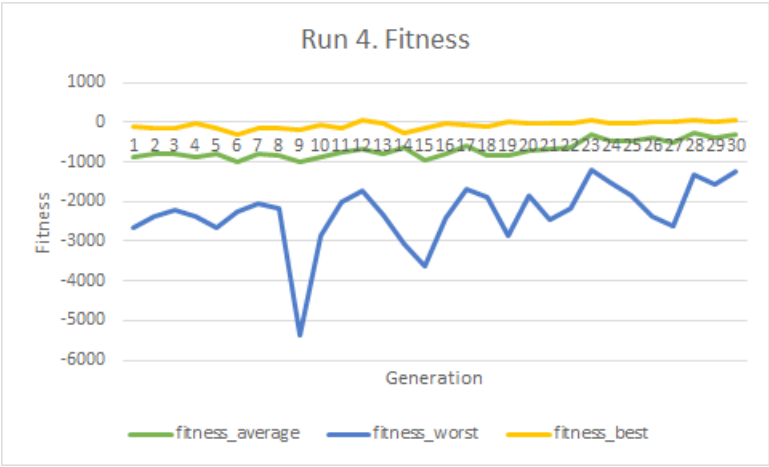
Learned SFM-MPDM parameters

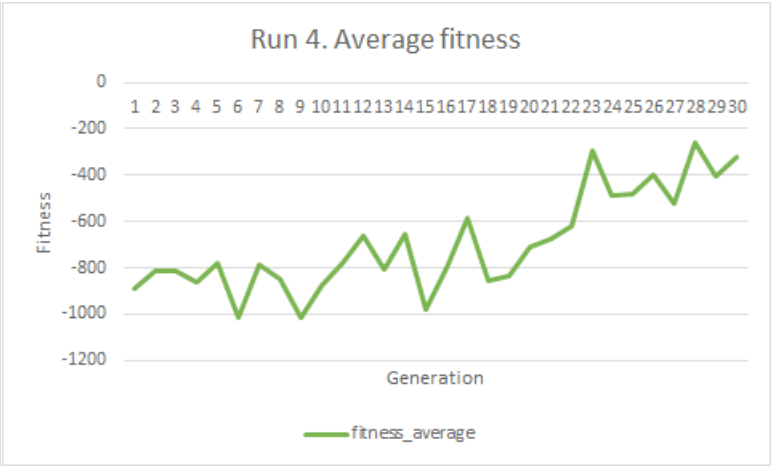
Parameter name	#1	#2	#3
fitness	41.5388621	36.2917385	28.1627371
v_max	0.698	0.698	0.698
k_goal	3.67386	3.67386	3.67386
a_ped	0.2962	1.01662	0.2962
b_ped	2.32125	2.32125	2.33201
a_obs	0.90765	0.94354	0.90765
b_obs	1.24989	1.24989	1.24989
lambda	0.4732	0.8991	0.4732
horizon	2.67403	2.67403	2.67403
time_step	0.35644	0.1957	0.1957
policy_election_cycle	8.15959	8.15959	8.15959
alpha	0.37131	0.30506	0.30506
min_follow_distance	1.88327	1.88327	0.93683
pass_length	0.50046	0.50046	0.50046
pass_width	1.96317	0.62538	0.62538
pass_strength	0.9384	0.40579	0.9384
pass_angle_bias	8.09551	5.48701	8.09551
pass_facing_angle_margin	44.11483	28.95035	44.11483
cross_length	2.68746	2.68746	0.47981
cross_degs	45.11019	56.13022	56.13022
cross_slow_strength	0.64842	0.64842	0.64842
cross_side_strength	0.06757	0.06757	0.06757
cross_facing_angle_margin	35.68947	35.68947	1.489

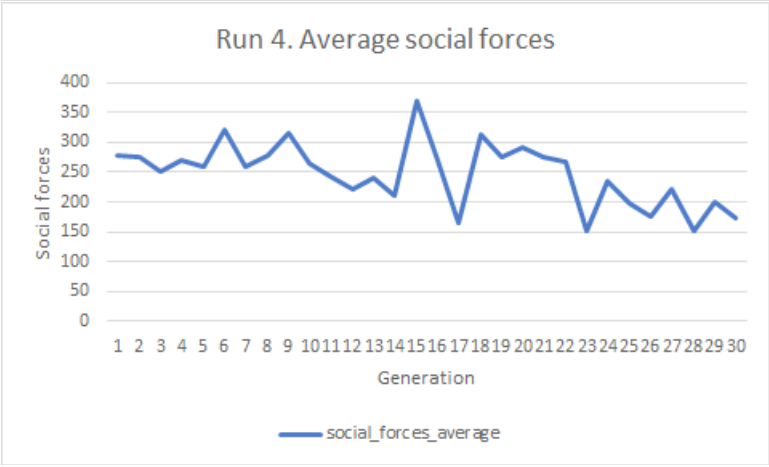
Notes

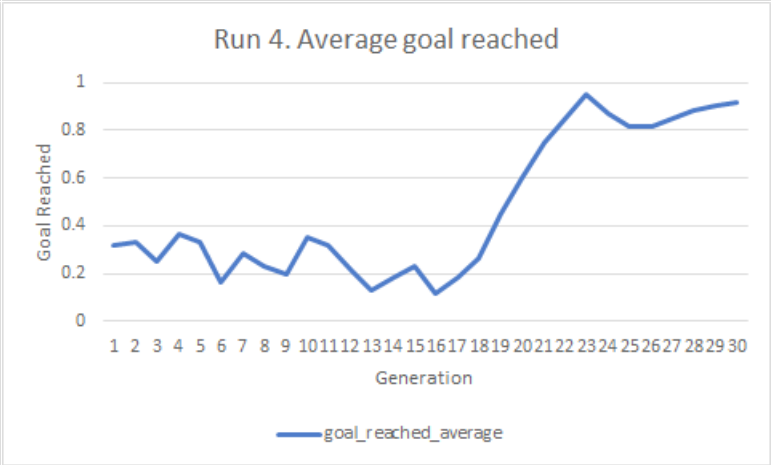
Behavior
Reintroduced pedestrians but altered the importance values from all neutral (0.333333) to instead favor social (0.42, 0.29, 0.29). The resulting behavior is that most of the time the end goal is reached, but fitness values fluctuate a lot.

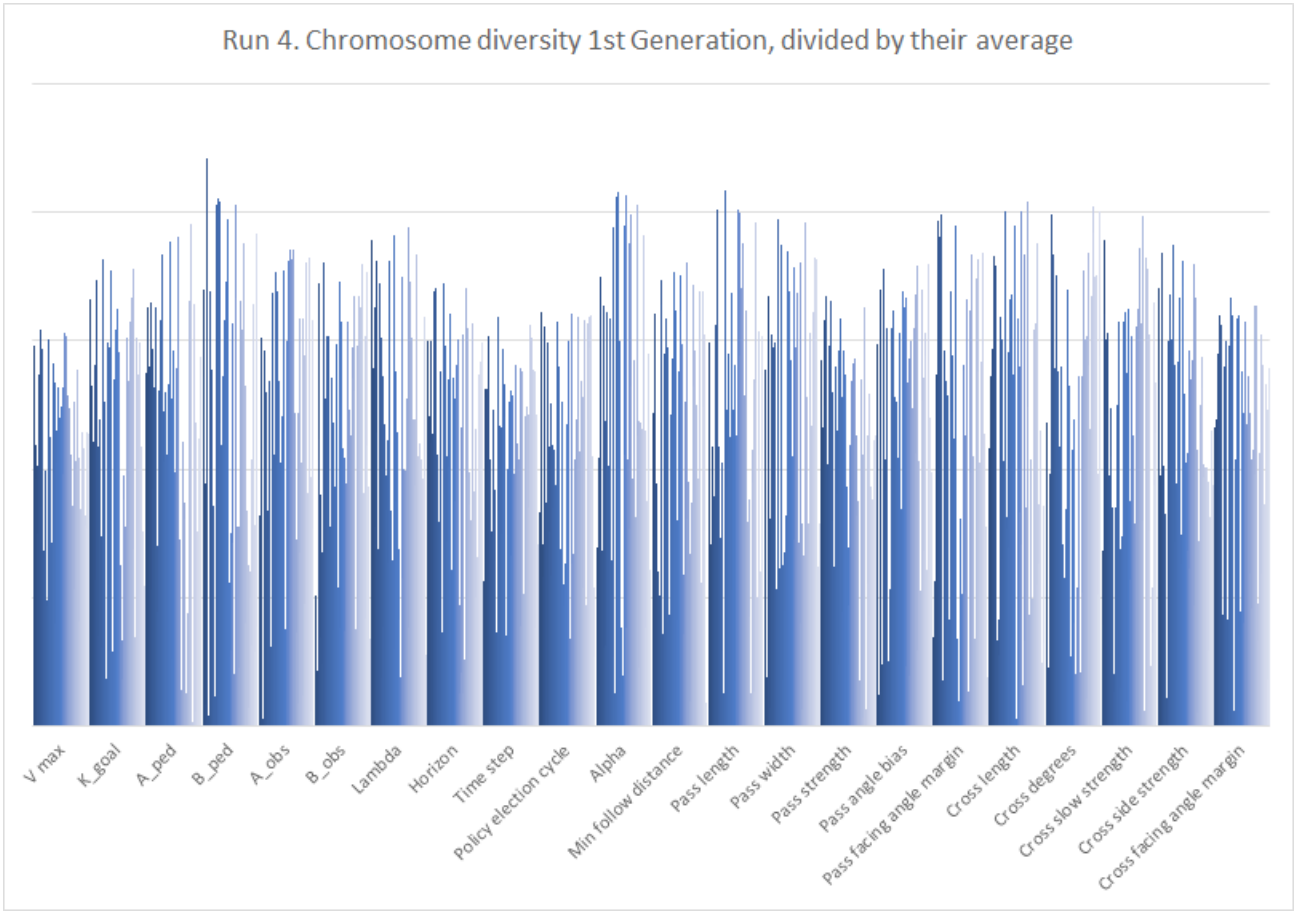
Adjustment for next time (if there is one)
Instead decrease social importance, increase service and technology importance.

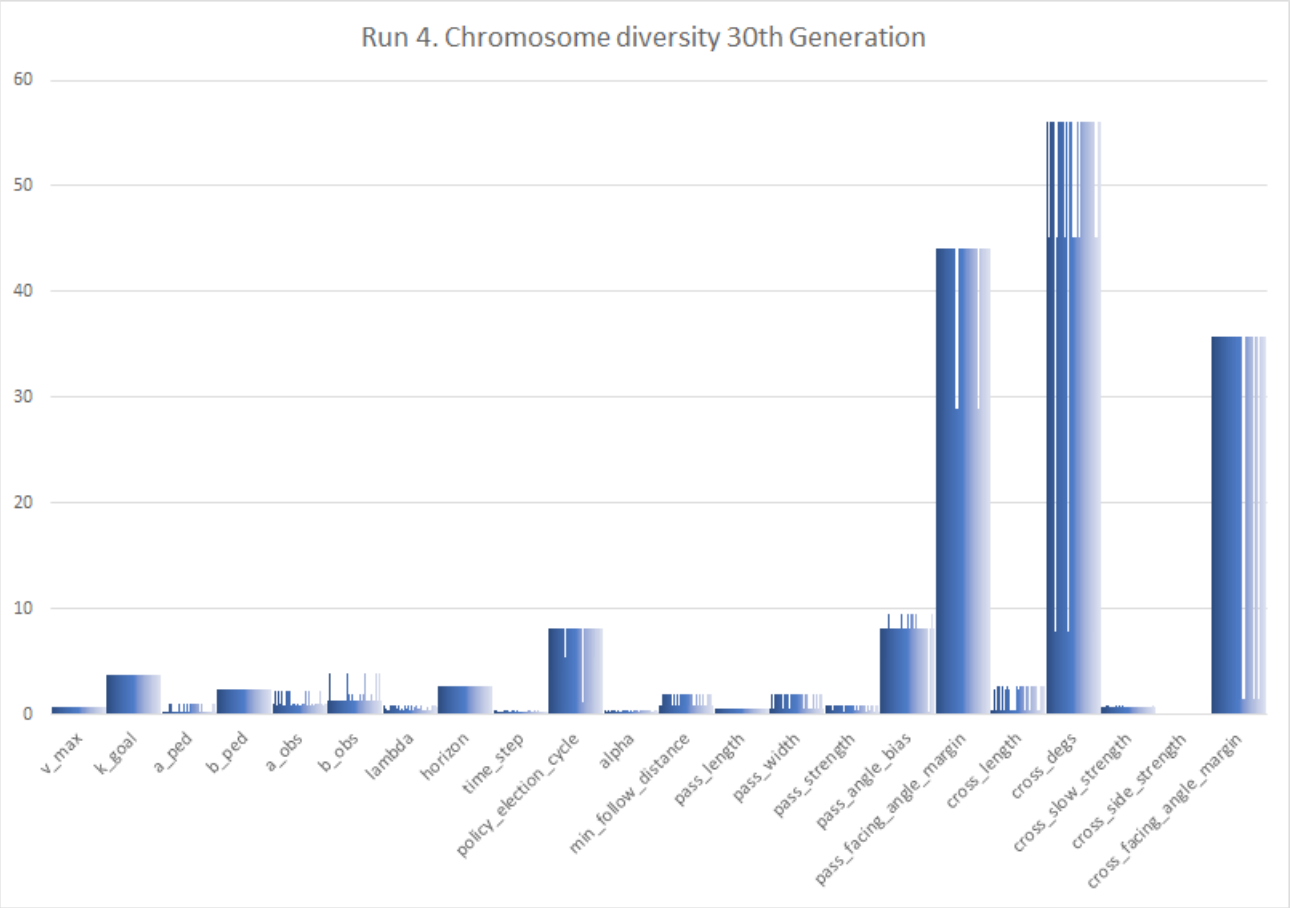

Run 4. Fitness
Fitness
Generation
fitness_average
fitness_worst
fitness_best

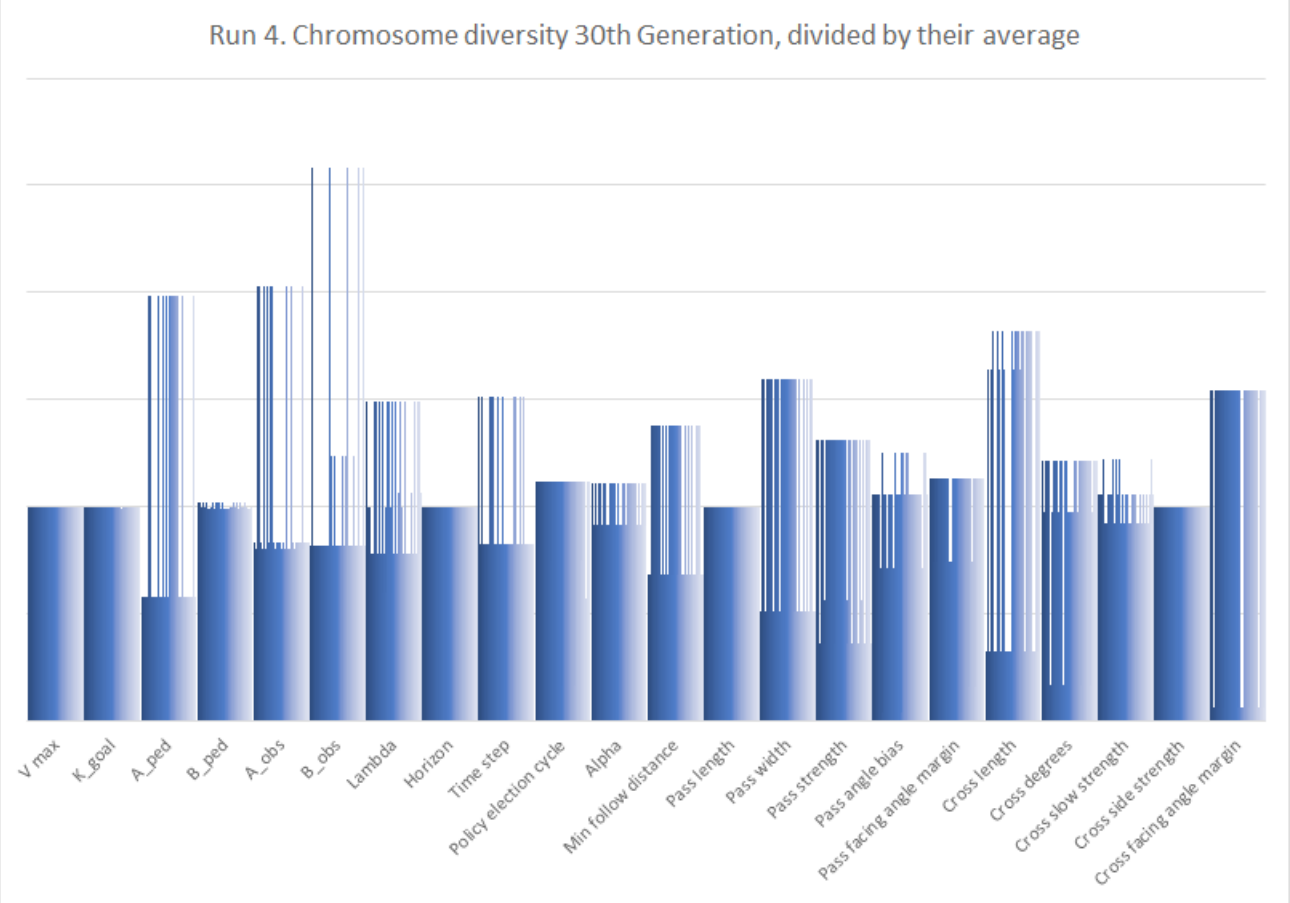

Run 4. Average fitness
Fitness
Generation
fitness_average


Run 4. Average social forces
Social forces
Generation
social_forces_average


Run 4. Average goal reached
Goal Reached
Generation
goal_reached_average


Run 4. Chromosome diversity 1st Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin


Run 4. Chromosome diversity 30th Generation
60
50
40
30
20
10
0
v_max
k_goal
a_ped
b_ped
a_obs
b_obs
lambda
horizon
time_step
policy_election_cycle
alpha
min_follow_distance
pass_length
pass_width
pass_strength
pass_angle_bias
pass_facing_angle_margin
cross_length
cross_degs
cross_slow_strength
cross_side_strength
cross_facing_angle_margin


Run 4. Chromosome diversity 30th Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin

Run 5. 'normal street' situation.

Evolutionary Algorithm parameters

Weight / parameter name	value
social_importance	0
service_importance	0.5
technology_importance	0.5
w_social_forces	1.5151515
w_collision_num	14.7058824
w_runtime	6.25
w_path_length	48.0769231
w_distance_to_goal	83.3333333
w_goal_reached	1000
w_stops_count	35.7142857
generation_size	30
population_size	60
max_mutation_probability	0.05
start_convergence_pressure	0.99
end_convergence_pressure	0.9
run_maxtime	55

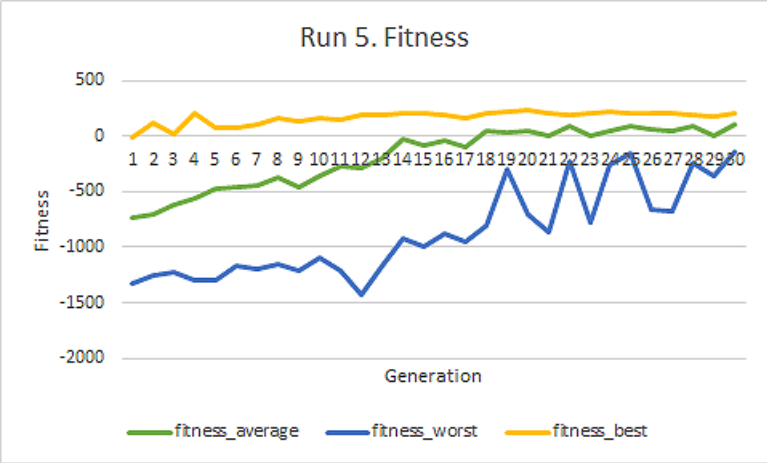
Learned SFM-MPDM parameters

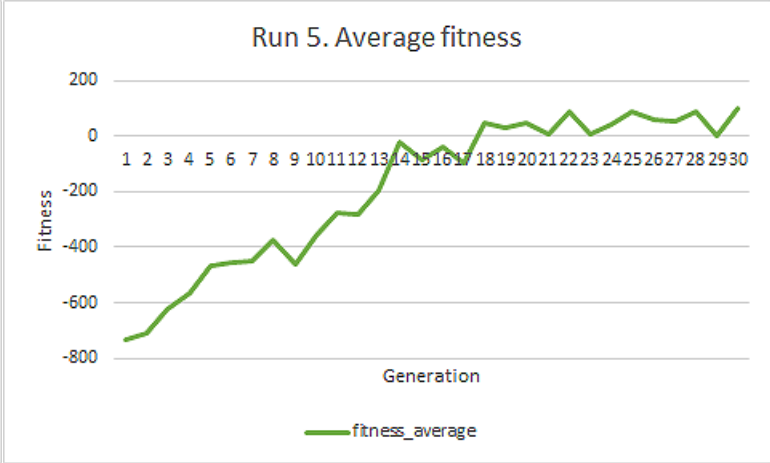
Parameter name	#1	#2	#3
fitness	211.011463	206.925123	203.975446
v_max	0.58387	0.58387	0.58387
k_goal	3.19291	3.19291	2.77437
a_ped	0.25489	0.25489	0.25489
b_ped	1.46129	1.46129	1.46129
a_obs	2.33652	2.33652	1.88557
b_obs	0.17423	0.17423	0.17423
lambda	0.32102	0.82761	0.82761
horizon	0.36126	0.36126	0.36126
time_step	0.31772	0.31772	0.31772
policy_election_cycle	7.52633	9.21024	7.52633
alpha	0.20266	0.17234	0.20266
min_follow_distance	2.72535	2.46646	2.72535
pass_length	1.12377	1.12377	1.12377
pass_width	1.86941	0.55059	0.926
pass_strength	0.74918	0.51506	0.51506
pass_angle_bias	4.86986	4.86986	6.96231
pass_facing_angle_margin	45.66897	3.60673	36.91491
cross_length	0.82262	0.82262	0.82262
cross_degs	37.38802	22.42952	22.42952
cross_slow_strength	0.24246	0.3084	0.24246
cross_side_strength	0.15497	0.15497	0.15497
cross_facing_angle_margin	15.62495	15.62495	10.13643

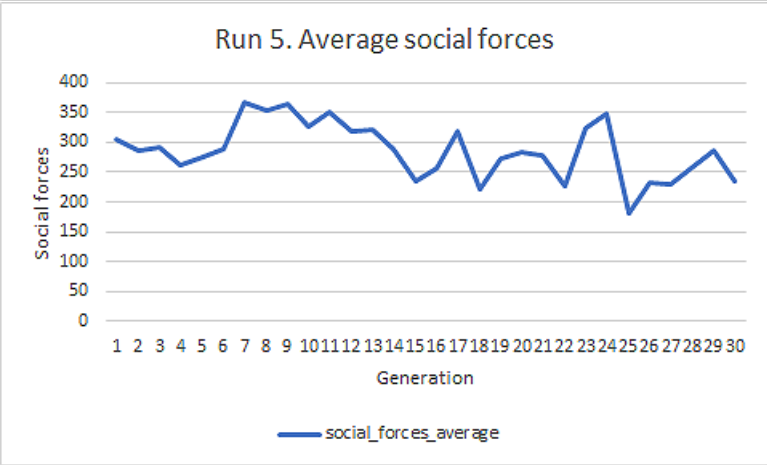
Notes

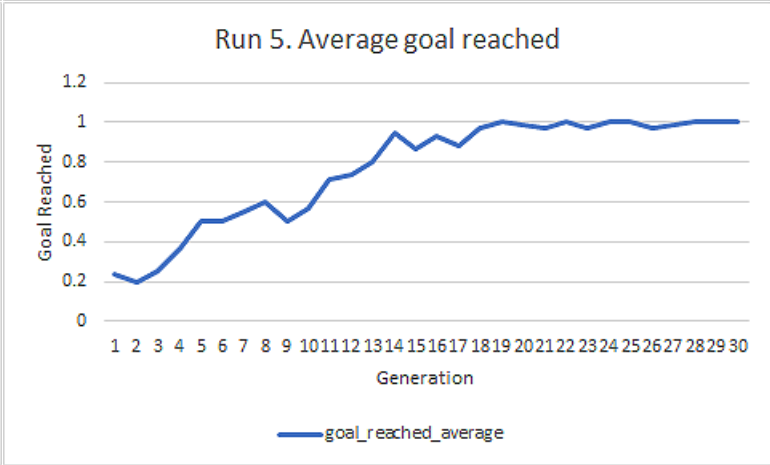
Behavior
Social importance factor was set to 0, other two aspect are 0.5 each. This resulted in behavior which always reaches the end goal, which does not keep a distance to the pedestrians but which does stop before colliding.

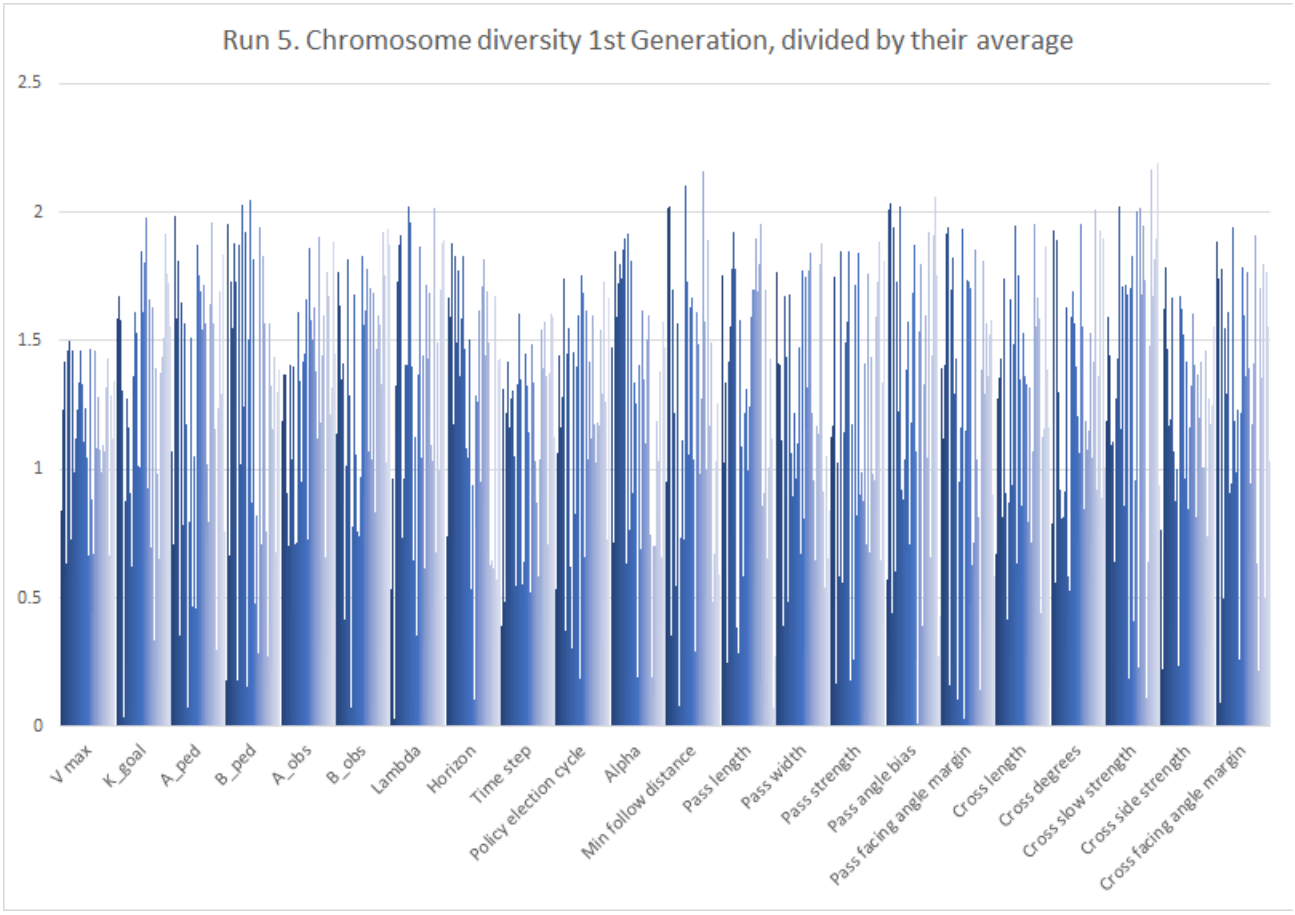
Adjustment for next time
Increase social importance, and decrease service and technology importance to explore the other side of the (near) extreme.

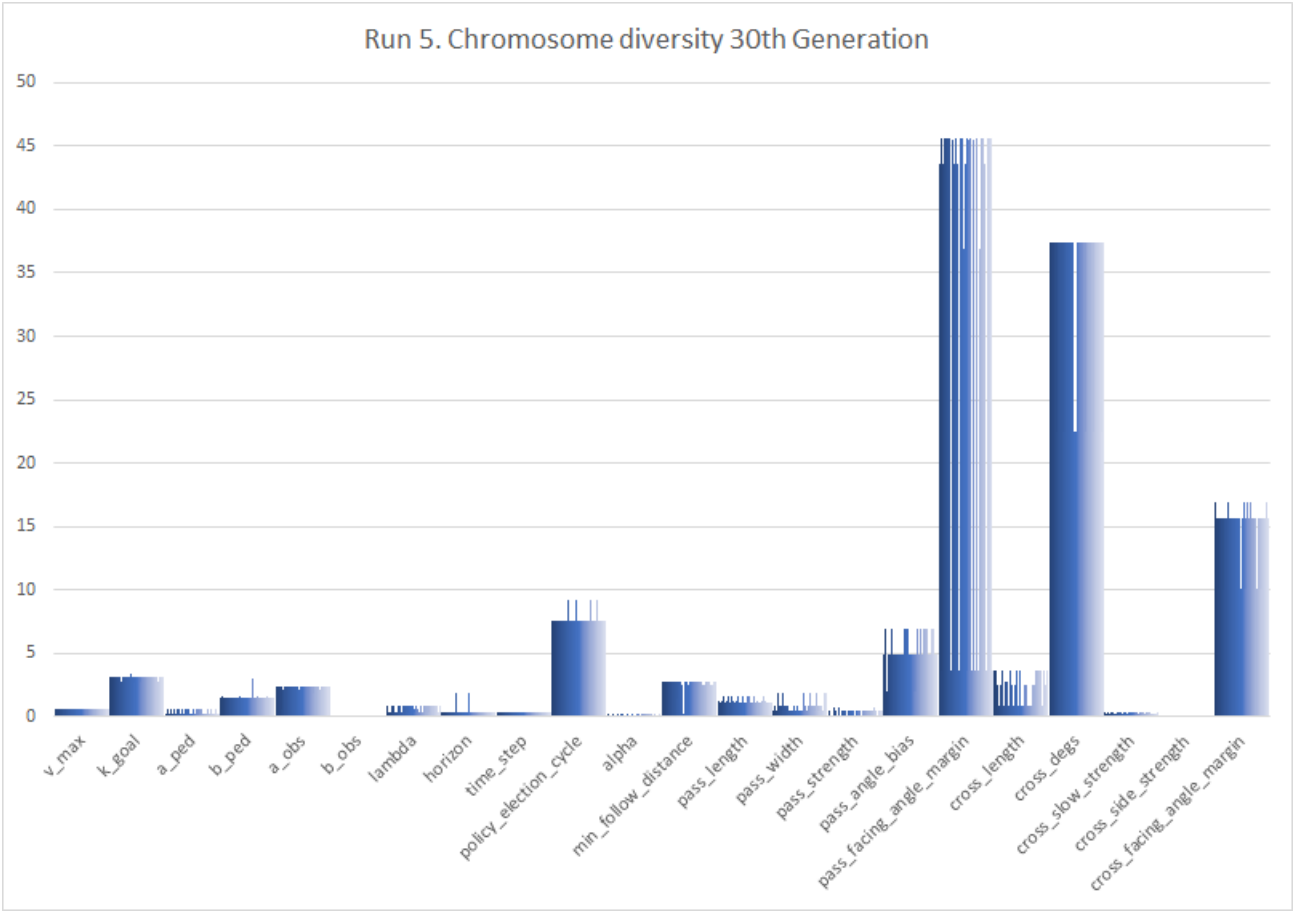

Run 5. Fitness
Fitness
Generation
fitness_average
fitness_worst
fitness_best

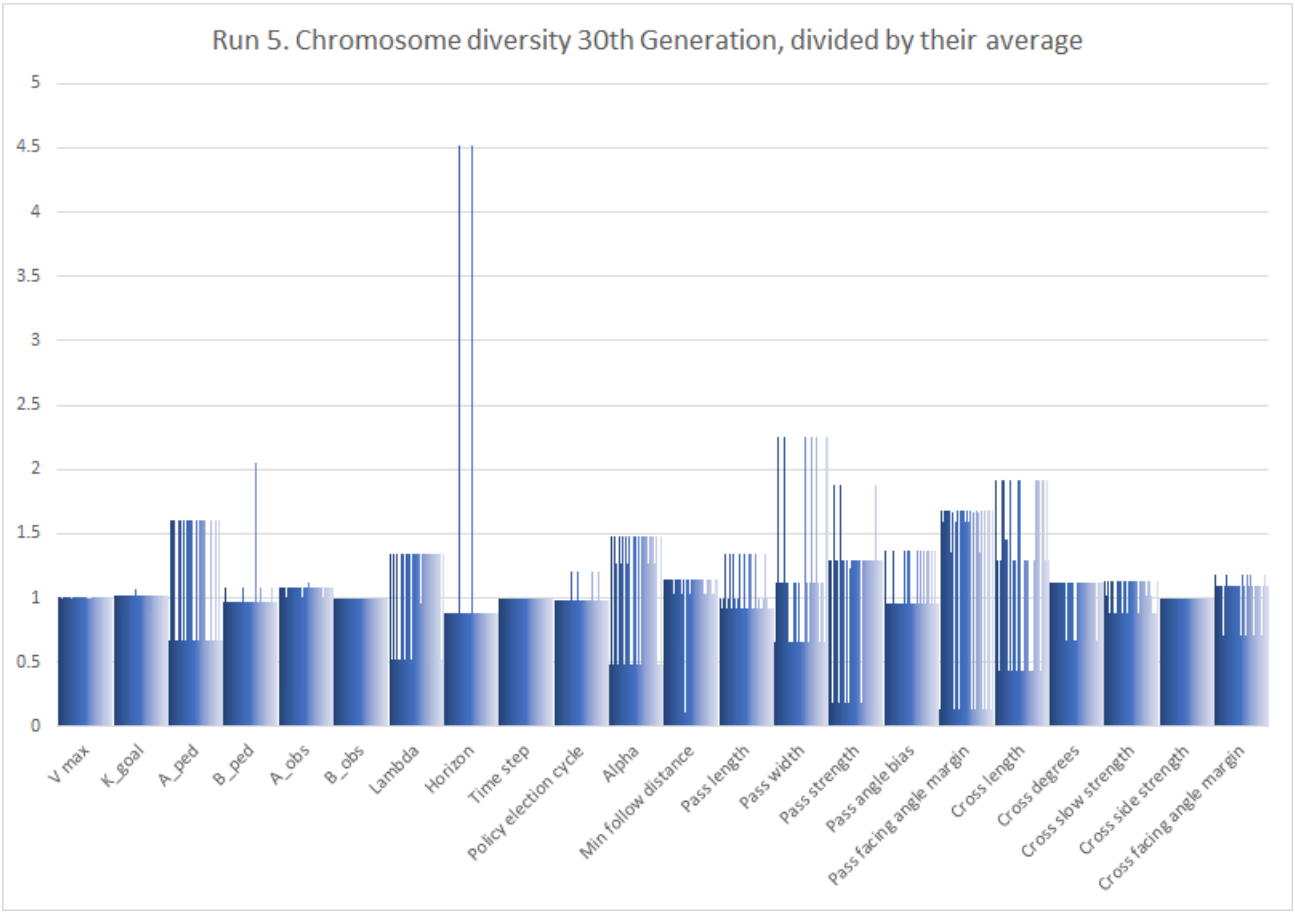

Run 5. Average fitness
Fitness
Generation
fitness_average


Run 5. Average social forces
Social forces
Generation
social_forces_average


Run 5. Average goal reached
Goal Reached
Generation
goal_reached_average


Run 5. Chromosome diversity 1st Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin


Run 5. Chromosome diversity 30th Generation
50
45
40
35
30
25
20
15
10
5
0
v_max
k_goal
a_ped
b_ped
a_obs
b_obs
lambda
horizon
time_step
policy_election_cycle
alpha
min_follow_distance
pass_length
pass_width
pass_strength
pass_angle_bias
pass_facing_angle_margin
cross_length
cross_degs
cross_slow_strength
cross_side_strength
cross_facing_angle_margin


Run 5. Chromosome diversity 30th Generation, divided by their average
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin

Run 6. 'normal street' situation.

Evolutionary Algorithm parameters

Weight / parameter name	value
social_importance	0.6
service_importance	0.2
technology_importance	0.2
w_social_forces	1.5151515
w_collision_num	14.7058824
w_runtime	6.25
w_path_length	48.0769231
w_distance_to_goal	83.3333333
w_goal_reached	1000
w_stops_count	35.7142857
generation_size	30
population_size	60
max_mutation_probability	0.05
start_convergence_pressure	0.99
end_convergence_pressure	0.9
run_maxtime	55

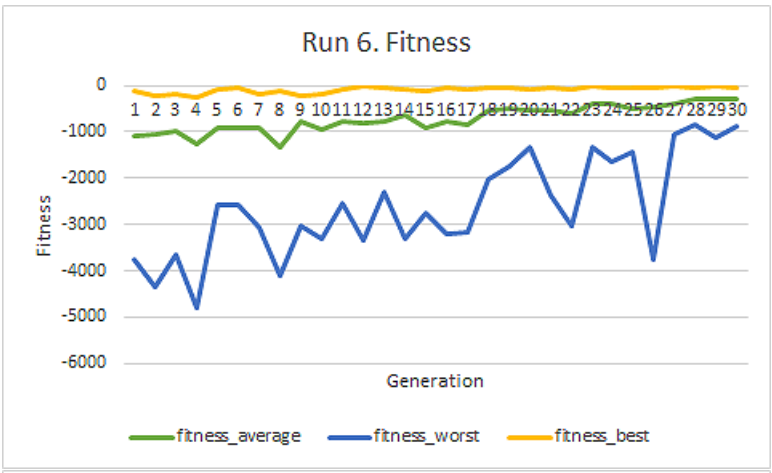
Learned SFM-MPDM parameters

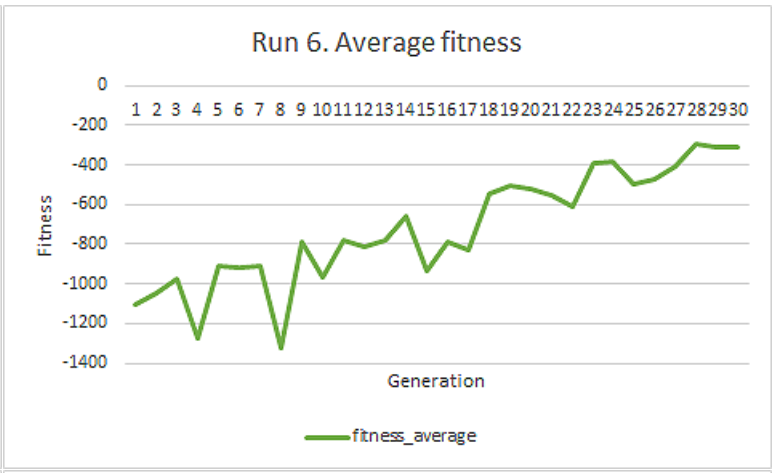
Parameter name	#1	#2	#3
fitness	-43.775809	-48.247472	-55.160059
v_max	0.67161	0.67161	0.67161
k_goal	2.67133	2.67133	3.5602
a_ped	0.45238	0.45238	0.45238
b_ped	0.76096	0.9363	1.39821
a_obs	2.16841	1.1269	1.77012
b_obs	3.40831	1.97282	0.6597
lambda	0.71396	0.71396	0.71396
horizon	2.77651	1.9113	2.77651
time_step	0.10005	0.10005	0.10005
policy_election_cycle	2.13098	2.13098	2.13098
alpha	0.00369	0.00369	0.00369
min_follow_distance	0.52897	2.26	0.52897
pass_length	2.64744	3.19035	2.03007
pass_width	1.62944	1.62944	1.76886
pass_strength	0.30627	0.30627	0.30627
pass_angle_bias	8.06671	4.40356	7.9276
pass_facing_angle_margin	13.15396	13.15396	13.15396
cross_length	1.15354	1.15354	1.15354
cross_degs	70.82023	70.98359	70.82023
cross_slow_strength	0.56532	0.56532	0.56532
cross_side_strength	0.41808	0.11078	0.12286
cross_facing_angle_margin	36.56424	36.56424	37.7552

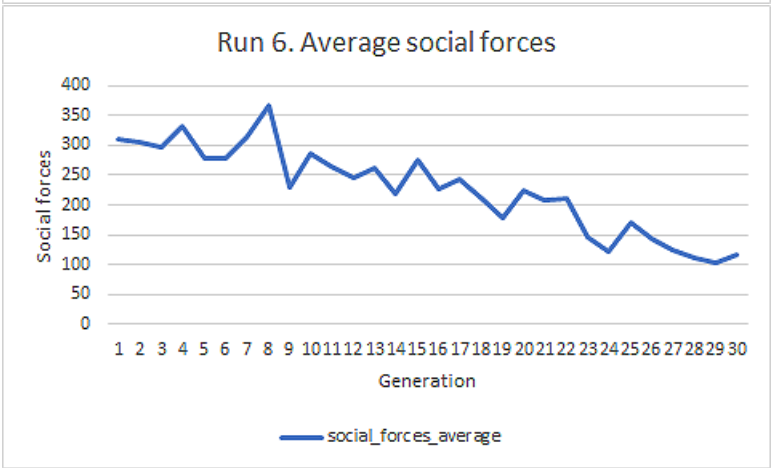
Notes

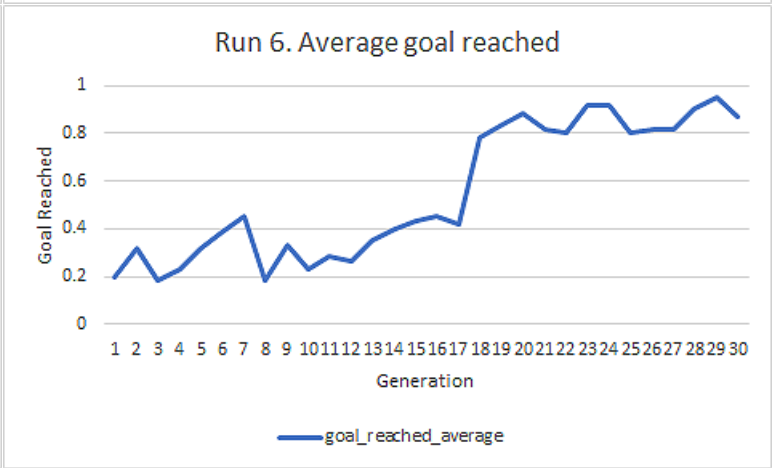
Behavior
Social importance factor was set to 0.6, other two aspect are 0.2 each. This resulted in behavior which can best be described as socially chaotic. The goal was reached 9 out of 10 times, inefficient driving, comes across as insecure.

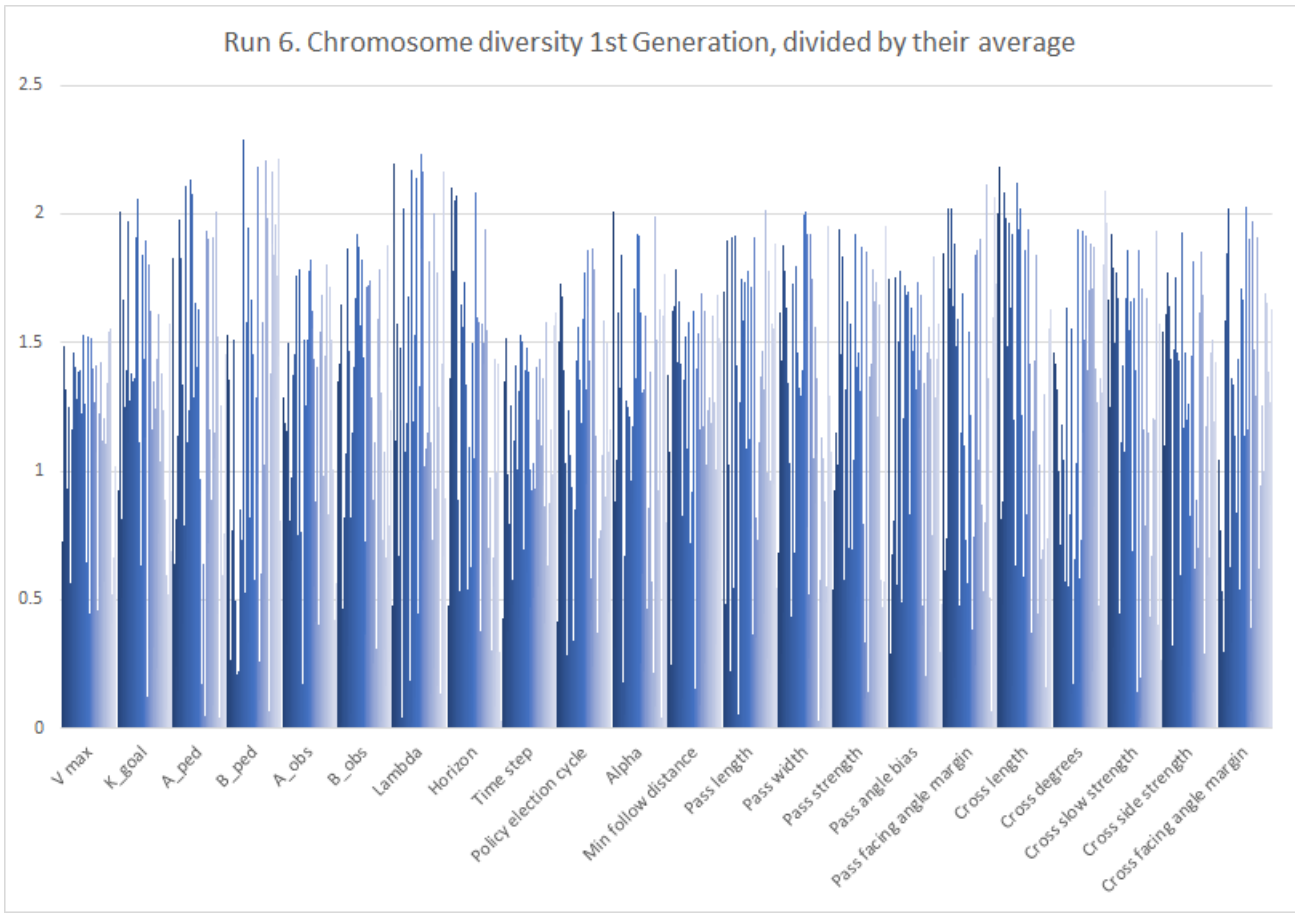
Adjustment for next time
Alter the set up of the evolutionary algorithm to better reflect the marked STS-triangle position.

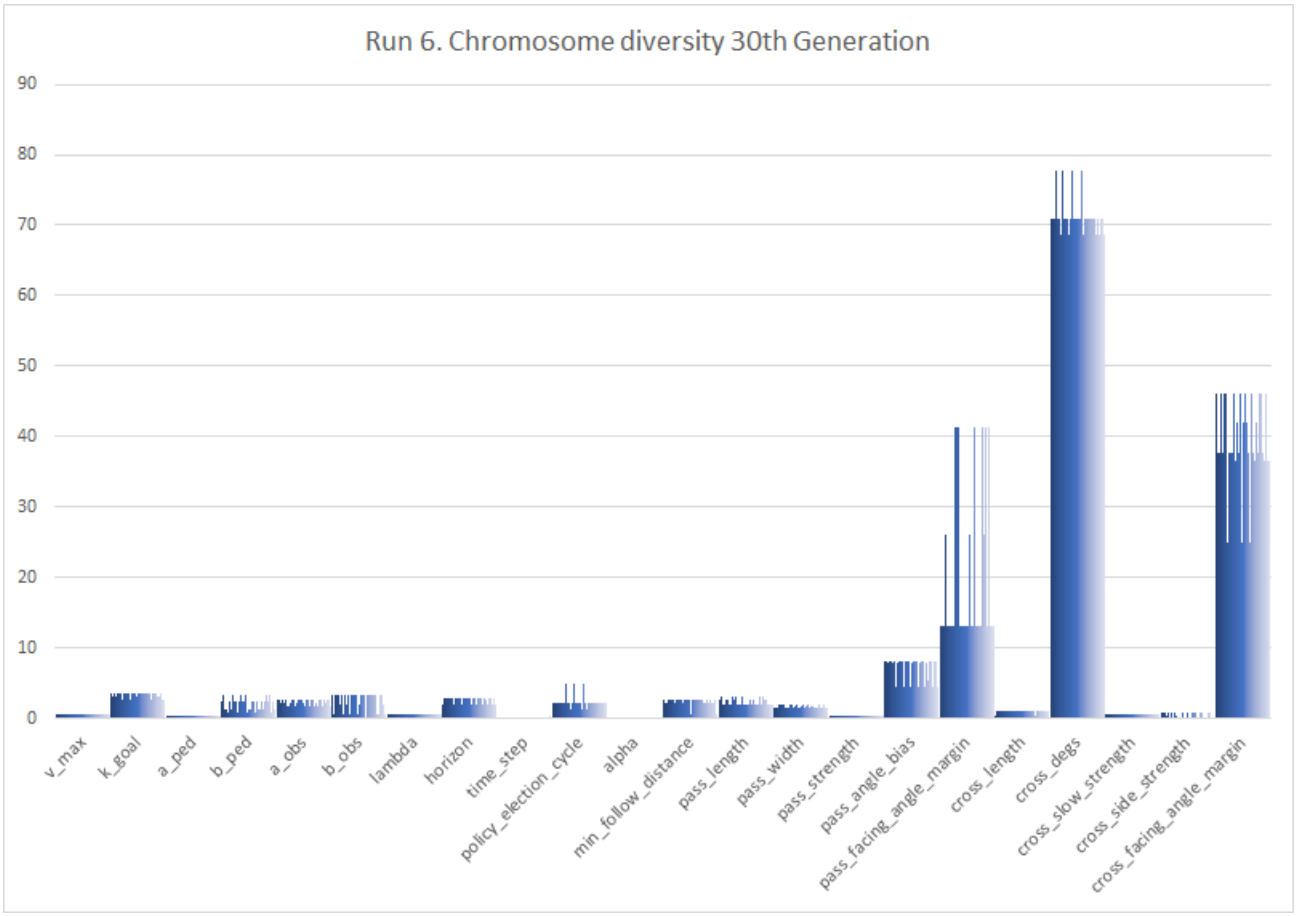

Run 6. Fitness
Fitness
Generation
fitness_average
fitness_worst
fitness_best

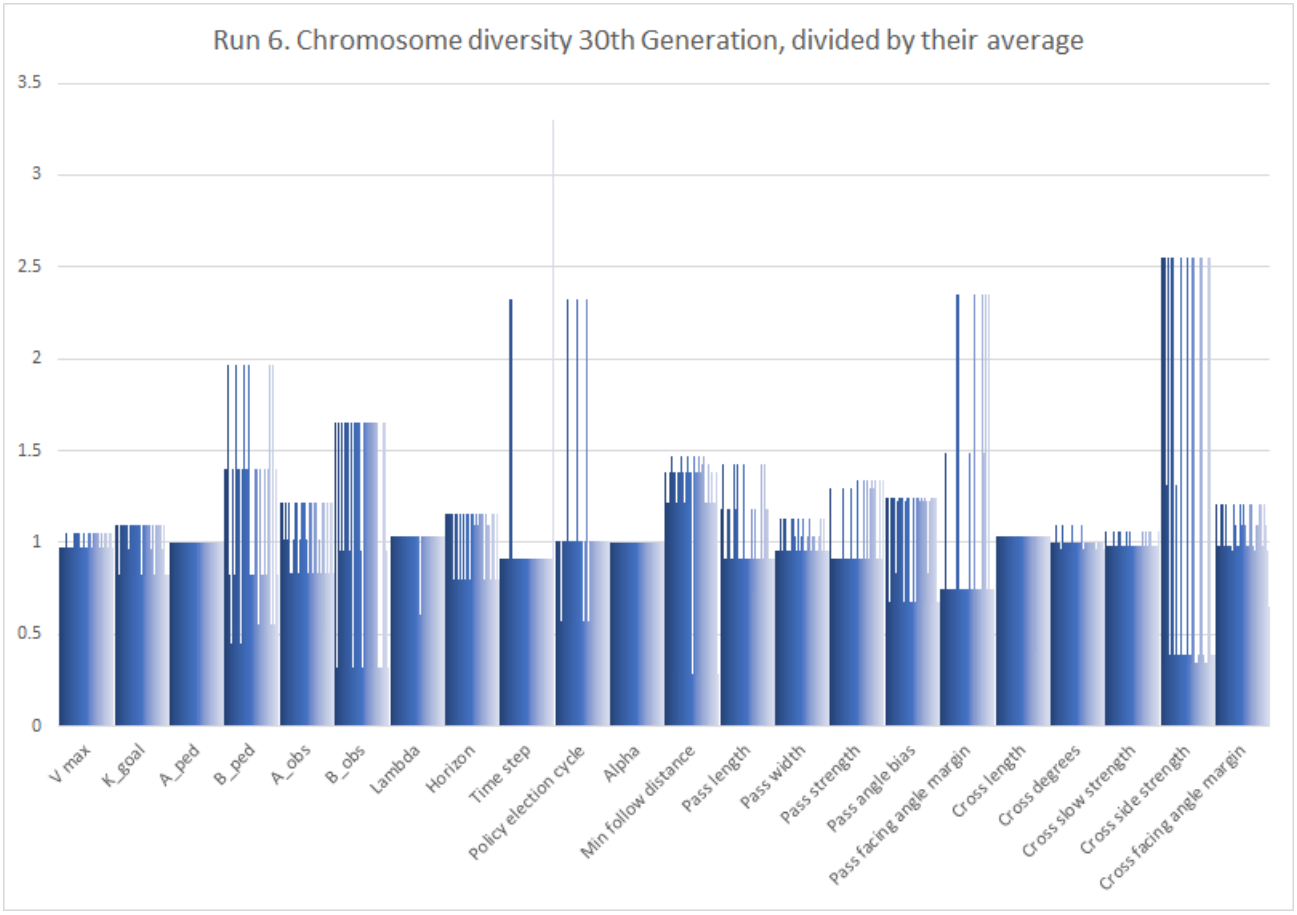

Run 6. Average fitness
Fitness
Generation
fitness_average

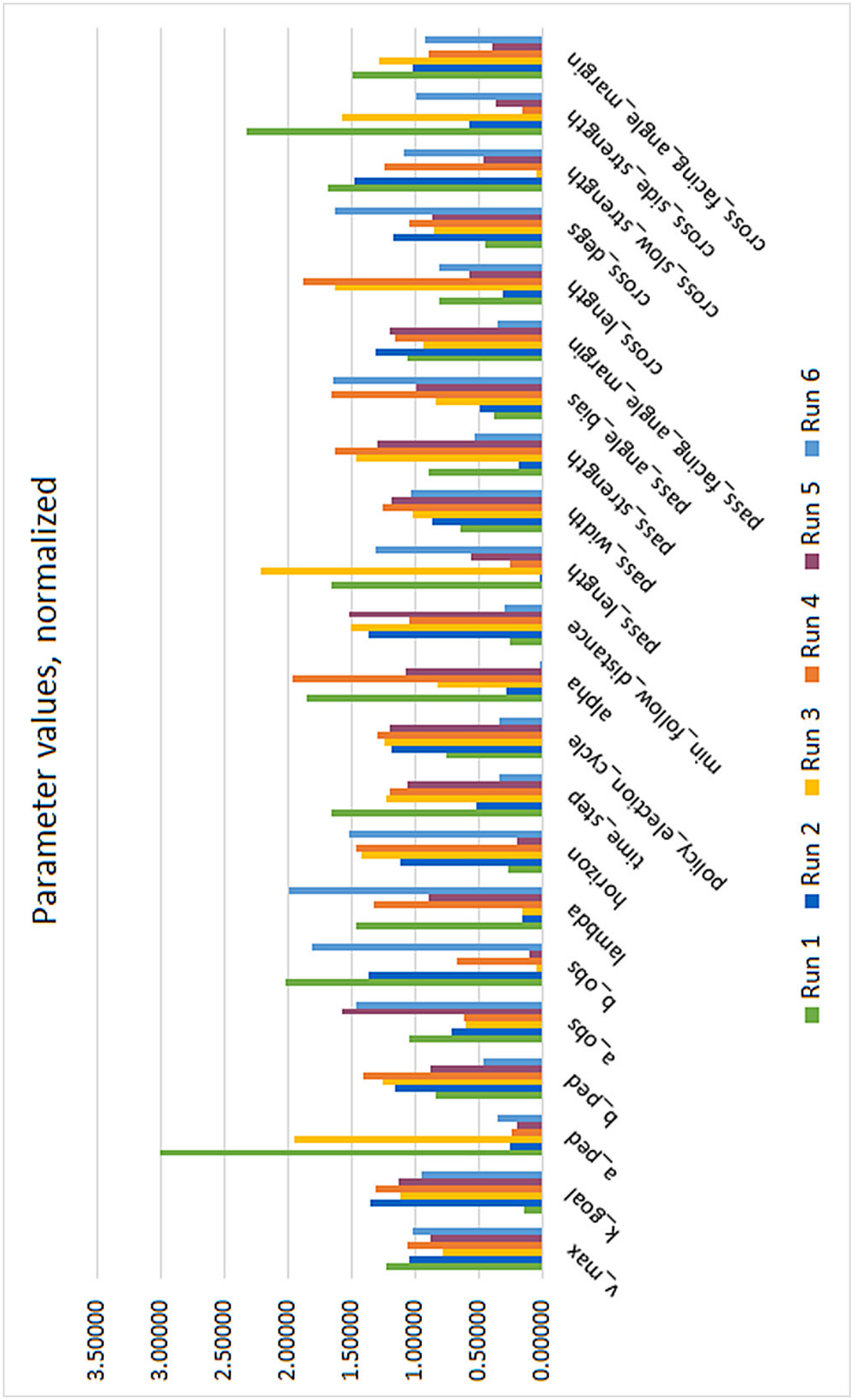

Run 6. Average social forces
Social forces
Generation
social_forces_average


Run 6. Average goal reached
Goal Reached
Generation
goal_reached_average


Run 6. Chromosome diversity 1st Generation, divided by their average
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin


Run 6. Chromosome diversity 30th Generation
90
80
70
60
50
40
30
20
10
0
v_max
k_goal
a_ped
b_ped
a_obs
b_obs
lambda
horizon
time_step
policy_election_cycle
alpha
min_follow_distance
pass_length
pass_width
pass_strength
pass_angle_bias
pass_facing_angle_margin
cross_length
cross_degs
cross_slow_strength
cross_side_strength
cross_facing_angle_margin


Run 6. Chromosome diversity 30th Generation, divided by their average
3.5
3
2.5
2
1.5
1
0.5
0
V max
K_goal
A_ped
B_ped
A_obs
B_obs
Lambda
Horizon
Time step
Policy election cycle
Alpha
Min follow distance
Pass length
Pass width
Pass strength
Pass angle bias
Pass facing angle margin
Cross length
Cross degrees
Cross slow strength
Cross side strength
Cross facing angle margin


Parameter values, normalized
3.50000
3.00000
2.50000
2.00000
1.50000
1.00000
0.50000
0.00000
v_max
k_goal
a_ped
b_ped
a_obs
b_obs
lambda
horizon
time_step
policy_election_cycle
alpha
min_follow_distance
pass_length
pass_width
pass_strength
pass_angle_bias
pass_facing_angle_margin
cross_length
cross_degs
cross_slow_strength
cross_side_strength
cross_facing_angle_margin
Run 1
Run 2
Run 3
Run 4
Run 5
Run 6

O. User test results

Left page, results of the Repeated Measures Anova test. Right page, histograms and means.

Comfort

Pairwise Comparisons

Measure: rating

(I) paramset	(J) paramset	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b] Lower Bound	Upper Bound
1	2	,167	,122	,181	-,081	,414
	3	,560*	,116	,000	,326	,793
2	1	-,167	,122	,181	-,414	,081
	3	,393*	,138	,007	,114	,671
3	1	-,560*	,116	,000	-,793	-,326
	2	-,393*	,138	,007	-,671	-,114

Based on estimated marginal means

*. The mean difference is significant at the ,05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Predictability

Pairwise Comparisons

Measure: rating

(I) paramset	(J) paramset	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b] Lower Bound	Upper Bound
1	2	,202	,174	,251	-,149	,553
	3	,726*	,168	,000	,387	1,065
2	1	-,202	,174	,251	-,553	,149
	3	,524*	,152	,001	,216	,832
3	1	-,726*	,168	,000	-1,065	-,387
	2	-,524*	,152	,001	-,832	-,216

Based on estimated marginal means

*. The mean difference is significant at the ,05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Communication of intent

Pairwise Comparisons

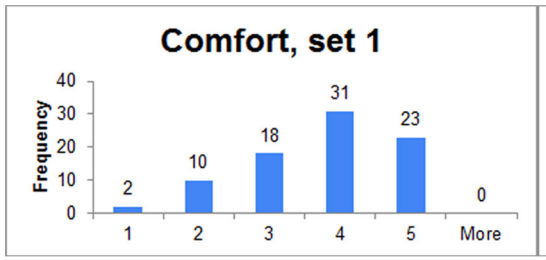
Measure: rating

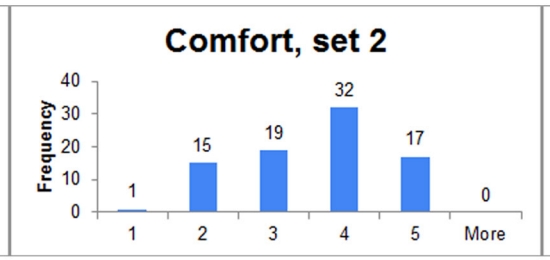
(I) paramset	(J) paramset	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b] Lower Bound	Upper Bound
1	2	,083	,180	,646	-,281	,447
	3	,667*	,137	,000	,390	,943
2	1	-,083	,180	,646	-,447	,281
	3	,583*	,190	,004	,200	,966
3	1	-,667*	,137	,000	-,943	-,390
	2	-,583*	,190	,004	-,966	-,200

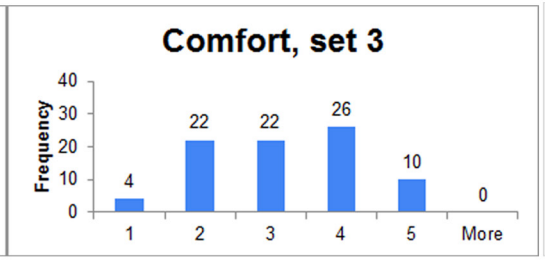
Based on estimated marginal means

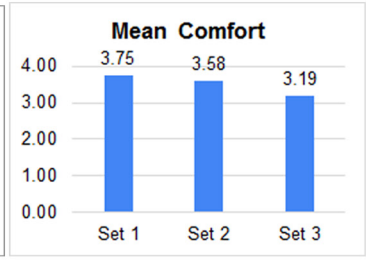
*. The mean difference is significant at the ,05 level.

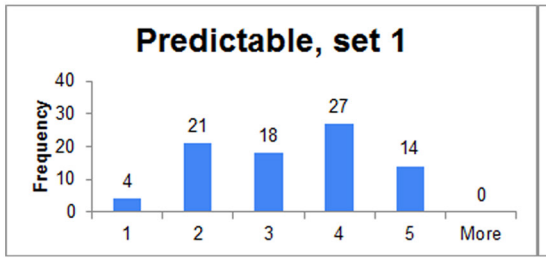
b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

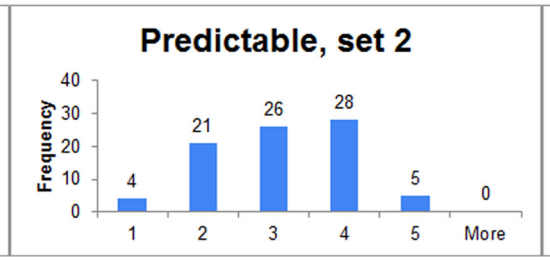

Comfort, set 1
Frequency
40
30
20
10
0
2
10
18
31
23
0
1
2
3
4
5
More

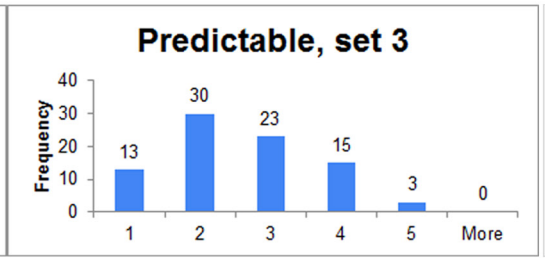

Comfort, set 2
Frequency
40
30
20
10
0
1
15
19
32
17
0
1
2
3
4
5
More

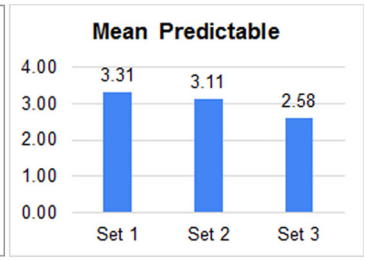

Comfort, set 3
Frequency
40
30
20
10
0
4
22
22
26
10
0
1
2
3
4
5
More

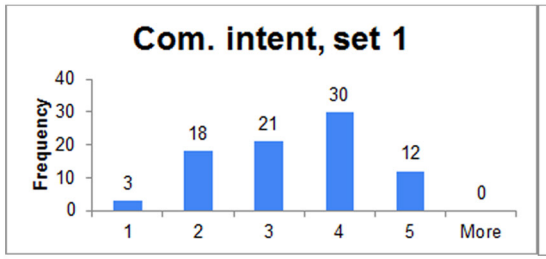

Mean Comfort
4.00
3.00
2.00
1.00
0.00
3.75
3.58
3.19
Set 1
Set 2
Set 3

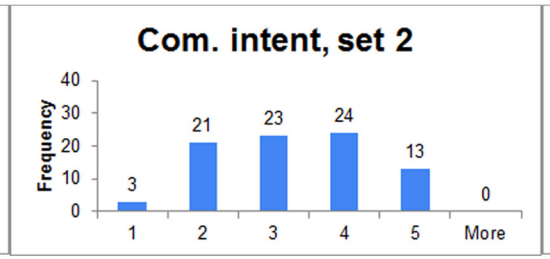

Predictable, set 1
Frequency
40
30
20
10
0
4
21
18
27
14
0
1
2
3
4
5
More

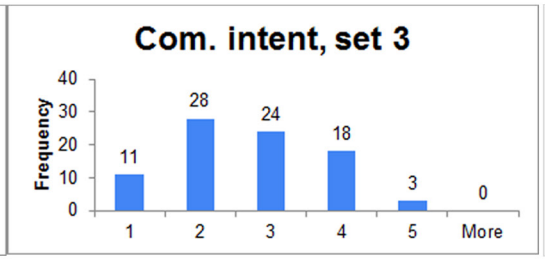

Predictable, set 2
Frequency
40
30
20
10
0
4
21
26
28
5
0
1
2
3
4
5
More

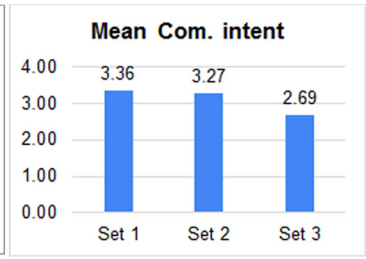

Predictable, set 3
Frequency
40
30
20
10
0
13
30
23
15
3
0
1
2
3
4
5
More

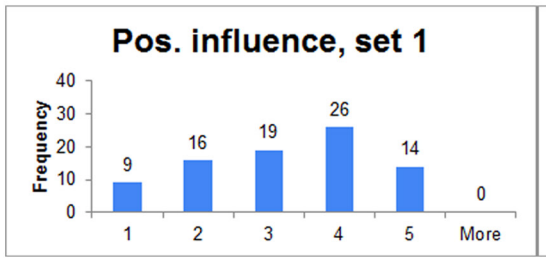

Mean Predictable
4.00
3.00
2.00
1.00
0.00
3.31
3.11
2.58
Set 1
Set 2
Set 3

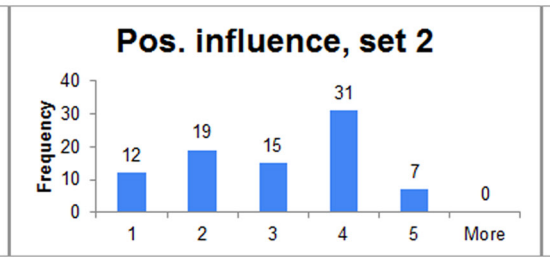

Com. intent, set 1
Frequency
40
30
20
10
0
3
18
21
30
12
0
1
2
3
4
5
More

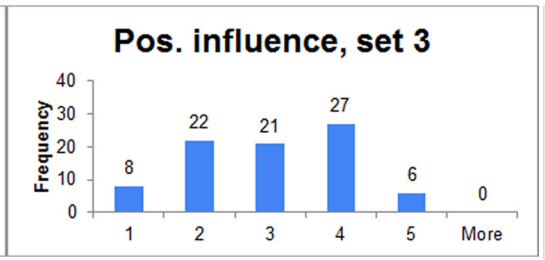

Com. intent, set 2
Frequency
40
30
20
10
0
3
21
23
24
13
0
1
2
3
4
5
More

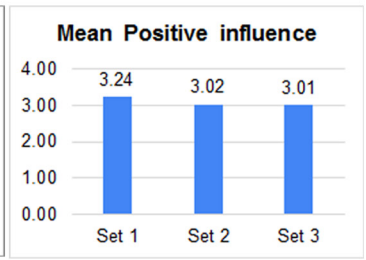

Com. intent, set 3
Frequency
40
30
20
10
0
11
28
24
18
3
0
1
2
3
4
5
More


Mean Com. intent
4.00
3.00
2.00
1.00
0.00
3.36
3.27
2.69
Set 1
Set 2
Set 3


Pos. influence, set 1
Frequency
40
30
20
10
0
9
16
19
26
14
0
1
2
3
4
5
More


Pos. influence, set 2
Frequency
40
30
20
10
0
12
19
15
31
7
0
1
2
3
4
5
More


Pos. influence, set 3
Frequency
40
30
20
10
0
8
22
21
27
6
0
1
2
3
4
5
More


Mean Positive influence
4.00
3.00
2.00
1.00
0.00
3.24
3.02
3.01
Set 1
Set 2
Set 3

Comments as submitted by the participants in the digital form of the user test

1. *Misschien kan 'ie een stukje achter uitrijden nadat je gebotst bent (of wanneer hij bijna botst)*
2. *Maybe it would feel more comfortable for the pedestrians as it moves a bit slower and less 'schokkerig' when it turns in a different direction.*
3. *Robot should give signals, be more interactive so it's more predictable*
4. *This was great.*
5. *In the first run I am too focus looking at the ground. I am notice that there a robot. I still have no suggestion at this moment*
6. *I saw a guy almost hit the robot. I am afraid it will broken. Use handicap maybe*
7. *"Last questions I'd tricky. Positively influenced comparative to which situation? A large braeking area. Lights indicating stops and direction"*
8. *Als robot is gestopt en begint opeens te bewegen als je vlakbij bent is meest wat onzeker gevoel geeft. Wat gaat ie opeens doen. Il denk dat voor soepel gedrag grote mate van voorspelbaarheid belangeijkste is*
9. *Some visual indication, like an LED strip indicating the current direction of motion can help the pedestrians understand the intent of the robot.*
10. *It is best when it goes quite straight so as a human you can take care if it not bumpingin there. Instead of moving unpredictabilly around*
11. *It can give a visual or auditory feedback to signal that it has noticed you*
12. *"It could drive backwards when it comes too close. Now it only stops when you get too close. This way it could feel more like it notices you. Maybe it could keep a larger distance. "*
13. *Looking at the behavior of the robot I did not feel like it was searching for a way, that made its behavior unpredictable*
14. *At run 46 i figured out that the robot was going from c* to c, that changed my perception of its intention and how I looked at him. Afterwards I could better evaluate how it felt to me because I was aware of its goals.*
15. *I think the robot need to be more noticeable(lights to indicate its direction). I noticed that most of us looked down and try to avoid the robot. In fact, pedestrians are not really predictable.*
16. *What is the use of this robot? Im very curious*
17. *As i said, maybe color but also sound (typical robot sound, maybe you could use another sound if thats's possible to make it more natural)*
18. *It does not always see people and can get stuck. Also it is nicer if it moves more continuesly or fluent.*
19. *The robot sometimes stops when I walk behind it*
20. *Light, so you will see the robot better*
21. *Aside from the audio feedback (the sound of the robot) it was very small and could be unnoticed. I could tell it gets to point B from A faster with less traffic infront of it but that may not be the case in real life. The movements of the robot caused chaos with the pedestrians actively trying to avoid it (sometimes people would assume theyll collide with it but it stops and theres a jam in the street)*
22. *Quickly accelerating, not at our pace of walking, so more unpredictable.*
23. *Standing still close to other people feels safe.*
24. *Parkour veranderen Miss geheugen van mens en robot*
25. *Maybe instead of stopping that it slows down but continues to move*

26. *Maybe also add a backwards feature*
27. *Proberen het geluid van de robot te verminderen want hij is wel aanwezig nu*
28. *Nope it is good*
29. *The sound the robot makes makes it easier tk predict its behaviour, which i think is a good thing*
30. *It seems stuck in between the chair. But it is nice robot because it let people go first*
31. *The robot is able to identify me but rather late. That means it will definitely identify me but at a point where it has already come in contact with me (accident).*
32. *Minder snel rijden*
33. *Valt nu niet echt op, zou misschien helpen als hij een andere kleur zou hebben (zodat je er niet op gaat staan bijvoorbeeld)*
34. *Robot makes small turns quite close to you.*
35. *When you walk behind the robot I would expect it accelerates forward to make space for your walk*
36. *I dont really understand the purpose of the robot. From the orher people I know for instance that they are walking from one point to another. The robot is randomly driving around it seems.*
37. *The robot is going too fast and feels like it is intruding in my personal space. I would suggest a more evasive and patient behaviour, and it would be best for it to follow people briefly in their tracks.*
38. *Robots also appears to stop when someone is behind it. Would be better/more fluent if the robot would continue in this case. Sometimes long time at rest without doing anything*
39. *Hard to fill in the answers because the robot was stuck sometimes*
40. *Last question in the survey kind of confusing, the things after the 'note' seem to contradict and make me confused as to what answer to give*
41. *Sometimes the distance the robot took, especially when going towards each other was within a person's personal zone / made me feel like avoiding the robot at that moment*
42. *Misschien parkoer veranderen*
43. *The last 3 runs, the robot was a bit confused and disturbed my pathway of walking. Also, it would be nice if it could indicate when it will drive and how fast in a visual or sound way.*
44. *The noise is very present and not very comforting.*
45. *Since there are no indications on the robot in which direction it goes (Like lights on cars)it is hard to define my own route.*
46. *The behaviour was not really predictable, therefore maybe something like lights could help for example or something you can trust. There was a lot of fluctuations in the runs*
47. *Its hard to know were the robot is when you look ahead when you walk maybe a flag like on bikes of kids would help but maybe it is needed to be small, i dont know. But i would be more comfortable with walking besides it if i have better understamding of where it is*
48. *I did not really meet the robot at this time.*
49. *Dmv licht signalen aangeven wat de robot gaat doen*
50. *Deze ronde was chaotisch, hij blijft vaak hangen bij een muur*
51. *Maybe try to see if you can twist right and left when robot stands still for a wall*

The floorplan for the user test and photos of the actual user test arena as built.