Universiteit Leiden
The Netherlands

Datascience and AI

Evaluating Adaptive and Static Risk-Taking in Robot Navigation on a Dynamic 2D Grid

Tomás Díaz Fiol

Supervisors:
Rob Saunders, Erwin Bakker

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/07/2025

Abstract

Robot navigation in crowded environments is a task that must balance safety and efficiency. This thesis investigates whether agents (robots) using adaptive risk-taking strategies can outperform static risk-taking strategies in dynamic environments across a few standard metrics.

To perform this investigation, a 2D grid simulation was developed containing static obstacles representing walls, and dynamic obstacles representing people. Three static risk-taking strategies, 'safe', 'risky', and 'very risky', with different safety distances were compared against two adaptive risk-taking strategies that adjust risk levels based on crowd density and movement status. Five path-finding algorithms were tested across 100 runs each, using a set of performance metrics.

The results of the experiments revealed some unexpected findings. Firstly, the intermediate 'risky' agent performed worse than both extremes, showing similar collision rates to agents using a 'very risky' strategy, while being slower due to spending more time in crowded areas. Secondly, adaptive risk-taking agents achieved zero collisions but were slower than static approaches. In addition, the adaptive risk-taking agents using 2 levels of risk were faster than those using 3 levels of risk, suggesting that binary risk-taking strategies may outperform approaches that use intermediate values under certain environmental conditions.

These findings contradict assumptions of a linear correlation between risk and performance and indicate that extreme strategies may be more effective than moderate ones in certain contexts. They also suggest that adaptive risk-taking, which has been proven to have performance benefits in certain environments, requires specific conditions in order to yield these results, otherwise, it will perform worse than static risk strategies.

Contents

1 Introduction **1**
- 1.1 Thesis overview . . . 1

2 Related Work **2**
- 2.1 Risk-aware navigation in robotics . . . 2
- 2.2 Social navigation . . . 2
- 2.3 Path-finding algorithms . . . 3
- 2.4 Adaptive risk . . . 3
- 2.5 Performance metrics . . . 4
- 2.6 Current gaps in the literature . . . 4
- 2.7 Adaptive risk-taking . . . 4

3 Methods **5**
- 3.1 Simulation Environment . . . 5
- 3.2 The Robots . . . 5
- 3.3 Risk-Taking . . . 6
  - 3.3.1 Static Risk . . . 6
  - 3.3.2 Adaptive Risk-Taking . . . 7
- 3.4 Path-finding . . . 9

3.5 Movement and Collisions . . . 9
3.6 Performance Metrics . . . 10
3.7 Experiment Design . . . 10

4 Results **11**
4.1 Static risk robots . . . 11
4.1.1 Average Collisions . . . 11
4.1.2 Time to Goal . . . 11
4.1.3 Path Recalculations . . . 12
4.1.4 Distance to People . . . 12
4.1.5 Description of the results for static risks . . . 13
4.2 Adaptive risk robots . . . 13
4.2.1 Average Collisions . . . 13
4.2.2 Time to Goal . . . 14
4.2.3 Risk level used . . . 14
4.2.4 Description of the results for adaptive risks . . . 16

5 Discussion **16**
5.0.1 Connection to the literature and background . . . 16
5.1 Implications and future research . . . 17

6 Conclusion **17**

References **20**

1 Introduction

In robotics, navigating crowded environments is one of the biggest and most studied challenges, including the need to balance safety and efficiency [KPAK13]. This thesis aims to investigate risk-taking in the context of path planning, focusing on comparing static risk and adaptive risk in a simulated 2D environment with both dynamic obstacles, representing people, as well as static obstacles for walls etc. The goal is to assess how being able to change the levels of accepted risk may impact the performance of the agent in the number of collisions and speed.

The motivation for this research comes from real-world scenarios in which robots have to operate in a crowded environment in which humans are also present. In such cases, it is crucial for the robot to keep a safety distance; however, if the distance chosen is too conservative, the robot may be too cautious and stop more than necessary, leading to lower efficiency. Aggressive behavior, on the other hand, may lead to an unsafe environment for people, which is even more undesirable. Keeping this in mind, a robot capable of adapting its behavior to fit the situation it finds itself in could balance safety and speed to attempt to achieve higher efficiency than a conservative agent, but also higher safety than an aggressive one.

1.1 Thesis overview

To compare static and adaptive risk-taking strategies, an agent-based simulation was developed. In addition to simulating agents using static risk-taking strategies, an agent was developed to use an adaptive risk-taking strategy, which dynamically adjusts the risk level based on whether the agent gets stuck, as well as the density of the crowd around it, i.e., how many people are within a certain radius of the agent. The performance of the adaptive risk-taking agent was evaluated using a series of metrics and compared to those of the agents with static risk. Three static risk-taking strategies were used and will be referred to as 'safe', 'risky', and 'very risky'. This thesis aims to answer the following research question:

Can adaptive risk-taking strategies improve the efficiency of an agent navigating a 2D grid while maintaining or improving its safety compared to agents using static risk-taking strategies?

In order to conduct this research, a 2D grid environment was created with static obstacles (walls), and moving obstacles (simulating people). The 'people' in this environment use the A* algorithm to go to their assigned goals. The goals are scattered throughout the grid and are chosen at random for each agent upon reaching their previous goal. This is done to make the agents move in a way that is similar to how a human would move and to avoid undirected or random moves that may be unrealistic. The grid also contains an agent that attempts to reach a fixed goal from a given starting point using on of the following path-planning algorithms: A*, Bi-directional A*, Dijkstra, Breadth-First Search, and Best-First Search. Agents using each of the static risk levels 'safe', 'risky', and 'very risky' were tested. Two agents using adaptive risk-taking were also tested: one that changes between all 3 risk levels and one that only changes between 'safe' and 'very risky'. The key measurements used to compare performance are (1) number of collisions, and (2) speed of traversal.

The following is the structure the thesis will follow:

- Review of related work about risk aware navigation and adaptive risk-taking.
- Description of the setup used for the simulation, the agents, and the path-finding algorithms used.
- Description of the measurements taken and the experiments
- Results obtained
- Discussion on the implications of the results, the possible improvements
- Conclusion regarding the findings from the thesis and discussion on possible future work

2 Related Work

2.1 Risk-aware navigation in robotics

Risk-aware navigation is one of the biggest challenges in robotics, and within that field, there is research that focuses on navigating environments in which humans are present. In robot navigation, risk refers to the probability of colliding given a certain way of acting. For example, a robot that tries to always stay at least 3 meters away from an obstacle is taking less risk than one that allows itself to be within 1 meter. Most of the traditional approaches to robot navigation have often set safety margins that are too conservative, leading to the robot stopping more times than may be necessary, and so making it less efficient than it could be [KPAK13].

The work by Borenstein et al. (1991) on the Vector Field Histogram (VFH) method established some of the principles used for obstacle avoidance [BK91]. It was not based on risk, but it was used to understand how robots can adjust their behavior based on the conditions of their environment to balance safety and efficiency. Their approach also demonstrated that reactive navigation methods could reach reasonable performance, however, it is possible that more sophisticated risk assessment methods could outperform it.

Fox et al. (1997) developed the Dynamic Window Approach (DWA) by building on these foundations [FBT97]. This approach evaluated the potential trajectories of the robot based on several criteria like safety, how direct the path to the goal was, and the speed. DWA was one of the first examples of multiple objective optimization in navigation, however, it weighted the objectives statically and did not use adaptive risk. This was something that limited its efficiency in complex environments with people in them.

2.2 Social navigation

Models of social navigation had a big impact on robot navigation. Helbing and Molnár (1995) introduced the Social Force Model (SFM) as a model that simulates how pedestrians navigate crowded environments [soc95]. It captures the movement patterns of humans and gives insight for

robots that can navigate in a way considered socially acceptable.

Sisbot et al. (2007) expanded on this and developed frameworks for social robots in which the comfort of people and the social rules were kept in mind [hum07]. Their work showed that in order for robots to navigate environments with people naturally, they needed to take into account more variables than collisions, such as how their own movement patterns and presence may affect those of the people around them. With this, we can conclude that in order to have socially acceptable navigation, we can not just focus on the efficiency of the paths chosen.

One of the key concepts we will use is personal space in human-robot interaction, developed by Hall (1966) [Hal66], who proposed that humans have a series of comfort zones around them depending on a series of things, like context, cultural factors, etc. Pacchierotti et al. (2006) then expanded on this idea, explaining that, for robots, it is important to adapt their safety margins depending on the state of their environment and the social context instead of keeping a static safety distance. [pas06]. An example of this could be the following: imagine a robot that acts as a guide in a mall, if the area it is in is empty it has no reason to come close to people, since that would just create uncomfortable situations, however, if the area is crowded and people are packed close to each other, the robot may come closer to them in order to keep moving.

2.3 Path-finding algorithms

Classic path-finding algorithms like A* [HNR68], or Dijkstra have been studied a lot and used in robot navigation. The A* algorithm guarantees computational efficiency, but due to its static nature, it tends to fall short in dynamic environments. A variation of it was created by Stentz (1995): D* (dynamic A*), which allowed the agent to re-plan when the environment changed [Ste95].

Bidirectional search algorithms have a computational advantage, since they search both from the start and from the goal at the same time [Poh71]. Even though these methods have this upside, they still struggle with dynamic environments for the same reason as A*. The combination of bidirectional algorithms and adaptive risk-taking is one that has not had much attention and is relatively new.

Breadth-first search and variations of it, like best-first search, have different trade-offs between optimality and computational efficiency [RN10]). Choosing different search algorithms can affect how the agent responds to changes in the environment, which is a critical part of adaptive risk-taking, however, there is little research on the combination of these algorithms with adaptive-risk taking in dynamic environments.

2.4 Adaptive risk

Robots are nowadays being deployed in more complex environments, and with it, the concept of adaptive risk-taking taking appeared. Althoff et al. (2009) developed methods for probabilistic collision checking that change depending on the conditions of the environment [AKW09]. They demonstrated that, often, using static safety margins can be inefficient.

Chen et al. (2017) worked on adaptive risk taking in crowded environments and developed algorithms that would take into account flow patterns and crowd density in order to adjust their safety margins [CELH17]. They demonstrated that adaptive risk-taking can lead to better results than static risk-taking, specially when crowd densities are changing. However they did not explore the interaction between different path-finding algorithms and different adaptive risk mechanics.

2.5 Performance metrics

Originally, path-finding methods measured performance with a series of metrics such as path length, planning time, and success rate [LaV06], however, since risk-aware navigation makes the problem more complex, other measurements are required in order to track performance accurately.

An example can be seen in Kuderer et al. (2012); a framework for social navigation in which the following things were taken into account: human comfort, path efficiency, and collision avoidance [KKSB12]. Other popular measurements can be collision frequency and severity, for example. The key takeaway from this study, as well as others similar to it, is that balancing multiple objectives is crucial for a risk-adapting agent in order to change its priorities based on the situation.

In stochastic environments, measuring speed and efficiency can sometimes be difficult. Trautman and Krause (2010) developed a series of probabilistic models for human motion prediction in order to make it easier to take these measurements in crowded environments [TK10].

2.6 Current gaps in the literature

Despite the recent work done in this field, there are still some gaps in the literature. For example, most of the current work is either reactive or uses global planning methods. The combination of different path-finding algorithms and different adaptive risk approaches is something that has not been explored extensively in existing literature. Another gap lies in the relationship between crowd density and optimal risk strategies, since we have not fully understood it.

2.7 Adaptive risk-taking

The foundations for adaptive risk-taking are from a series of different disciplines, like decision theory, game theory, and behavioral psychology. The Prospect theory, developed by Kahneman and Tversky (1979), showed how humans make decisions when they are not certain: risk-averse when faced with gains, but risk-seeking when faced with losses [KT79]. This is an example of a theory that could be used to develop navigation strategies similar to those of humans.

Combining theoretical frameworks and navigation algorithms is a topic that is still in development. In this thesis, the aim is to contribute to this development by combining different path-finding algorithms and adaptive risk strategies and comparing them to their static risk counterparts. This approach is inspired partially by the work by Trautman ans Krause [TK10], who highlighted the importance of considering interaction-aware strategies, which motivates the exploration of adaptive risk levels that are capable of responding to changes in the environment

3 Methods

This section describes details of the simulation environment, including the simulation of 'robot' and 'people' agents. It also describes the different path-finding strategies that were implemented and the criteria for varying risk-taking in adaptive 'robot' agents. Finally, it describes how the performance metrics were measured.

3.1 Simulation Environment

The experiments were carried out on a custom-built 2D grid simulation that was made using the pygame library, allowing for visualization. This was done to make the debugging process easier and for demonstration purposes. Figure 1 is a screenshot of a running simulation:

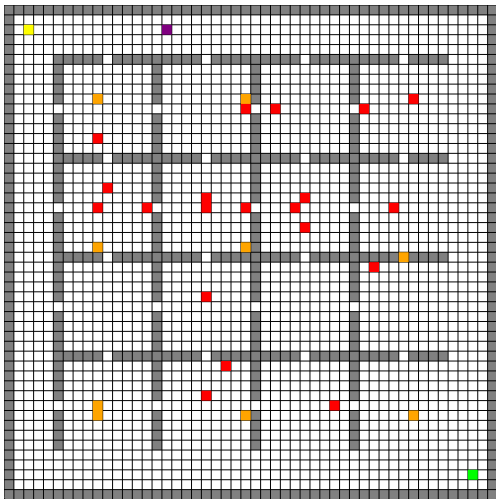

Figure 1: Example of the environment. The gray cells are walls, the purple cell is the 'robot' agent, the red cells are 'people' agents, the yellow cell is the start, the green cell is the robot's goal, the orange cells are the goals between which "people" must navigate. Here is a URL to the pygame library: https://www.pygame.org/docs/

The size of the grid is 50x50, and each cell is either walkable (black) or an obstacle for a robot (gray). Static obstacles consist of the walls shaped in a structure similar to that of a maze. These walls can be edited in the visualization by clicking on them to create and delete them in order to allow us to see how the robot would act under different conditions. The unedited grid, shown in Figure 1, was designed to have areas in which the robot may move more freely, as well as room-like areas that, at times, may get more crowded. This is to ensure the 'robot' agent faces the challenges that it would be expected to overcome in possible real-world scenarios.

3.2 The Robots

The 'robot' agent, or robot, as the agent shall be referred to from now on, is the main focus of the experiments. It is tasked with reaching a fixed goal located in the position (47, 47) of the grid from a fixed starting position at (2, 2) on every individual run. In order to do this, the robot will use a

variety of path-finding algorithms mixed with different risk-taking strategies that will be discussed in the coming sections.
The 'people' agents, which for simplicity will be referred to as people, or a person, for the remainder of the thesis, are simulated as moving obstacles that are present on the grid and that the robot must avoid. When talking about an individual one, in the future, it will be referred to as a person. There is 20 people in each simulation.

People are initialized in random walkable positions of the grid, then they are randomly assigned a goal out of a set of goals made only for people, the 'robot' agent ignores these goals. People use the A* algorithm to find a path to their goal. While a person does not actively avoid other people, their paths are recalculated frequently so that they are rerouted around other people. Every time step, each person updates their position and direction as they follow a path, and recalculates their path if their current one is blocked. In addition, each person checks if they are at their current goal, and if so, they randomly select a new one.

The goal of the implementation of the people is to create dynamic obstacles that move in a manner that resembles human navigation. Around each person, there is a radius R that, with X probability, may become a temporary obstacle for the robot. This X depends on the risk level, which is connected to a variable called $AGENT\_SAFETY\_DISTANCE$. For example, if $X = 50\%$ there is a 50% chance of the positions of that R being considered as obstacles by the robot.

3.3 Risk-Taking

Risk-taking is the main concept of this thesis. In this context, it is connected to the safety distance between the robot and people: the risk accepted by the robot is the distance it tries to maintain from the people around it, higher risk meaning it allows this distance to be lower. The three static risk levels and the two adaptive risk-taking strategies will now be explained.

3.3.1 Static Risk

The three risk levels used were the following, with the explanation of their respective X values below:

- Safe: the safety distance chosen was 6 cells, i.e., $R = 6$.. If at any point the robot comes into contact with the area around a person, due to the movement of the person and not the robot, the robot stops moving until the person has moved far enough that the robot is again outside of this range. $X = 100$.
- Risky: the safety distance is $R = 4$. If the agent ends up inside this range, it has a 50% chance of moving anyway. This is to avoid stopping too often. This, however, may cause collisions, since the agent and the person may move into the same cell. . $X = 70$.
- Very risky: the safety distance is $R = 2$. If the agent ends up inside this range, it has a 95% chance of moving anyway. $X = 10$.

Here is a more detailed explanation on how X works: for very risky X is equivalent to 10%, and so there is a 10% chance of the positions of R being considered as obstacles by the robot when

planing the path and carrying out checks when moving, and so it is more likely for the robot to see a path through this area as walkable. This value is 70% for risky and 100% for safe. These values of X are not to be confused with the probabilities of the agent moving despite being in R. Said probabilities are also shown above (0%, 50% , and 95% , respectively). They allow the agent to move regardless of R being considered an obstacle to give agents higher chances of leaving crowded areas and avoiding getting stuck.
In summary, the safety distance is the area around a person that the robot considers an obstacle when choosing a path. The robot can, however, stochastically ignore these obstacles with different probabilities depending on the risk level. This is to allow the robots to more easily exit crowded areas.

3.3.2 Adaptive Risk-Taking

The two types of adaptive risk-taking robots were tested:

- Three levels of risk: this type of robot changes its risk level among the three static risk levels mentioned above.
- Two levels of risk: this type of robot changes its risk level among only two of the previous static risk levels: safe and very risky.

Both types would change their risk levels following the same logic. A record of the robot's moves are kept, as well as the density of people within a radius of 5 cells of the robot. If the robot does not move for a certain period of time, it considers itself stuck, and to avoid pausing for longer, it increases the accepted risk level. The same would be the case if the density around it were to increase, or, in other words, there were too many people around it. Once the robot is moving freely and the density around it has decreased, the accepted risk is decreased. Algorithm 1 provides the corresponding pseudocode for managing risk level.

Algorithm 1: Update Risk Level Based on Density and Movement (3 risk levels)

Input: Robot position $\mathbf{p}_{robot}$, list of persons $\mathcal{P}$, current time t
Output: Updated safety distance d

if *last position is not initialized* **then**
  last_pos $\leftarrow \mathbf{p}_{robot}$
  last_time $\leftarrow t$
  return current_safety_distance
end
if *last_pos* $\neq \mathbf{p}_{robot}$ **then**
  successful_move_count $\leftarrow$ successful_move_count + 1
  stuck_time $\leftarrow 0$
  if *successful_move_count* $\geq$ *threshold* **then**
    is_stuck $\leftarrow$ False
  end
else
  stuck_time $\leftarrow$ stuck_time + (t $-$ last_time)
  successful_move_count $\leftarrow 0$
  if *stuck_time* $\geq$ *stuck_threshold and not is_stuck* **then**
    is_stuck $\leftarrow$ True
  end
end
last_pos $\leftarrow \mathbf{p}_{robot}$
last_time $\leftarrow t$
density $\leftarrow$ number of persons within radius r of $\mathbf{p}_{robot}$
if *density* $\geq$ *high threshold* **then**
  $d \leftarrow$ minimum safety distance (very risky)
else
  if *density* $=$ *medium threshold* ***or*** *is_stuck* **then**
    $d \leftarrow$ medium safety distance (risky)
  else
    $d \leftarrow$ maximum safety distance (safe)
  end
end
return d

In the case of the robot using two risk levels, the safety distance is lowered to a minimum when it is stuck or the density rises, and goes back up when the state of the environment allows it.

This logic aims to prevent excessive stopping caused by crowds of people, but also prevents the robot from getting too close to people in situations where it is not necessary. This way, the robot will maintain socially acceptable behavior by keeping a distance when possible and getting close when needed.

3.4 Path-finding

The robot employs different path-finding algorithms to select the path it can take to the goal. If a path becomes unavailable for some reason, the robot selects a new path using the same algorithm. Five popular path-finding algorithms were implemented using the pathfinding library:

- A*
- Dijkstra's Algorithm
- Bi-directional A*
- Breadth-First Search (BFS)
- Best-First Search

Diagonal movement was disabled for all algorithms. Consequently, diagonal movement was also disabled for the people because they shared the same A* implementation.

Separate simulations were run for each combination of path-finding algorithms and risk-taking strategies.

Paths would be calculated every 0.5 seconds to acknowledge changes in accepted risk levels as well as changes in the environment, such as the position of the people on the grid. The robot is allowed to move every 0.1 seconds as long as the position it is moving to is safe. The people, on the other hand, move every 0.2 seconds.

In some cases, no path would be found because of the position of the people, in these cases the robot would freeze. To avoid this, a timeout of 1 second was set, and the run would be skipped. Since the people move every 0.2 seconds, 1 second was considered enough time for them to move in such a way that a path may be found.

3.5 Movement and Collisions

At each step, the robot tries to move to the next position in its path, but before doing so, it checks that the new position is safe according to the current state of the environment, according to the risk acceptance that is active at the moment. If this safety condition is not met, then the robot will act according to its risk-taking strategy, i.e., either using a static risk level or an adaptive strategy.

3.6 Performance Metrics

Each simulation run was evaluated using the following set of performance metrics:

- Time to goal: total time taken by the robot to reach the goal.
- Average safety distance (this measurement applies only to adaptive robots): amount of time each safety distance was used on average throughout the run.
- Collisions: number of proximity violations (following the rules explained above).
- Average distance to people: average Euclidean distance from the robot to all people.
- Number of path recalculations: number of times the robot had to change the path it came up with.

Collisions are tracked depending on proximity and risk level:

- For 'very risky' robots, collisions were counted when the robot was one position away (not counting diagonal moves).
- For 'risky' robots, collisions were also counted when the robot was one position away (not counting diagonal moves).
- For 'safe' robots, collisions were counted when the positions were the same.

The reason behind for the difference in counting collisions is that when it comes to social navigation, it is not seen as acceptable to speed past a person while leaving such little distance. Since the safe robot stops when close, people would not be startled by it, however, the other two types of robots would keep moving. This way we count as collisions not only when the positions are the same, but also when the robot gets into a situation that would startle people in real life.

In the case of adaptive risk-taking robots, the logic used for collisions was dependent on the accepted risk level that was being used.

Euclidean distance was used to measure the distance between the 'robot' agent and other objects. In order to make it easier to understand and work with, the thresholds of the Euclidean distance required to count collisions were changed to act as described above. This allowed the thresholds to act in intuitive ways.

3.7 Experiment Design

The experiments were divided into three phases. First, for each of the static risk levels, the different path-finding algorithms were tested over 100 runs using the same environmental setup. Then the adaptive risk robot with three risk levels was tested, also over 100 runs and using the same path-finding algorithms, and finally the same was done for the adaptive risk robot with two risk levels. After each phase, a series of plots were produced with a series of metrics. At a later time, we will go over these plots to see what the outcome of the experiments was and what conclusions we may draw from this.

4 Results

In this section, the results obtained from the simulations will be shared. These results will be interpreted and discussed in the section that follows this one.

4.1 Static risk robots

Using the experiments described in the Experiment Design section, static risk 'robot' agents were tested for the following: average number of collisions, average time to goal, average path recalculation,s and average distance to people. The combinations of the different risk levels and algorithms were measured and will be shown below.

4.1.1 Average Collisions

The number of collisions is used to see how often the 'robot' agent collides with people. This is something that should be avoided, and so the lower the number of collisions, the better. As seen below, in Table 1, safe had no collisions, meanwhile risky and very risky had similar numbers of collisions on average.

Risk Level	**A***	**Dijkstra**	**Bidirectional A***	**Breadth-First Search**	**Best-First**
Safe	0.00	0.00	0.00	0.00	0.00
Risky	1.29	1.16	**0.59**	1.19	0.81
Very Risky	1.25	1.12	**0.99**	1.21	1.07

Table 1: Average number of collisions across algorithms for the static risk levels

4.1.2 Time to Goal

The time to goal describes how fast the 'robot' agent gets from the start to the goal. This is something that should be minimized to be as fast as possible. In the Table 2 it can be seen that very risky is the fastest, followed by risky, and finally by safe.

Risk Level	A*	Dijkstra	Bidirectional A*	Breadth-First Search	Best-First
Safe	15.00	**14.46**	15.75	15.36	14.78
Risky	12.22	12.32	12.12	12.21	**11.78**
Very Risky	**9.69**	9.79	9.78	9.74	9.70

Table 2: Average time taken by the robot to reach the goal across the different algorithms for the static risk levels.

4.1.3 Path Recalculations

Path recalculations measure how often the robot had to change the path it came up with. This is something that should be balanced, as having to recalculate the path leads to longer times to get to the goal, but not recalculating it could mean taking paths that may lead to a lot of collisions. Because of this, a different balance is required for different strategies. The very risky 'robot' agent does not need to recalculate its path as often because it accepts routes with higher probabilities than the other two risks, as seen in Table 3.

Risk Level	A*	Dijkstra	Bidirectional A*	Breadth-First Search	Best-First
Safe	20.22	19.89	20.23	20.34	20.82
Risky	20.58	20.89	20.85	20.52	20.47
Very Risky	18.51	18.72	18.55	18.58	18.53

Table 3: Average number of times the robot changed its calculated path in a run across the different algorithms for the static risk levels.

4.1.4 Distance to People

The distance to people shows how far the 'robot' agents are on average from the people on the grid. This helps understand the amount of time the 'robot' agents spend near people. In Table 4 it can

be seen risky has the lowest distance, meaning it spends the most time near people.

Risk Level	A*	Dijkstra	Bidirectional A*	Breadth-First Search	Best-First
Safe	27.90	28.69	28.09	28.34	28.47
Risky	25.98	25.90	26.63	26.32	27.16
Very Risky	27.97	27.94	27.97	**27.92**	28.19

Table 4: Average distance to the people in the grid across the different algorithms for the static risk levels.

4.1.5 Description of the results for static risks

The risky robot, as seen by looking at Table 1, had a higher number of collisions on average than the safe robot and a similar number to the very risky, with some algorithms even surpassing it. When it comes to the time taken to reach the goal, if we look at Table 2, we see that while it is faster than the safe robot, it is slower than the very risky robot.

4.2 Adaptive risk robots

Once again, the setup from the Experiment Design section will be used. In this case, adaptive risk 'robot' agents were tested for the following: average number of collisions, average time to goal, and time spent using each risk level. The combinations of the different strategies, 2 levels and 3 levels of risk, and algorithms were measured and will be shown below.

4.2.1 Average Collisions

Like before, the number of collisions is used to see how often the 'robot' agent collides with people. This is something that should be avoided, and so the lower the number of collisions, the better. As it can be seen in Table 5, both strategies averaged 0 collisions.

Risk Level	A*	Dijkstra	Bidirectional A*	Breadth-First Search	Best-First
2 levels	0	0	0	0	0
3 levels	0	0	0	0	0

Table 5: Average number of collisions across the different algorithms for the adaptive robots.

4.2.2 Time to Goal

The time to goal, again, describes how fast the 'robot' agent gets from the start to the goal. This is something that should be minimized. In the Table 6 it can be seen that using 2 levels of risk leads to better results.

Risk Level	A*	Dijkstra	Bidirectional A*	Breadth-First Search	Best-First
2 levels	**17.8**	18.8	18.7	**17.8**	19.6
3 levels	23.3	**22.8**	**22.8**	23.0	27.2

Table 6: Average time taken to reach the goal across the different algorithms for the adaptive robots.

4.2.3 Risk level used

The following tables show the average time each risk level was used per run for the two different adaptive strategies. This gives information on which risk levels were accepted more often and which were accepted less often for the different adaptive risk strategies.

In Table 7 it can be seen that the Safe distance was used the most. In Table 8 it can be seen that safe was again used the most, followed by risky, and then very risky.

Risk Level	A*	Dijkstra	Bidirectional A*	Breadth-First Search	Best-First
Safe	12.0	12.4	12.3	11.7	12.7
Very Risky	5.2	6.0	6.2	6.1	6.2

Table 7: Average time using the different safety distances across the different algorithms for the adaptive risk strategy with 2 risk levels.

Risk Level	A*	Dijkstra	Bidirectional A*	Breadth-First Search	Best-First
Safe	14.1	13.5	13.3	14.1	15.5
Risky	8.4	8.4	8.2	8.3	10.2
Very Risky	1.14	0.90	1.19	0.93	1.16

Table 8: Average time using the different safety distances across the different algorithms for the adaptive risk strategy with 3 risk levels.

4.2.4 Description of the results for adaptive risks

Given the results seen while studying static risk, it was decided that implementing two different adaptive risk robots to further study these findings would be needed, one that used all three risk levels and one that used only safe and very risky. Both of them were capable of safely navigating the environment, as shown by Table 5; however, the robots using only 2 levels managed to do so faster (see Table 6).

5 Discussion

The theory that adaptive risk-taking could outperform static risk approaches could not be proven. While Table 5 shows that the average number of collisions per run for adaptive risk robots was 0, Table 6 shows higher times to reach the goal than Table 2.

The following section will go over some of the key findings of this thesis, including the ones required to answer the research question, as well as unanticipated discoveries.

The first key finding was one that was discovered unintentionally while trying to measure the performance of the static risk levels. The risky robot is slower than the very risky robot, and it does not see any safety benefits when compared to it, as seen in the Description of the results for static risks. This goes against the hypothesis that was originally formulated, in which a balance between safety and speed would make it perform safer than very risky in exchange for a slower speed.

In order to find an explanation to this behavior, we will look at Table 4. In it, we see that on average, the distance between the 'robot' agent and the people is lower for the risky robots. This means that it tends to spend more time in crowded areas. The safe robot tends to stop before entering crowded areas or avoids them if possible, the very risky robot is more willing to go into crowded areas, but also exits them faster, meanwhile the risky robot enters them but is incapable of exiting them as fast, causing it to spend more time in areas with higher density, and so making it more probable for it to collide.

The second key finding came from comparing the static risk robots and the adaptive risk robots. As mentioned above, the adaptive risk robots demonstrated the capability to safely navigate the environment; however, they were slower than the static risk robots. This, however, does not mean these findings can be generalized to all adaptive risk applications. Prior research, e.g., [CELH17, AKW09], showed that adaptive strategies can outperform static ones in more complex or uncertain environments.

5.0.1 Connection to the literature and background

Previous research, e.g., [CELH17], showed adaptive risk navigation to perform better than its static counterpart; this thesis has not found evidence to support or refute this claim. This is because of the nature of the simulation more than the concept of adaptive risk itself, however, the research presented here raises some interesting questions about the specific contexts where adaptive

risk-taking performs better.

Kuderer et al. (2012) expressed the need to balance between multiple objectives as a crucial part of adaptive risk-taking [KKSB12], however, the trade-off between efficiency and safety used for this experiment was not complex enough to produce benefits in performance in comparison to static risk.
Certain conditions must be met in order to see the benefits of adaptive taking. Over-simplified environments and interactions may cause the results to contradict the previous findings. Given this non-continuous environment with simple interactions between elements, we see that this is indeed the case. The risk levels chosen and the probability cutoffs were also not optimized and so may not generalize across environments. The behavior of people in the environment was also simplified. They do not form groups, react to other people, or react to the robot, etc. This on the other hand, is something we would expect people to do in real life scenarios.

5.1 Implications and future research

The results found suggest that, for some settings, binary risk strategies may outperform continuous ones in which the risk level may vary along a spectrum, suggesting the trade-off between risk and performance may not be linear. This is because the extremes may perform better than intermediate values. It can not be said under which specific conditions this will be true, but further research on the topic may be able to identify them and compare binary and continuous strategies under a variety of conditions.

Finally, it was also seen that the results were similar regardless of which navigation algorithm was used, allowing us to conclude that risk strategies will show similar behavior no matter the navigation algorithm used.

6 Conclusion

The main question this thesis tried to answer was the following: Can adaptive risk-taking improve the efficiency of a robot navigating a 2D grid while maintaining or improving its safety compared to robots using static risk levels?

Based on the results, it can be said that for the environment used, adaptive risk-taking did not improve performance compared to static approaches. While adaptive robots did avoid collisions, they were slower than the static robots.

A key finding was that the intermediate risk level, which was expected to balance caution and speed, actually performed worse than the extremes. It had similar collision rates to very risky robots and longer goal times. This was likely due to the robot getting stuck in crowded areas without either stopping or pushing through quickly.

This led to the discovery that binary risk strategies (safe and very risky only) might be more effective under certain conditions than continuous or multi-level approaches. However, these results

cannot be generalized. The simplicity of the environment and the limited interactions between robots and people affect how the different approaches perform.

In conclusion, while this thesis did not prove the superiority of adaptive risk-taking, it revealed that intermediate strategies may not always be the most effective. These findings suggest the need for future research into when adaptive risk can be most useful and how to design environments and robots that benefit from it.

References

[AKW09] D. Althoff, B. Kluge, and D. Wollherr. Safety assessment of driving behavior in multi-agent scenarios. In *IEEE International Conference on Intelligent Robots and Systems*, 2009.

[BK91] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[CELH17] Y. F. Chen, M. Everett, M. Liu, and J. P. How. Socially aware motion planning with deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[FBT97] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[Hal66] Edward T. Hall. *The Hidden Dimension*. Doubleday, Garden City, NY, 1966.

[HNR68] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[hum07] A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, November 2007.

[KKSB12] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: Science and Systems*, 2012.

[KPAK13] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[KT79] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979.

[LaV06] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, UK, 2006. Also available at http://planning.cs.uiuc.edu/.

[pas06] Evaluation of passing distance for social robots. In *15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2006)*. IEEE, October 2006.

[Poh71] I. Pohl. Bi-directional search. In *Machine Intelligence 6*, pages 127–140. Edinburgh University Press, Edinburgh, 1971.

[RN10] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2010.

[soc95] Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282, 1995.

[Ste95] Anthony (Tony) Stentz. The focussed d* algorithm for real-time replanning. In *Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI '95)*, page 1652 – 1659, August 1995.

[TK10] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.