DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2022

Simulating crowds of pedestrians using vector fields and rule-based deviations

Master's Thesis Project

Filip Berendt

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Author

Filip Berendt, berendt@kth.se
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Place for Project

The Visualization Studio
Stockholm, Sweden

Examiner

Tino Weinkauf
Division of Computational Science and Technology
KTH Royal Institute of Technology

Supervisor

Björn Thuresson

Division of Computational Science and Technology

KTH Royal Institute of Technology

Abstract

In the area of steering behaviours of autonomous agents and crowd simulations, there is a plethora of methods for executing the simulations. A very hard-to-achieve goal of crowd simulations is to make them seem natural and accurately reflect real-life crowds. A very important criterion for this goal is to have the agents avoid collisions, both with each other and with the environment. A less important, but important nonetheless, criterion is to not let the time taken or distance covered to reach the goal in the simulation be too high, compared with when not implementing collision avoidance. This paper proposes and explores a novel method of enhancing vector field-based steering with rule-based deviations to implement collision avoidance. This method is called 'DevVec' ('Deviation + Vector Field steering'). The rules which are used for the deviations are extracted from a user survey, and they describe what the agent should do in different collision avoidance scenarios. The viability of DevVec is tested by comparing it with another already established method, called 'Gradient-based Steering', in terms of fulfilling the criteria mentioned above. Both methods are used to simulate pedestrians moving throughout different scenes. The results suggest that DevVec has potential, but would require additional time and resources, and perhaps a few changes in future works to be presented in its best possible version.

Keywords

Steering Behaviour, Autonomous Agents, Crowd Simulation, Vector Field, Deviation, Gradient, User Survey, Unity3D

Abstract

Inom ämnesområdet för styrbeteenden hos autonoma agenter och simuleringar av folkmassor finns det många metoder för att framställa dessa simuleringar. Ett väldigt svåruppnåeligt mål för denna typ av simuleringar är få dem att verka naturliga och verklighetstrogna. Ett viktigt kriterie för detta mål är att få agenterna att undvika kollisioner, både med varandra och med den kringliggande omgivningen. Ett mindre viktigt, men viktigt oavsett, kriterie är att inte låta en agent ta för lång tid eller gå för långt för att nå sitt mål i simuleringen, i jämförelse med när de inte försöka undvika hinder. Denna studie presenterar och utforskar en ny metod som utökar en vektorfältsbaserat styralgoritm med regelbaserade avvikelser för att ta hänsyn till att undvika kollisioner. Denna nya metod kallas för 'DevVec' ('Deviation + Vector Field steering'). Reglerna som används för avvikelserna är framtagna från en enkät, och de beskriver vad en agent borde göra vid olika kollision-scenarion. Användbarheten av DevVec prövas genom att jämföra den med en redan etablerad metod som kallas för 'Gradientbaserad styrning', med avseende på de ovan nämnda kriterierna. Båda metoderna används för att simulera fotgängare i olika omgivningar. Resultaten antyder att DevVec har potential, men att det krävs ytterligare tid och resurser, och troligtvis några ändringar i framtiden för att framställa den bästa möjliga versionen.

Nyckelord

Styrbeteenden, Autonoma Agenter, Folkmassesimuleringar, Vektorfält, Avvikelser, Gradient, Användarenkät, Unity3D

Contents

Chapter 1

Introduction

This paper explores a novel method for steering behaviours in crowd simulations, combining vector field following and collision avoidance deviation[9]. The rules for the deviations are constructed from a survey where the respondents evaluate collision-risking scenarios identified in a previous paper[10]. It is also compared with an already established and very recent method[5], to determine its viability. In this paper, these methods will be referred to as ”DevVec” and ”Gradient” methods respectively.

1.1 Background

Crowd simulations are methods for generating behaviour for autonomous agents, specifically for simulating crowds. An autonomous agent can be either physical or virtual, but the core ideas for steering these agents are the same regardless which type of environments the agents act within. These agents are moving throughout the environment towards a goal, using a set of rules to follow, given some sensory inputs. Some may argue that every living creature capable of moving are autonomous agents as well, because when they move, they have a goal, sensors which feed information, and rules for e.g. avoiding a tree. Because of this, it could be inferred that it is possible to design agents which accurately simulate human behaviour.

The use of simulating crowds is to increase immersion within virtual environments[15], and if that environment also employs some game mechanics, it may provide some means of fulfilling the intended game design of the game. A few examples of games that could use more realistic crowd simulations are ”Spy Party”, ”Total War”-series, and ”Age of Empires/Mythology”-series[1, 11, 12]. For these games, smarter crowd simulation could prevent agents forcing themselves against obstacles and/or introduce a strategic element regarding navigation.

1.2 Challenges

Simulating complex behaviours, such as humans moving in a crowd is difficult, both implementation- and computation-wise[15]. Therefore, efforts have been made to make both aspects of crowd simulations more viable for practical use. There is the very simple idea of having autonomous agents follow a vector field, but that doesn’t account for avoiding collisions. There are also methods for avoiding collisions, but that is not a complete set of steering behaviours.[9] This paper documents the attempt for a novel idea to combine vector field following with simple collision avoidance, by setting natural rules for an agent to deviate from its intended path towards the goal, as opposed to attempting to get mathematical optimisations to act ”natural” in this context. An example of a ”natural” rule is for an agent to turn left if it’s expecting a frontal collision with another agent. The intended path is supposed to take the agent straight to the goal, if no collision risks are presented.

1.3 Research Question

What is the viability of a steering algorithm implemented using vector field following and collision avoidance deviations (DevVec), and how does it compare to established, more mathematical methods (Gradient)?

To clarify the term ”viability” in this case, a viable method is determined in some manner to have a applicable use, that can compete with other viable options. For a method to be viable, it has to offer similar benefits as others and/or be found superior in one or several aspects to compensate for any shortcomings.

1.4 Purpose

The aim for the study presented in this paper is to explore the possibilities of crowd simulations, and to contribute to the field of navigation of autonomous agents. Regardless of the efficacy of the method presented, it is sure to give some insights to the community of the field.

1.5 Delimitations

This study only concerns itself with the steering behaviour of agents acting as human pedestrians in a few simple, pedestrian-only environments. It does not concern itself with emergent behaviour of said pedestrians carrying burdens, pushing on strollers, etc.

The study is not a perception study, and therefore does *not* focus on the perceived realism of the crowd simulations, although there will be a few comments regarding that in the discussion.

1.6 Outline

In chapter 2, more details are provided for each of the related works. A study that precedes this one is also briefly presented and explained.

In chapter 3, the methods used for conducting this study is presented as they have been applied. There are also some mid-way results presented here, that are used as part of the implementation.

In chapter 4, the results meant to answer the research question are presented, and context is provided when necessary.

In the final chapter 5, the results are interpreted to determine the conclusion to the research question. It will also contain some speculation for what could be done in future works.

Chapter 2

Background

The topic of crowd simulation is not a particularly new one. There is a plethora of previous work done within the area, as well as areas closely connected to it. This study is based on a simpler pre-study done a few months earlier, and it is compared to an established algorithm for steering. The steering behaviours used and different collision scenarios to identify are simple to identify given enough thought, but are nevertheless documented within previous studies. It will also need to have some form of objective evaluation, which is also a study done previously.

2.1 Pre-study

This study is based on a pre-study[2]. The idea, results, and discussions can be inferred from the slides used for the presentation. The other papers referenced later in this chapter were directly involved in the execution of this one. The pre-study involved a survey for people to send data on how to deal with collision scenarios, which was then used for constructing the rules of the algorithm. The research question here was to determine which vector blending method was best fitted to blend avoidance vectors (explained further in chapter 3) for this kind of crowd simulation. The results of the study suggested that priority-based blending was best fitting. Although the results were later discovered to be flawed, the conclusion was nevertheless the same with corrections to the implementation afterwards.

The environment used has been repurposed for this study. The survey has also been improved and repurposed for this study.

2.2 Related Work

Proposing a novel (as of writing) method for steering behaviour requires a few prerequisites. First off, the method has to fulfil some kind of need, or additionally propose a novel need unknown of beforehand. For this particular case, it fulfils an already established need. The method for steering autonomous agents is a congregation of simpler algorithms. These need to be documented somewhere, to provide confidence that these simpler algorithms actually do what they are supposed to do. The agents also need some kind of sensory input for which to react to. These inputs are 'scenarios' involving the potential obstacles and the agent itself. For each scenario, there is at most a handful of appropriate actions. These scenarios preferably needs to be documented somewhere, again to provide confidence that they are relevant to the execution of the method.

Because the study proposes a new method and its viability, it is appropriate that it is tested against an established, and preferably recent, method. If it manages to hold up to the contender in most areas, the confidence in its viability would increase. This contender method needs to be used for similar purposes for the comparison to be valid.

Finally, to pair up different methods to determine the viability of one of them requires some manner of evaluating their performances. This can be done either by having observers watch them and then determine evaluation based on some criteria or by measuring tangible events during the simulation.

In this section, previous external studies will be presented and how they are relevant for fulfilling the prerequisites mentioned above.

2.2.1 Crowd Simulations and its applications

In the study by Xu Ming Liang et al.[15] different kinds of crowd simulations are surveyed, as well as the motivations for using each. Among those, pedestrian crowd simulations, specifically using vector fields are surveyed. Applying vector field following is a macroscopic method for crowd simulations; it provides basis for the entirety of the agent group to move throughout the simulation. It provides the main inspiration for the study, as well as justification for the whole concept of crowd simulations, although only to the extent of increasing immersion. They do not consider collision avoidance scenarios for any of these cases, which provides a basis for applying such measures in this study.

2.2.2 Steering behaviours for autonomous agents

Craig Reynolds presents several simple steering behaviours and how to practically implement them[9]. Specifically important behaviours for this study are seek/flee, obstacle avoidance, and flow field following (i.e. *vector* field following). Seeking and fleeing a target are the opposite actions based on the same data. It means for an agent moving forward to rotate towards or away from a specified target, respectively. Obstacle avoidance is similar to fleeing, as it attempts to rotate away from an obstacle target, but stops doing so once the obstacle no longer threatens collision, according to some set thresholds. Flow field following is simply having the agent follow an abstract representation of a flow, which is made possible with a discrete vector field using bilinear interpolation.

It was published 1999 and has (as of writing) 2068 citations. The very early publishing date is testament to the general ubiquity of the methods he introduces.

2.2.3 Autonomous pedestrians

Wei and Terzepoulos make an attempt at crowd simulation of pedestrians in a virtual rendition of "Penn Station"[10] in New York City, where each pedestrian navigates the station towards its individual goal, while also avoiding collision with others. The scene of Penn Station is quite complex, with several individual regions, and the simulation has to make use of pathfinding to determine series of subgoals a pedestrian needs to reach to ultimately reach their final goal.

The main take-away from their study however, is the identification of different collision avoidance scenarios, which inspires the survey questions used for this study. They also provide the basis for the scenarios identified and acted upon during the simulation. These scenarios are explained in further detail in chapter 3.

2.2.4 Gradient-based steering for vision-based crowd simulation algorithms

Because of the novelty of the DevVec-method, it is reasonable to compare the performance of it relative to other reportedly good methods. While there are many studies exploring steering algorithms that can be considered successful, they don't necessarily consider using them to simulate realistic scenarios, which would include crowd simulation. Their efficacy for crowd simulation is therefore untested, and it wouldn't make sense to compare with them. Those who have tested for crowd simulation, or similar topics, are better candidates.

One such study, and a quite recent one, is the study done by Dutra et al.[5] They had identified the issue of previous methods leaving unwanted behavioural artifacts, specifically those that often made simulation look quite unrealistic. The two kinds of artifact highlighted were either that the algorithms fails to adequately assess future collision threats or they make agents move in realistically abnormal ways. They wanted to address these issues by introducing a gradient-based method, which would essentially combine the strengths of various strategies and hopefully eliminate the weaknesses. A very abstract way to describe how it works is that it goes through a loop continuously of three phases: Perception, evaluation, and action. The perception phase constructs an image using ray tracing of what the agent sees. The evaluation phase evaluates the obstacles seen on the image, as well as the agent's own orientation towards its goal, and generates associated evaluation metrics. The action phase uses the evaluation to calculate partial derivatives to construct a gradient, which is then applied to the agent's current speed and rotation.

The idea was sound, and the results were as expected; good. Because of their success and the relative recency of their work, this study will use their algorithm to compare with. In some areas of their algorithm, it used normal distributions to influence the behaviour, which had constant standard deviations (sigma values) as settings specified by the user. The ones they considered to be optimal for their study are not necessarily so for this one, so they need to be taken to account.

2.2.5 Scenario space: characterizing coverage, quality, and failure of steering algorithms

While comparing the method of this study with established methods is important, it is also important to do evaluate objective metrics, if possible. Kapadia et al. conducted a study to determine the viability of their proposed evaluation method[8]. Their method involved measuring whether or not a collision occurred for an agent, how much time it took to reach its goal compared to a calculated optimum, and how much distance it covered compared to a calculated optimum. These metrics can be measured accurately on both the comparison and this study's method, to do an objective comparison between the two. The metrics were then combined to generate the score of an algorithm over a vast array of different scenarios. The results were that their method could accurately rank different evaluation algorithms according to the perceived rankings done without this method.

They considered a scenario to be a set of agents trying to reach a goal, and that it was finished once a collision had occurred, once all agent had reached their goals successfully, or if the scenario timed out. Also, the evaluation was done for all agents in the scenario collectively. In this study however, their evaluation method has been altered slightly to fit the needs of the study, as the simulations done here are longer and evaluates the method used for each autonomous agent individually. More of this in chapter 3.

2.3 Application of the related works

With the related works presented, here is an overview how they will be used for the methods of this study.

'Crowd simulations and its applications' provides basis for the macroscopic method for navigating the agents through a crowd, as well as motivations for doing these studies in the first place. This is then paired with the microscopic approach with the methods mentioned in 'Steering behaviour for autonomous agents' to implement proper collision avoidance for the agents.

'Autonomous pedestrians' had identified distinct collision avoidance scenarios, which will be the scenarios the proposed DevVec method will try to identify. They are also used to create survey questions which would help the participants to evaluate their own behaviour in a way that is transmittable.

’Gradient-based steering for vision-based crowd simulation algorithms’ provides the basis for a contender method to test DevVec against. It’s implementation as a method is completely independent from the implementation of DevVec, but will receive similar input settings where applicable (e.g. comfort speed, vision cone angle).

’Scenario space: characterizing coverage, quality, and failure of steering algorithms’ gives insight for the relevant metrics to observe and use to pass judgement. There are no continuous simulations that are evaluated in their study. Regardless, the methods are still useful, with a few reasonable adjustments. No observers will be used to judge the methods for this study.

Chapter 3

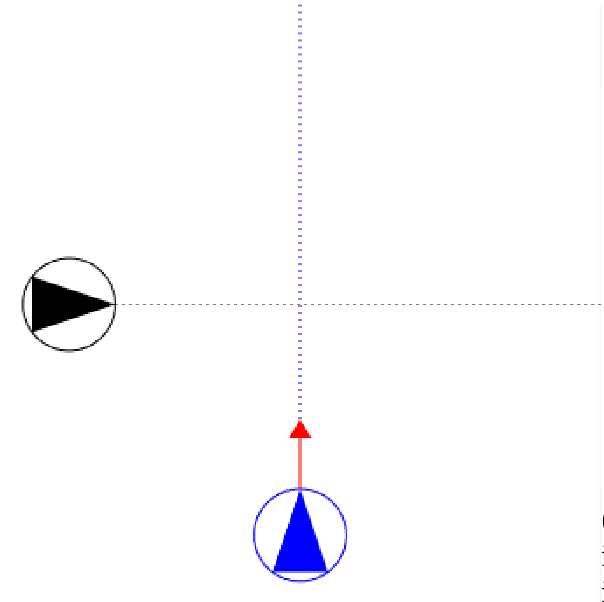
Method and Implementation

Conducting this study consists of three main parts. The first part is conducting the user survey of which responses are used to set certain thresholds for the DevVec method. The second part is to interpret the survey results to complete the algorithm of the survey, and then to test DevVec, both on its own and against the Gradient method. The third part is constructing the testing environment for which to test both methods, and everything needed for testing them, like scenes, collision detection, measuring, etc.

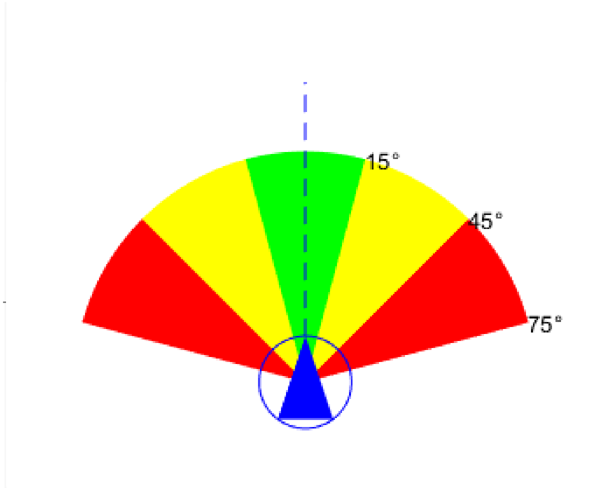
3.1 User survey

The user survey involves asking the users questions of evaluating different collision avoidance scenarios. There are a few kinds of questions. The most common kind asked the user to rank different actions in different collision avoidance scenarios, referred to as 'Ranking Questions'.

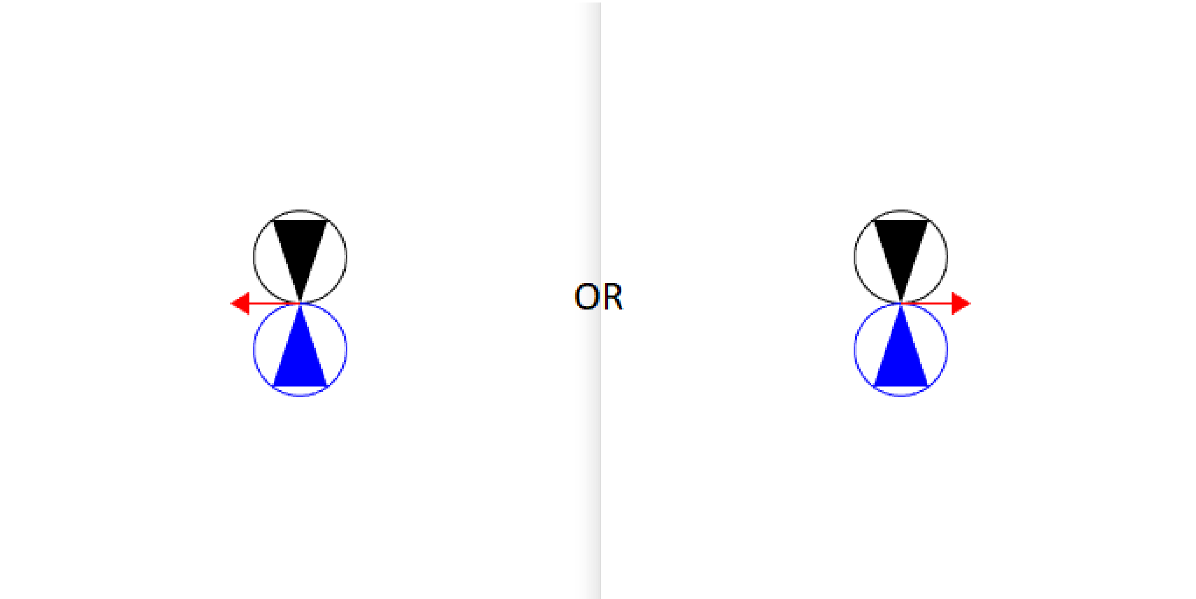
There were a few singular questions that differed from the ranking questions: One asking if the user preferred to walk around a slower pedestrian in front of it, if possible; one asking how many degrees of rotation deviation from their desired direction was too much, of which the answers was 15°, 45°, and 75°; and a final one that asked if the user felt it was inconvenient to adjust their speed frequently. The first of these was to determine some kind of maximum deviation from the intended path, because not limiting the deviation could potentially allow agents to walk in circles, or take unrealistically long and complicated detours, which is by all means unwanted behaviour. The second of these ended up not affecting the implementation of the algorithm, due to lack of idea to implement.



(a) A scenario describing 'no action' when faced with a threat from the periphery.



(b) An image helping the participant to visualise what different angles of deviation from the intended path looks like.



(c) A scenario describing 'avoid threat' when faced with an imminent threat from the front.

Figure 3.1.1: Different images provided as visual aid for the survey participants.

The scenarios presented in the ranking questions were inspired by Wei and Terzopoulos[10], but also expanded upon by identifying scenarios encountered which would not necessarily require deviating from the path, e.g. adjusting the speed. There was a difference made between suspecting a future collision and a collision already imminent or had happened. The answers to these questions, while technically possible to be presented numerically, were instead presented using natural language to help the users get a more consistent sense of what e.g. a value of '3' means in the context of the question. The answers are the following, in ascending order of score: 'Absurd', 'Non-ideal', 'Neutral', 'Reasonable', 'Expected'. The users were then asked to choose the most appropriate description of the scenario presented. This scale of answers is similar to that of the Likert scale[6].

The survey was constructed using Google Forms, which facilitated the design of the survey, as well as getting it to reach participants. The survey was then spread out through social channels as wide as possible, while also trying to avoid the risk of respondents replying with malicious intent to sabotage the results.

3.2 Interpreting and implementing survey results

The survey results were interpreted to produce thresholds for which a uniform randomly generated number (RNG) could land between, to determine which behaviour each pedestrian will employ. For determining the maximum deviation angle, the result of the RNG will apply throughout the pedestrian's lifetime, while for the rest it will only apply for a single encountered collision scenario with another pedestrian. The output range is the closed continuous interval [0,1], and the thresholds are therefore also somewhere within the interval.

The thresholds for the maximum deviation angle were implemented according to the proportions of each answer.

For the ranking questions, each answer is mapped to a numerical value, to be used for calculating arithmetic means. The answers were mapped to the values -3, -1, 0, 1, and 3, respectively. Because there is no objective way to map ordinal to numerical data for calculating means[4, 7], these values are completely arbitrary. Although, it is the relative distance between each value that affects the calculation. The intent is for the extreme answers to have a slightly more significant, non-linear, impact on the results. The mean for each combination of scenario and action was calculated, and then the proportion of each mean for a given scenario sets the thresholds in the RNG output range. The results between imminent and not-so-imminent collisions scenarios were very similar, so the implementation uses only the data from the not-so-imminent scenarios, with an additional action if the collision is imminent (explained in section 3.3.3.

It is however possible to calculate the mode answer of each question, reflecting the most popular sentiment. If the popular sentiment for an action in a specific scenario is 'less' than the popular sentiment for another action in the same scenario, it *should* typically have a lesser proportion in the range divisions.

The details of the survey can be found in Appendix A. The survey results had 62 respondents, and gave the following divisions of the output ranges, for a given RNG $X \sim U(0,1)$:

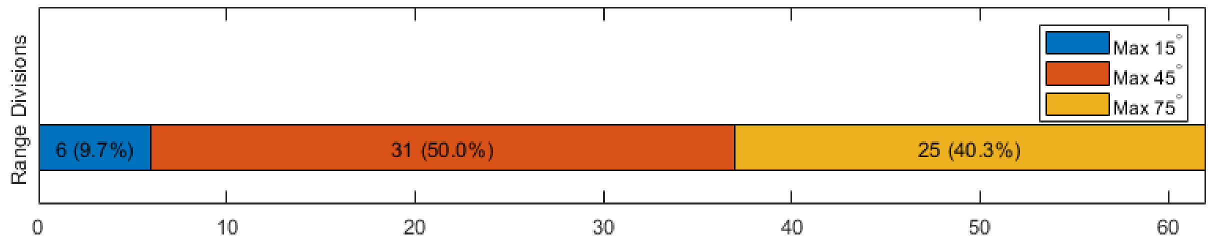


Figure 3.2.1: The division for maximum deviation angles.

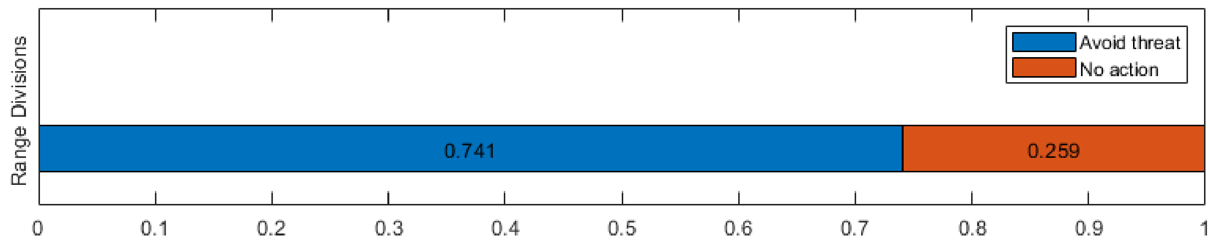


Figure 3.2.2: The division for frontal collision threats.

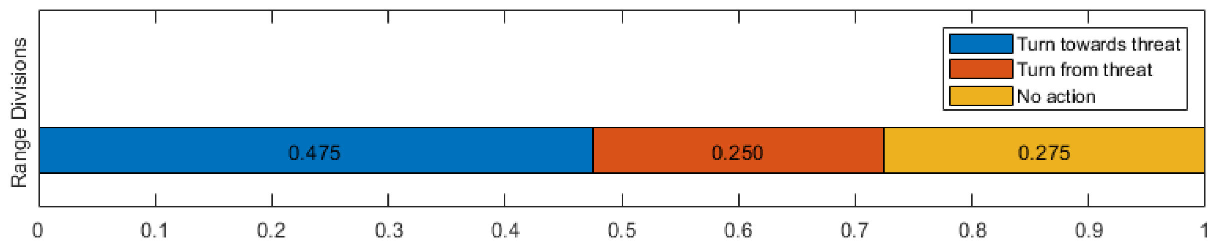


Figure 3.2.3: The division for peripheral collision threats.

The mode for each action in peripheral collision scenarios are 'Reasonable', 'Non-ideal', and 'Non-ideal', for 'Turning towards threat', 'Turning from threat', and 'No action', respectively. The mode for each action in frontal collision scenarios are 'Expected' and 'Non-ideal', for 'Avoid threat' and 'No action', respectively. These modes tells that the interpretation is at least not completely absurd.

The exact use of the thresholds will be explained further in section 3.3.2.

3.3 Test environment

To test the algorithms and to see and measure the results, a test environment was needed. It requires collision detection and viewport graphics. Unity[13] is readily available, familiar, and has the required and many other functionalities already included, so that was the choice. Unreal Engine 4/5[14] was briefly considered, but the unfamiliarity of the engine wasn't outweighed by the other potential benefits. The scene and pedestrians are built using simple standard 3D objects provided by the engine. The entire Unity project is found on a GitHub repository[3], free to download and simulate oneself.

3.3.1 The Scenes

The scenes are meant to reflect simple crowd environments of pedestrians, like small plazas, tunnels, or pedestrian-only streets. Each possible entry/exit of any scene is located at one of the cardinal directions. For any pedestrian, whichever direction is its entry cannot also be its exit. The different possible scenes are therefore a four-way crossing, a three-way crossing, a curved path, and a straight path. These are called 'Fourway', 'Threeway', 'Twoway', and 'Gauntlet, respectively. Because each entry/exit are functionally identical, each possible mirror/rotation variation of any of the scenes are practically identical. Each scene is also fitted with an invisible bounding box, which destroys any agents going out of bounds (OOB), to allow new pedestrians to spawn.

Each time a pedestrian shall spawn, one of the entries are chosen randomly, which then spawns pedestrians randomly within its volume. An exit is randomly chosen from the remaining locations. For the natural rules algorithm, the scenes are spanned by adaptive steady vector fields. For any goal a pedestrian has, there is a specific steady vector field that guides it. They are also designed to steer pedestrians away from walls. For the gradient-based algorithm, the pedestrian only has the exits as goals to move towards.

Spawning rules

Each scene spawns agents periodically, according to a set waiting value. The number of agents allowed to exist simultaneously is set by a maximum value. Once the maximum number is reached, the spawning cycle restarts without spawning an agent.

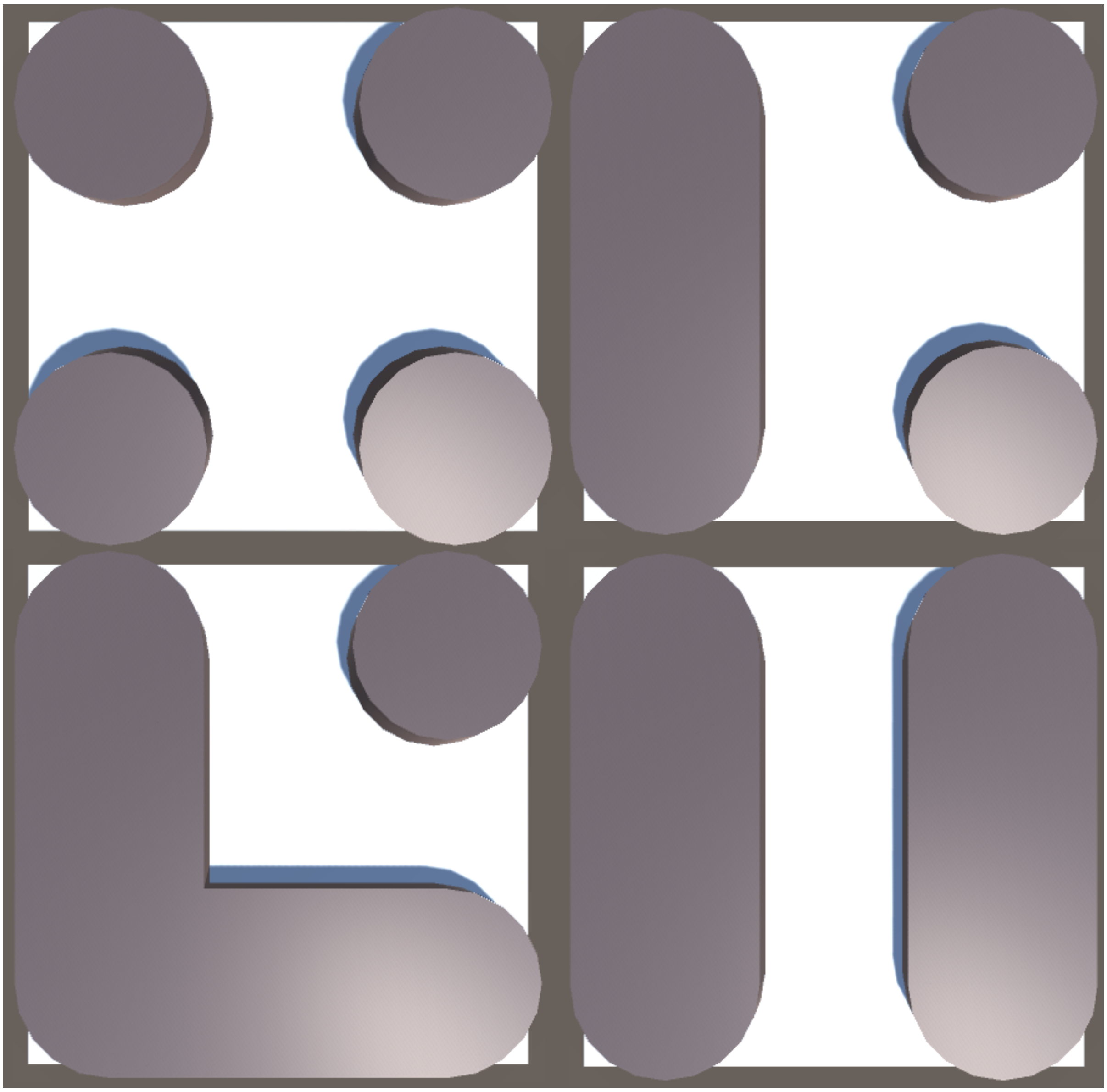


Figure 3.3.1: The scenes, from left to right, top to bottom, are Fourway, Threeway, Twoway, and Gauntlet.

3.3.2 The Pedestrians

The autonomous agents for this project are called 'Pedestrians'. A pedestrian consists of a simple capsule mesh of radius 1, along with a matching collider to detect collision with obstacles or its goal. There is also a smaller, inner capsule collider within to detect collisions with the outer collider of other pedestrians, to capture how pedestrians are capable of manoeuvring themselves in close-quarters to avoid collisions. The inner capsule collider is half the radius of the outer. To visually distinguish which direction it is heading, it has a purely decorative, small cylinder 'nose' always aligned with its internal 'forward'-vector. Each pedestrian are given a name, which is a hexadecimal

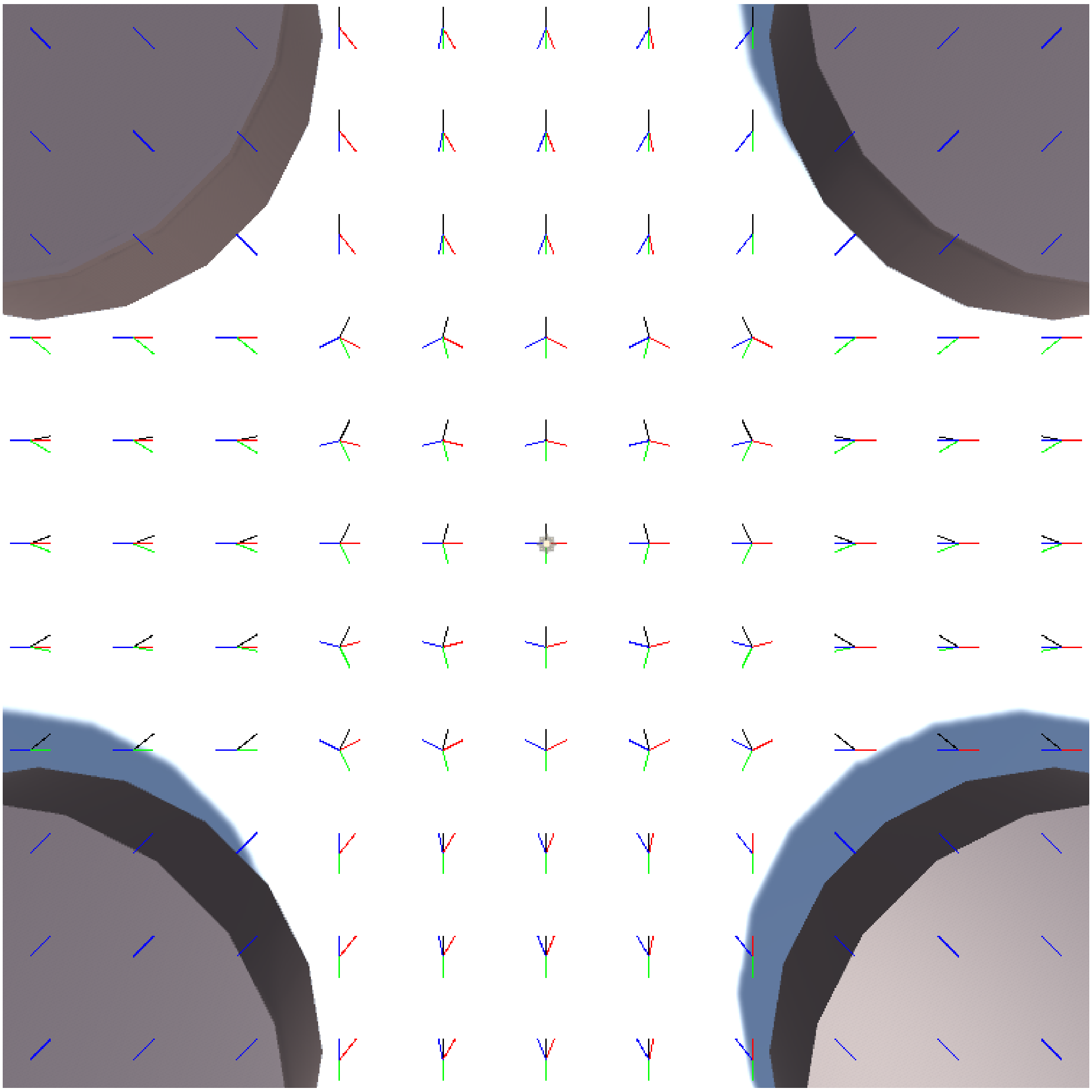


Figure 3.3.2: The four vector fields used by the DevVec algorithm. Pedestrians heading towards the north, east, south, or west exit uses the black, red, green or blue vector fields respectively. Note that there when there is overlap, then only one of the colours are displayed.

representation of a randomly generated number between 0 and 2,147,483,647 (signed 32-bit integer max value). These names are important for determining which pedestrian are responsible for different actions when they interact with each other.

For the DevVec method, the pedestrians has some additional collider meshes to act as their synthetic vision of surrounding obstacles. The Gradient method instead uses a series of ray casting originating from the pedestrians centres to implement their synthetic vision. To avoid potential arbitrary collisions with other pedestrians occurring

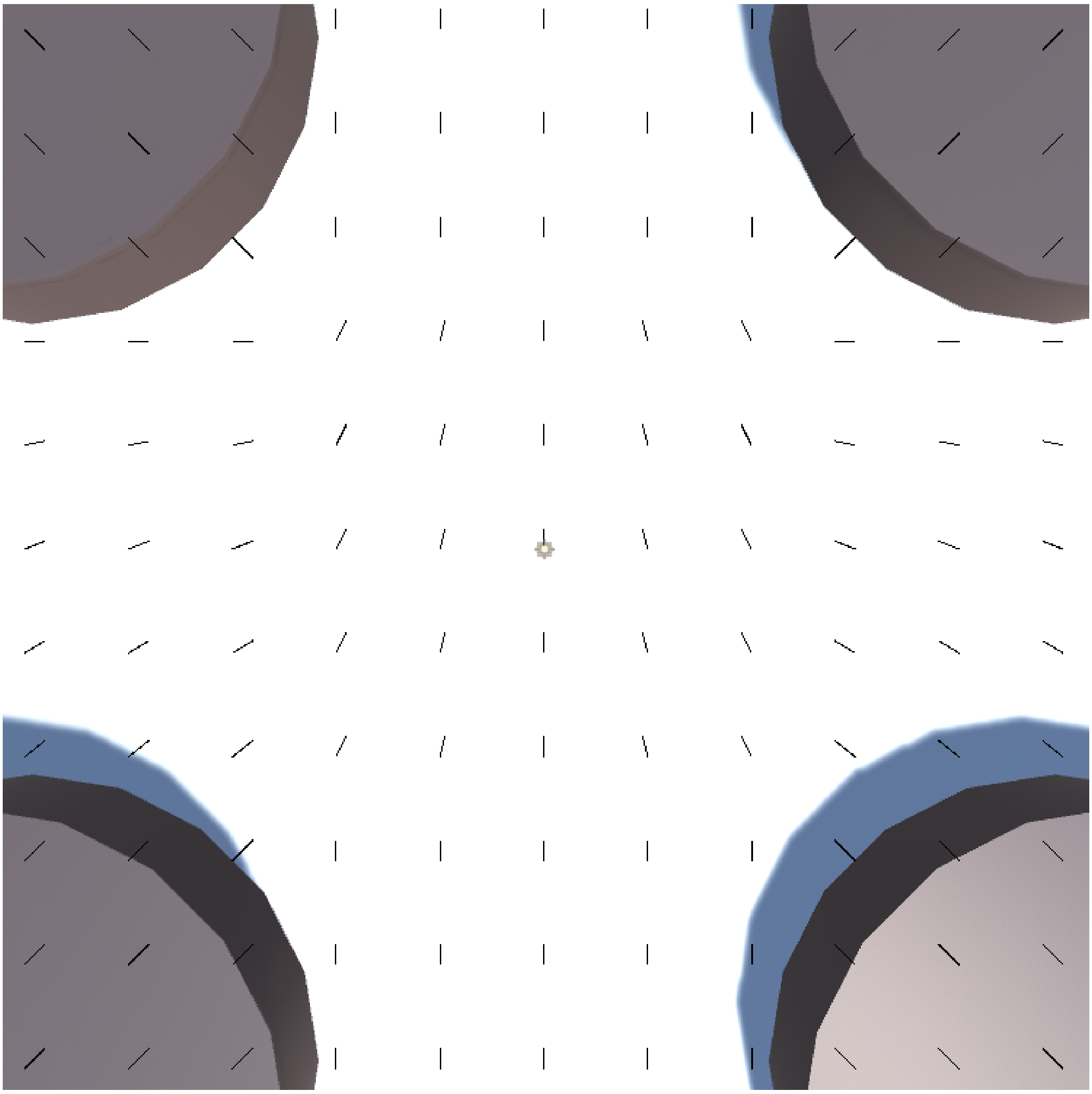


Figure 3.3.3: A single vector field, heading north.

the instant or soon after they are instantiated, there is a two-second ‘grace period’ of collision immunity after instantiation.

Once a pedestrian successfully reaches its goal, it is considered done, and is destroyed, along with appropriate functions being called to document. A pedestrian can also fail, if it collides with a wall or goes out of bounds, which results in other appropraite functions documenting that instead.

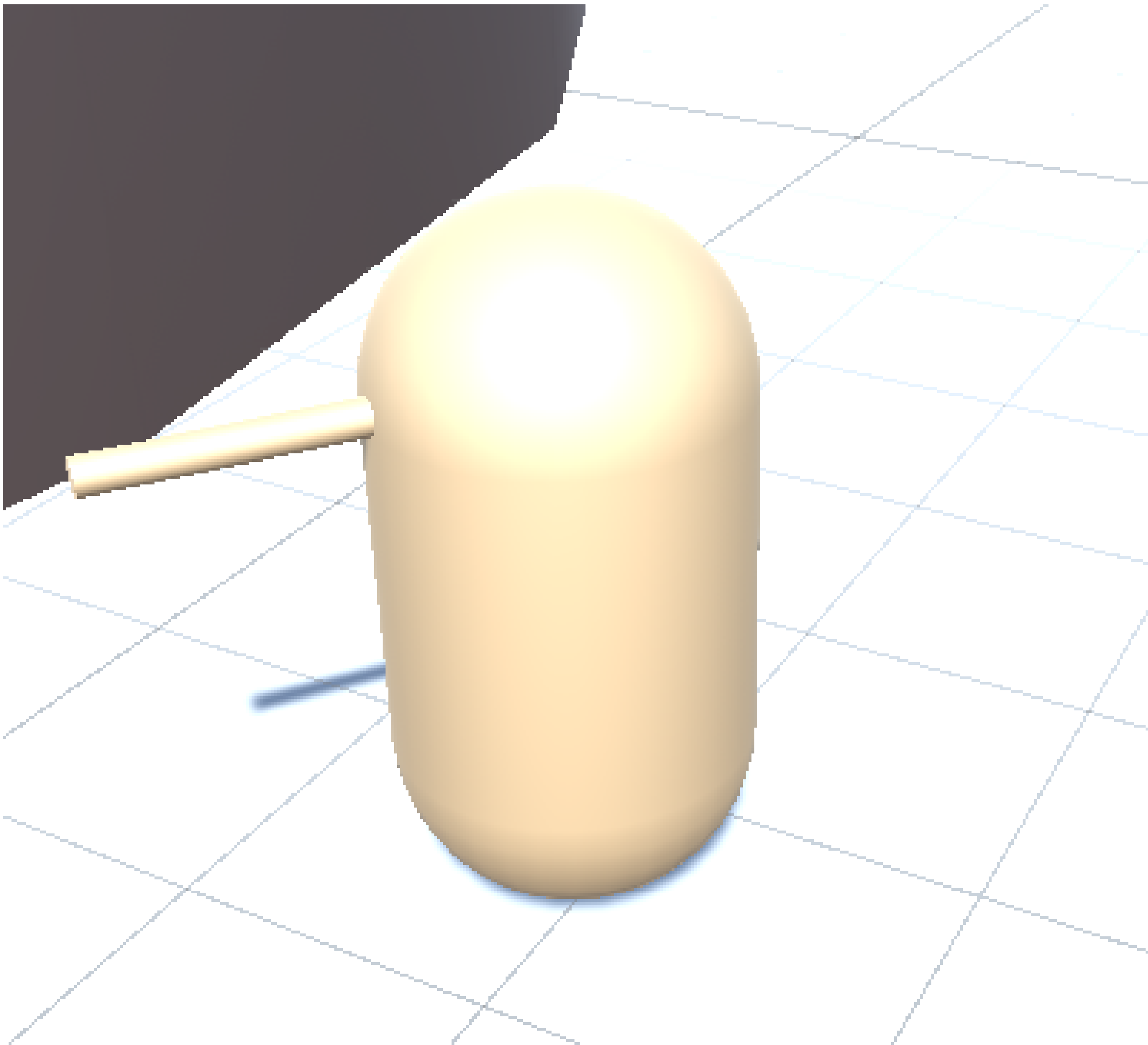


Figure 3.3.4: The simple shape of the pedestrian.

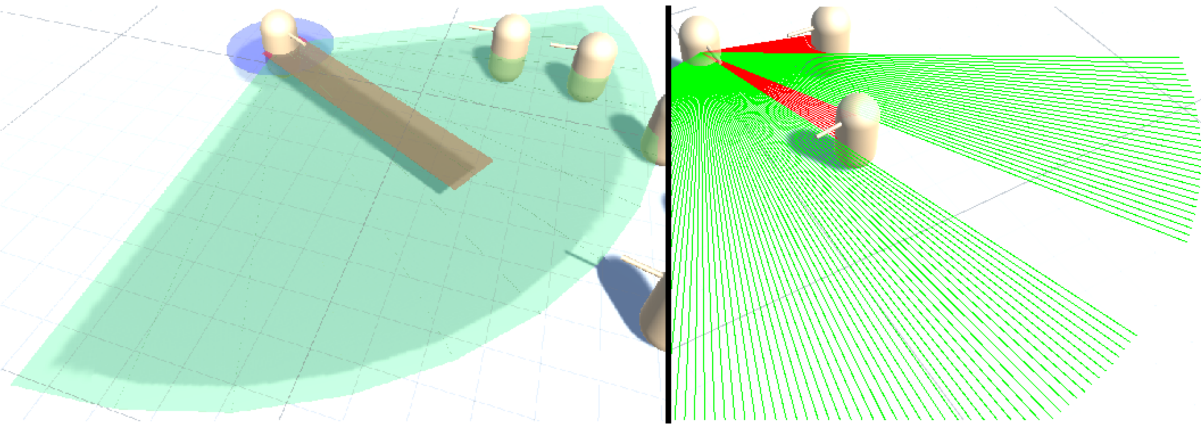


Figure 3.3.5: To the left are the colliders of the DevVec method and to the right are the ray casts of the Gradient method.

3.3.3 The DevVec method

The DevVec method consists of the idea that each pedestrian follows their given vector field, when there is no perceived threat, and otherwise reacts to the most imminent threat within its colliders. Each specific kind of threat identified maps to a specific action, which sometimes also includes no action. The three synthetic vision colliders used are the 'Front Detector', 'Peripheral Detector', and 'Proximity Detector' (seen in figure 3.3.5). Each will keep track of all relevant obstacles that enter or leave the colliders, allowing them to assess the threat level of each obstacle. The first two detectors may generate a single avoidance vector each for the pedestrian to rotate towards or from, while the last one may adjust the speed and/or the position of the pedestrian. This avoidance vector can be anything relating to the pedestrian's or the threat's own internal vectors, or it could be a vector between each pedestrian. For this method, the pedestrian has a maximum speed, which is the default speed if nothing else reduces it.

Front Detector

The Front Detector is the red collider which is directly in front of the pedestrian. Its length is 7 and its breadth is 1.3. The breadth is slightly wider than the pedestrian, so that pedestrians try to keep at least *some* distance to each other when they pass, which gives some margin of error. The length is at first glance arbitrary, but it was found that increasing the length made collisions more unlikely. This benefit stopped being noticeable beyond 7. This collider detects the most imminent threats, as anything within it is potentially in the pedestrian's path. An avoidance vector generated by the Front Detector will always be prioritised over one generated by the Peripheral Detector. It can detect walls and pedestrians as obstacles.

Because walls don't move, any wall will be prioritised in terms of avoiding obstacles. Walls will always generate an avoidance vector that is a vector perpendicular to the part of the surface of the wall that is within the collider, and will as well set a boolean telling that a wall has been detected to 'true'.

As for avoiding other pedestrians, the closest threat oriented towards the pedestrian, that is the angle between their forward vectors is greater than 90°, will generate an avoidance vector. This avoidance vector can be either the threat's negative forward vector, if the threat doesn't have the pedestrian in its Front Detector, or the threats left vector, otherwise. This is to have the pedestrian try to walk behind threats that haven't registered it, and to walk on the right of threats that have.

Peripheral Detector

The Peripheral Detector is the green collider which is a 150° cone of radius 12 projected from the centre of the pedestrian, along its forward vector, which also acts as its line of symmetry. The radius is chosen similarly to the length of the Front Detector; no significant improvements were discovered going beyond the given radius. The angle for the cone matches that of the Gradient method[5], but can be changed to anything really. This collider detects less imminent threats in the pedestrian's periphery. While it overlaps the Front Detector, that is a minor overhead of the implementation, as the priorities set between both colliders always favours the avoidance vectors from the Front Detector.

The Peripheral Detector ignores walls, as stationary obstacles can never suddenly enter the pedestrian's set path. It will only generate avoidance vectors for potential threats which also have the pedestrian in its collider, along with the requirement that their front vectors intersect within the radius of the collider. This avoidance vector will always be the threat's negative forward vector. If there are multiple threats for generating avoidance vectors, only the closest one will be chosen.

Proximity Detector

The Proximity Detector is the blue collider surrounding the nearby area of the pedestrians, with a radius of 1.5, to only account for pedestrians in the immediate proximity. It differs from the other detector colliders, as it doesn't generate an avoidance vector of which to rotate towards or from, but rather any non-zero avoidance vector reduces the pedestrian's speed instead. If no vector is generated, then the pedestrian seeks to restore its original comfort speed. A vector is only generated when the threat is within the Peripheral Detector, and its forward vector is no more than 90° rotated away from the pedestrian's own forward vector. This is mainly to not collide in the event of congestion, but also to slow down if a collision is imminent when two pedestrians are walking towards the same point almost parallel simultaneously.

It also has the authority to increase the turning speed of the pedestrian, if the threat it is reacting to is within the detector.

Reacting to avoidance vectors

An avoidance vector from either the Front or Peripheral Detector will prompt the pedestrian to perform an action which supersedes following its given vector field, which may or may not cause significant deviation from the path given by the vector field.

For an avoidance vector generated by the Front Detector, the pedestrian has the option to either turn away from the threat or do nothing at all. If the threat also considers the pedestrian to be a threat, then the choice is determined by the RNG output range. A problem occurs when both pedestrians decide to do nothing, then a cooperation routine kicks in for the pedestrian with the lesser valued name, e.g. `'AF6593DE'` is less than `'B784B12F'`. This routine will forcibly switch the affected pedestrian's action to follow the avoidance vector. It will also allow it to side-step in an appropriate direction, by moving the pedestrian along its right or left vector (whichever is least effort), when the Proximity Detector has the threat within its bounds.

For an avoidance vector generated by the Peripheral Detector, the pedestrian has the option to turn towards its threat, do nothing, or turn away from its threat. This choice is similarly determined by the RNG output range. It also has a problem when both pedestrians decide to do the same action, then the cooperation routine kicks in, in the same manner as before. For this case, the lesser named pedestrian will randomly choose an action differing from the one before.

3.3.4 The Gradient method

The gradient-based algorithm is described in vivid detail by its authors.[5] Regardless, the a more detailed description of it can be found in Appendix B, along with descriptions for the notation. The purpose of the algorithm in this study is to have an established method for which to compare the novel steering algorithm with. It has been faithfully implemented within the test environment. However, there are a few minor adjustments made to it so that it fits in the test environment and to the purpose of this study:

- It does not produce images of the pedestrian's synthetic vision. The abstract image the ray casts produce are one-dimensional, as movement and obstacles are limited to the XZ-plane.
- The sigma values presented in their study are not necessarily optimal in this

study, so they have been adjusted.

- The numerical limitations of their test method were omitted in the paper. It was therefore unclear how they dealt with e.g. floating point overflow or division by zero. Rules and limitations were introduced to guarantee runtime safety.

As described by the authors the behaviour of this algorithm depends on which values each standard deviation has for different normal distributions. These distributions are used for the Evaluation phase, where the costs of each possible action is calculated. The standard deviations for these distributions are described as different sigmas: σ_α, which determines how strong a turning action is expressed; σ_s, which determines how strong a speed adjustment is expressed; σ_{ttca}, which determines how long in advance the pedestrian is able to take the threat to account; and σ_{dca}, which determines a minimum distance the pedestrian strives to be from each threat. The sigma values proposed by them were the best, learned values for their simulation environment, and not necessarily as such for this environment. Therefore, it has been tested extensively to determine which fits best for this environment. Each sigma, except for σ_{dca}, has been iterated 10 times, totalling to 1000 test runs of 10 minutes each. The reason σ_{dca} doesn't need to be learned for this test environment, is because it is easy to observe visually how close the pedestrians will get to obstacles when not stressed. The distance which corresponds to 'walking right next to an obstacle' was achieved using $\sigma_{dca} = 0.3$. Whichever set of sigmas had the best outcome in terms of collisions would be the candidate for comparison with the natural rules algorithm. The settings for this test are the following:

$n \in \mathbb{N}, n < 10$

$\sigma_\alpha = 0.5 + n * 0.5$

$\sigma_s = 0.5 + n * 0.5$

$\sigma_{ttca} = 3.0 + n * 0.7$

$\sigma_{dca} = 0.3$

The results, which can be found in the repository (sigmaResults.txt), determined that test run 24 was best. It had the least number of collisions total, and zero OOB. Any consideration for the ratios were ignored for this case, as the number of collisions and OOB takes precedence. Because test 24 was best, the sigma values to be used in comparing the Gradient method with DevVec are the ones used in the test:

$\sigma_\alpha = 2.5$

$\sigma_s = 1.5$

$\sigma_{ttca} = 3.0$

$\sigma_{dca} = 0.3$

This method implements the default speed of the pedestrian as its 'comfort speed', not maximum. This allows it to go both above and beyond this default value. It is increasingly difficult for the speed to go further away from this comfort speed value.

3.3.5 Evaluating the results

Following the example set by Kapadia et al[8], the metrics being evaluated are collisions, time to reach the goal, and distance covered. The number of collisions are counted, and a distinction is made whether the collision was a wall or with another pedestrian. It is also interesting to measure the number of times a pedestrian goes out of bounds.

The simulation for both methods has been done without pedestrians trying to avoid each other, to get recorded times and distances. For the DevVec method, this simply means following the vector fields. However, the Gradient method still needs to avoid colliding with walls, so that was still accounted for. The time and distance varies depending on where at an entry a pedestrian spawn and where its goal is located. Measuring is done for 20 different segments of an entry and for either its goal being on opposite or to the left/right (called 'curved'). Each time a pedestrian reaches its goal, the recorded metrics are divided by the corresponding metrics of the pedestrian to get a ratio. Each ratio for each segment and goal are used to calculate a mean to tell the average performance for each situation. Note that the recorded metrics are not necessarily "optimal", which makes it possible to get a ratio larger than 1.

The recorded metrics are described as $M(i,p) = (M_{time}(i,p), M_{distance}(i,p))$, where $P(i,p) \in \{P(i,p)_0, P(i,p)_1, \dots\} = AllP(i,p)$ is a single pedestrian's time taken and distance covered and $i \in \mathbb{Z}_{20}, p \in \{'opposite', 'curved'\}$, and are calculated as such:

$$M(i,p) = \frac{\sum P(i,p)}{|AllP(i,p)|}$$

During the simulation with collisions avoidance, it produces test metrics for each pedestrian $T(i,p) \in AllT(i,p) = \{T(i,p)_0, T(i,p)_1, \dots\}$. They are then divided by the recorded metrics to produce ratios, and the arithmetic mean is calculated for these ratios, producing mean ratios $R(i,p)$:

$$R(i,p) = \frac{\sum \frac{T(i,p)}{M(i,p)}}{|AllT(i,p)|}$$

These ratios will give some indication how well a method performs in terms of how adding the collision avoidance affects the time and distance metrics.

A low collision/OOB count and a high (close to 1) mean ratio for time and distance covered will reflect positively on the method for the simulation attempt. Wall collisions and OOB are to be considered more severe flaws than pedestrian collisions, as they are indicative of severe failures in prioritisation.

Chapter 4

Results

A series of simulations were ran with different options. Each simulation runs for 20 hours (72000 seconds), and there is a single such simulation for each combination of method, scene, and congestion level. A test run of 10 hours was done a few times with the same settings, and the results were acceptably stable at this level. The decision for 20 hours was made for good measure.

The methods are, as previously explained, 'DevVec' and 'Gradient'.

The scenes are, also as previously explained, 'Fourway', 'Threeway', 'Twoway', and 'Gauntlet'.

Finally, the congestion levels are 'Low', 'Medium', and 'High', which corresponds to a maximum number of pedestrians allowed to exist simultaneously. These congestion levels are 10, 30, and 100, respectively. They were chosen subjectively by simply observing the amount of action for each level. The intent is that low congestion rarely sees complex collision avoidance scenarios; most of the time, there will only be zero or a single avoidance vector for each pedestrian, for the DevVec method. Medium can quite often encounter more than a single avoidance vector, and the behaviour of the pedestrians become quite less predictable. High is more or less a stress test of the methods; they are clearly being overwhelmed and both methods handles it quite poorly. The spawn rate for each congestion level is determinated through checking which spawn rate manages to reach the maximum number of allowed pedestrians regularly. They are 0.5, 1.5, and 5.0 pedestrians/second respectively.

The maximum/comfort speed is set to 1.5 for both methods.

As mentioned in section 3.3.5, the data that will be evaluated are the number of collisions/OOB and the time and distance ratios.

Accounting for each method, scene, and congestion level, there are a total of 24 simulations of 20 hours each. These have been executed in parallel across multiple devices, and multiple sessions on each device, which took roughly 40 hours in total. The hardware of each device does not affect the simulation, other than allowing more sessions of higher congestion to be run simultaneously.

4.1 Counting collisions

After the simulations had finished, it became apparent that no OOB ever occurred. That is to be considered quite good for both methods. Because of the lack of OOB, the presentation of them has been omitted entirely. Below, you will find bidirectional, horizontal bar graphs for displaying the number of collisions with either pedestrians or walls. The bars are presented in such a way to facilitate comparisons between the two methods for otherwise same settings.

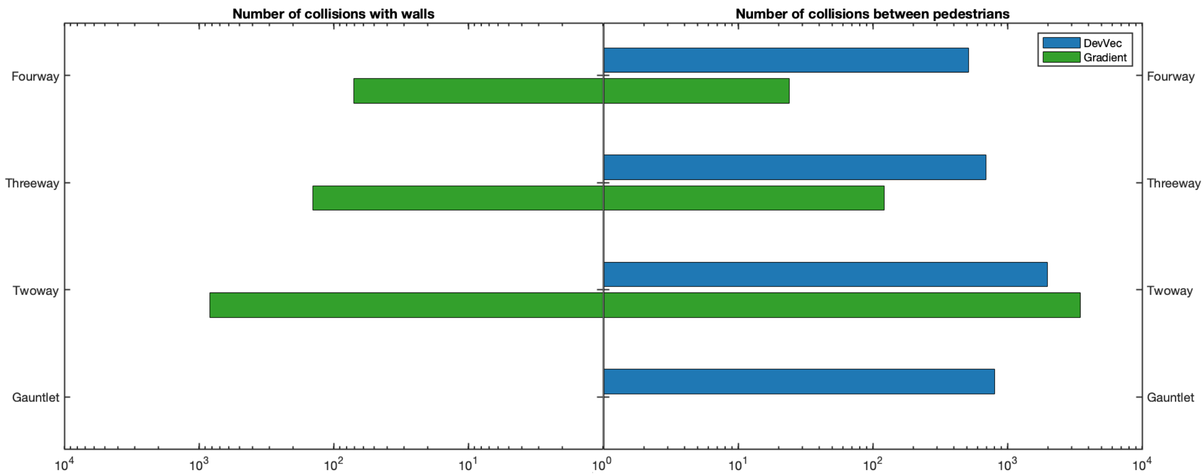


Figure 4.1.1: Number of collisions for low congestion levels.

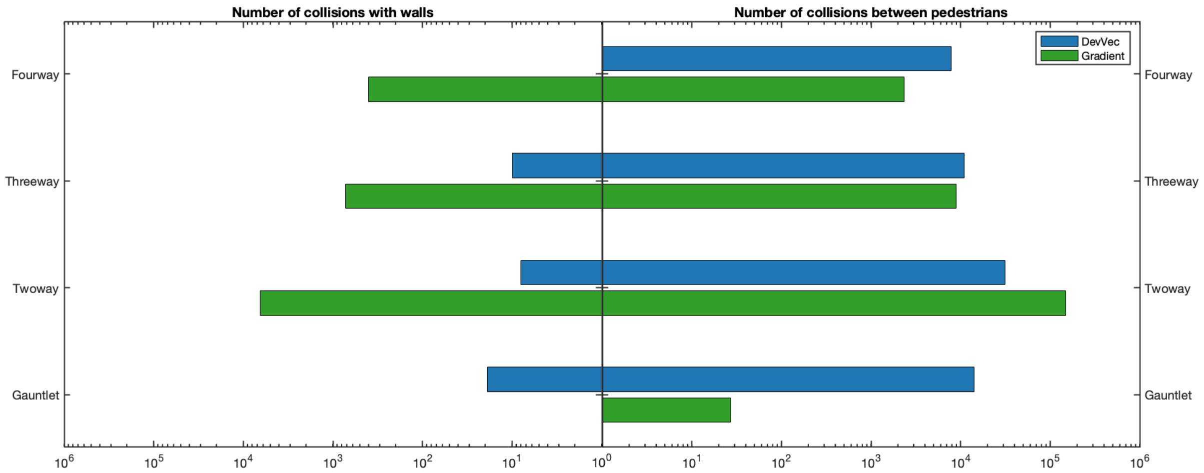


Figure 4.1.2: Number of collisions for medium congestion levels.

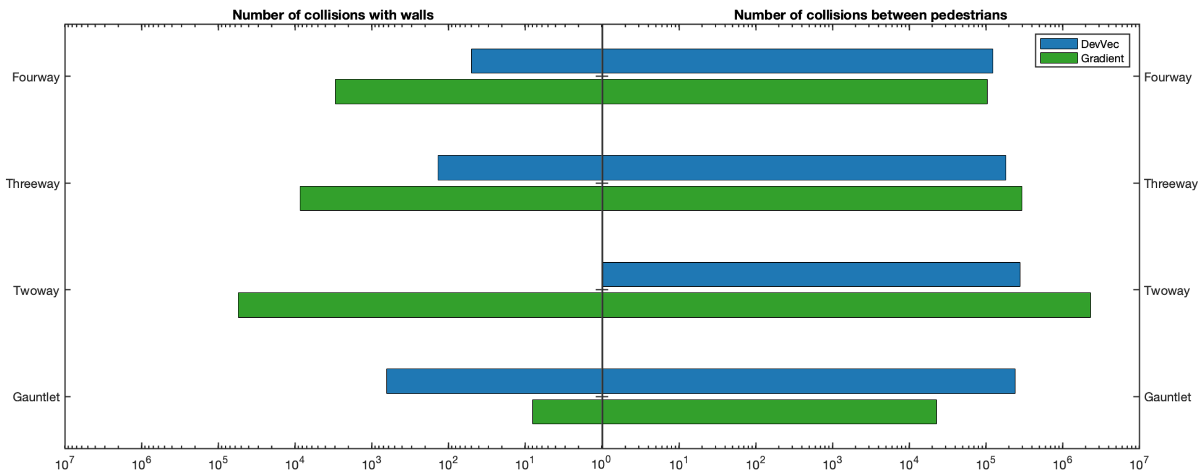


Figure 4.1.3: Number of collisions for high congestion levels.

4.2 Ratios for time and distance

Below, you will find ratios for time taken and distance traversed to reach the goal. Note that if the pedestrian's speed isn't adjusted much or even at all, the ratios will be very similar. Again, the tables are constructed to facilitate comparison between each method for otherwise same settings. The ratios presented are the arithmetic mean of each mean for the 20 different starting positions and whether or not the goal was opposite or to the left/right of the start. In the Twoway and Gauntlet scenes, goals to the left/right and opposite respectively wasn't possible, so the number of means for those cases are half of the other cases.

Scene	Low		Medium		High	
	DevVec	Gradient	DevVec	Gradient	DevVec	Gradient
Fourway	0.989	1.013	0.966	1.006	0.912	0.993
Threeway	0.991	1.011	0.966	1.002	0.894	0.989
Twoway	0.976	1.005	0.911	0.992	0.580	0.970
Gauntlet	0.997	1.005	0.983	1.002	0.806	0.994

Table 4.2.1: Ratios of time.

Scene	Low		Medium		High	
	DevVec	Gradient	DevVec	Gradient	DevVec	Gradient
Fourway	0.991	1.013	0.973	1.008	0.945	0.999
Threeway	0.992	1.012	0.974	1.005	0.947	0.997
Twoway	0.980	1.006	0.945	0.999	0.920	0.997
Gauntlet	0.997	1.006	0.989	1.004	0.978	0.997

Table 4.2.2: Ratios of distance.

Chapter 5

Discussion

With the results of the simulations compiled and presented in chapter 4, an evaluation can be done around these results. In this chapter, a few things will be analysed, interpreted, or reflected upon.

First off, the survey which has helped construct the DevVec method. Second, the trial to determine the best sigma values, which made the Gradient method the best possible contender. And third, the results of the final trial will be analysed and interpreted to produce a valid answer to the research question. Then, having finished the analysis, a conclusions will be presented. Finally, because this study touches a very small area of crowd simulations and steering behaviours, some suggestions for future work will be presented.

5.1 The survey

62 respondents is admittedly somewhat lacking. It can't really be specified how many it potentially reached, but definitely up in the tens of thousands. Having less than a hundred respondents is not exactly a good response rate. Fortunately, the DevVec method is not entirely dependent on the survey or its results. Because of the non-ideal response count, a choice was made to make the method have a quite rigid definition without any interpreted survey results, so much so that it could be argued it would do fine without it. Regardless, the experiment to incorporate the public in this manner has been interesting, and the idea should not be written off based on these results.

Interpreting the results was a challenge to overcome, due to the fact that the answers for most of the questions followed an ordinal Likert scale[6]. There is no way to interpret an ordinal scale[4, 7] to produce meaningful arithmetic means of the data, without adding a subjective interpretation of said data. It is therefore not claimed here that the method used here is *the* method to use, but rather *a* method to use. The intent was to have the more extreme values have a more significant impact on the result, to reflect the surety of the respondents answering as such, and that could be considered to have been accomplished.

5.1.1 Survey feedback

There was the option for respondents to leave comments on the survey. Filtering out comments that did not comment the way of conducting the survey, there were a few trends:

- Navigating any arbitrary crowd comes so much natural to pedestrians, that they do not reflect upon their way of navigating. Therefore, it is hard to answer the questions without doing some guessing or subjective interpretation of ones personality.
- The survey does not take to account different cultures or social roles, and pretends that every respondent is the same.
- The survey does not take to account the different mental states a single pedestrian may have, whether it be different crowds or the same.

The first bullet is completely understandable. Moving by foot is such an integral part to people's lives, that it would be silly to assume that any functional human had to learn in a manner that requires reflection. Asking anybody with a driver's license how they would handle different scenarios in traffic, you'd probably receive a pretty clear and close-to-reality reply, probably because that is how they learned to drive by the institution teaching them.

The second bullet is less clear. It can be assumed that the respondents have a pretty clear identity of themselves and their social role. The typical crowd one has to navigate rarely has any specific individuals which radiate a differing social role that requires different navigation behaviour. Regardless, anyone with a concern like this was told to assume the others were of similar stature.

The last bullet is a bit of a problem, because responding to the survey in the comfort of your home, office, or pretty much anywhere one is being stationary probably puts the respondent in a different state of mind than actually navigating a crowd. Sadly, there's no remedy to that, excepting the absurd notion of enforcing respondents to participate *while* in a crowd situation.

The feedback can be considered to be valid and important, but because they survey results ended up not as integral to the method as intended, it is very likely these problems did not affect the method in any significant way.

5.2 The sigma trial

Determining the correct sigma values to use for the algorithm was an important task to do, as the intent was for the Gradient method to be the high threshold for which to compare DevVec with. If it was not performing at least somewhat ideal, then the comparison would have been very flawed. The issue is that the sigma values are four in total. Iterating over each x times would result in x^4 different simulations. However, it was later discovered that σ_{dca} didn't need iteration, because the result of its value was very apparent, regardless what the others were. x^3 iterations is better, but still a lot for e.g. $x = 10$, which would result in 1000 iterations. The test range and its fidelity was determined by the examples of Dutra et al.[5] which resulted in $x = 10$. The original intent was for the each iteration to run for 1 hour each, as it was discovered through random sampling that the results were more consistent at these runtimes. Upon calculating the runtime for 1000 iteration, it was apparent that it would take a little more than 41 days, which wasn't feasible given the time frame. One week needed to be the maximum, so the resulting runtime for each iteration was set to 10 minutes. It was less consistent, but deemed consistent enough to still make valid choices.

Having inspected the results, it was apparent that for some high or low sigma values, the method's performance was very bad, regardless of the other values. One such example was low σ_{ttca}, which effectively made the pedestrians almost completely unaware of any collision threat. Looking back at that fact, the number of iterations could have been reduced, and the iterations could therefore have been done for longer. Regardless, the resulting values proved to make the Gradient method a good contender to DevVec.

5.3 Evaluation of the DevVec method

With results for both the number of collisions and time spent/distance covered, it is time to evaluate the DevVec method. It is worth reminding that the purpose of this study is *not* to determine superiority of any method, but rather the viability of DevVec. Having good collision results will weigh heavier than good time or distance results, as collisions do typically not happen in real-life scenarios, while taking slight detours or walking a little slowly through the environment often do happen. This is not a perception study, but having virtual pedestrians clip through each other or walking against an obstacle continuously *should* be pretty detrimental to the immersion for the viewer.

5.3.1 Collisions

The fact that there are numerous collisions for any of the simulations is testament to the difficulty of crowd simulations. A 'perfect' crowd simulation should preferably have no collisions at all. There is no clear better method in terms of collisions. However, in most of the cases the blue bars for both metrics for DevVec are smaller than the green ones for Gradient. It was mentioned in section 3.3.5 that wall collisions are typically worse than with pedestrians, so a higher number of those should reflect worse on the method than a higher number of the other.

Collisions with pedestrians

There is only kind of simulation where DevVec is not significantly worse than Gradient, and that is for all ones taking place in the Twoway scene. Why is that? As evident in the 5.3.1, when there is only two starts/goal, and the path is curved, the pedestrians tend to cluster towards the one and only convex corner in the scene. Because DevVec gives the pedestrians the instruction to follow the vector field when not encountering any threats, this vector field can actively 'space out' the pedestrians to not cluster as much in the corner, depending on where they spawn. It is also evident for the high congestion case that the pedestrians tend to stand still a lot for DevVec, because of the congestion happening.

For all the other scenes, either the results were similar, or Gradient was better. This can be attributed to the fact that Gradient successfully reacts to several obstacles simultaneously. DevVec has many cases where it successfully avoids collision, only to encounter a new, different threat directly afterwards; it has no time to react properly. The Gauntlet scene is the only scene to have Gradient perform consistently better, and even having *zero* collisions for low congestion levels. Why is that, then? Well, other than the previously mentioned fact, having the only two starts/goals be opposite of each other, there's not much reason for any pedestrian to cross paths with each other. When pretty much every collision threat is a frontal threat, they are almost always seen at the maximum vision distance, giving the pedestrian a lot of time to react.

Collision with walls

It is evident that while DevVec often performed worse than Gradient in terms of collisions between pedestrians, it is clearly better at avoiding walls. The only real exception to this observation is the Gauntlet scene. This can probably be explained with the fact that DevVec prioritises avoiding static obstacles like walls, and will most of the time make the pedestrian walk straight into another pedestrian to avoid going into walls. Gradient doesn't differentiate between static or dynamic obstacles, although it does take account to the speed of the obstacles, which is non-zero for any dynamic obstacles. It is therefore not unlikely that with no real valid options, the calculated gradient tells it to go towards a wall. Why the Gauntlet scene is the exception for this case is probably due to the same reason as explained previously; there is almost always plenty of time to react properly.

5.3.2 Time and Distance

Even if the Gradient method is superior in every case in these results, it is often by a *very* small margin. The only exception to this the time ratios for the high congestion, which is the only time the ratio drops below 0.9 for DevVec, although not for the Fourway scene. This can be explained by looking at how the congestion plays out for the DevVec. Because there is so many pedestrians, they end up creating very dense areas, that slow them down to a full stop, as seen on figure 5.3.1. This does not affect the distance covered much, of course, but the time ratios suffer greatly from this. The Gradient method do slow down the pedestrians sometimes, but practically never to a full stop, which explain why its ratios are consistently high for this case. The ratios are evidently worse for the scenes that promotes further congestion by reducing the number of entries and exits and has curved paths rather than straight ones; this is bound to increase the pedestrian density in the scenes.

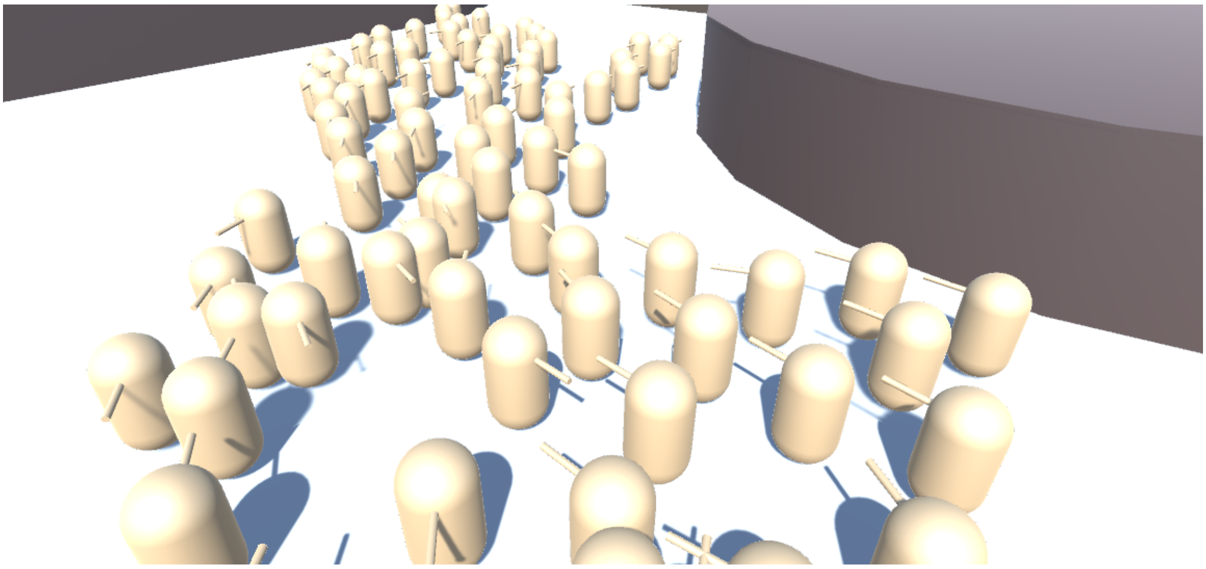


Figure 5.3.1: The high congestion in the Twoway scene has practically put the pedestrians at a full stop when using DevVec.

It is interesting to note that for many cases the ratios are higher than 1 for the Gradient method. Why is that? It is probably due the fact that the method has capability to increase its speed above what otherwise is max speed for DevVec. It should only do so if it help avoiding obstacles, which is bound to happen at least a few times in any scene and for any congestion level. It is an unfortunate difference between the methods, that does not have any clear remedy, other than accounting for this fact in the evaluation.

5.3.3 The perceived realism of the simulation

As mentioned in the delimitations in chapter 1, this study is not a perception study. Nevertheless, it can be stated as safely as possible that the simulation at the very least does not look absurd when using the DevVec method. Each pedestrian reaches the goal or crashes into a wall, and never goes OOB. For the cases when they reach the goal, the path in almost all of the observed cases looks *plausibly* realistic and natural.

To reiterate, this evaluation is *not* the focus of the study, but it was worth mentioning for anyone considering if it works to do a perception study with the method.

5.4 Future Work

There is a lot of possible future work to do for the interested. Here are some suggestions for those interested in building upon the work done here:

- Improve upon the survey. Get more responses from different people, make sure that the questions doesn't confuse them, give some incentive to respond, have some kind of quality control of the answers, and the list goes on.
- Go beyond the survey, and find some way to observe people in crowds, and translate the observations to algorithmic instructions. Shibuya Crossing is a good example. This way, the uncertainty of people misjudging their behaviour or answering inconsistently between each other is removed. However, there are a plethora of new challenges with this approach.
- Tied together with the two points above, improve the integration of the survey or observational results. With better input, there is better justification to spend the time to implement those results in a more all-encompassing way in the method.
- Try an alternative method of having each pedestrian's interpretation of the vector field rotate by an angle of θ to account for threats encountered, removing the complex deviations that may sometimes lead the pedestrian along a longer than necessary path.
- Construct more complex scenes, like the one mentioned in Shao's and Terzopoulos's study.[10] Having different vector fields switch between each other as the pedestrian moves around the complex scenes sounds like quite a simple way to get it done.
- Use the method in a perception study. Applying more realistic models to the pedestrians, animating them, having them walk around in a photo-realistic environment, and then asking observers what they think of it sounds like good basis for a study. It can be compared with other crowd simulation methods, as well.

5.5 Conclusion

Having finished the analysis of the method and results of this study, it feels safe to claim that the DevVec method is definitely viable in terms of simulating crowds. With the method being so novel as it is, there is no denying that more can be done to take the method through additional trials and tribulations. Compared to the Gradient method, it held up fine, even if it was only better in a few specific cases.

It certainly helps to answer the research question by lowering the threshold for the claims, while still keeping the question relevant and interesting. It is much more difficult to argue for superiority than viability. Superiority requires the method to be useful *and* better in all regards, or at least most of them, while for viability it is enough to prove that it is able to keep up in most regards, and DevVec has certainly done the latter.

Bibliography

[1] *Age of Mythology/Empires-series*. Developed by Ensemble Studios. Published by Microsoft Game Studios. URL: https://www.ageofempires.com. 2000-2021.

[2] Berendt, Filip. “Evaluating different vector combination methods for complex collision avoidance, in a four-way crossing of pedestrians”. URL: https://docs.google.com/pres Fm8nZn2Okboqu5NQdSY/edit?usp=sharing, slides 55-66. 2021.

[3] Berendt, Filip. *ExSteering*. Absolutely no guarantees of expectations or quality are implied. Read more at the README of the repository for more information. URL: https://github.com/SuMinRo/ExSteering.

[4] Bhandari, Pritha. *Ordinal Data: What Is It and What Can You Do With It?* Accessed 2022-05-27, URL: https://www.scribbr.com/statistics/ordinal-data/. 2021.

[5] Dutra Teófilo Bezerra, et al. “Gradient-based steering for vision-based crowd simulation algorithms.” In: *Computer graphics forum* 36.2 (2017), pp. 337–348.

[6] Jamieson, Susan. *Likert scale | social science | Britannica*. Accessed 2022-06-02, URL: https://www.britannica.com/topic/Likert-Scale.

[7] Jeff Sauro, PhD. *Can you take the mean of ordinal data?* Accessed 2022-05-27, URL: https://measuringu.com/mean-ordinal/. 2014.

[8] Kapadia Mubbasir, et al. “Scenario space: characterizing coverage, quality, and failure of steering algorithms.” In: *Simulating Heterogeneous Crowds with Interactive Behaviors, AK Peters/CRC Press* (2016), pp. 193–210.

[9] Reynolds, Craig W. “Steering behaviors for autonomous characters.” In: *Game developers conference* 1999 (1999).

[10] Shao, Wei and Terzopoulos, Demetri. "Autonomous pedestrians." In: *Graphical models* 69.5-6 (2007), pp. 246–274.

[11] *Spy Party*. Developed and published by Chris Hecker. Co-developed by Cimino. URL: http://www.spyparty.com. 2018.

[12] *Total War-series*. Developed by Creative Assembly, Feral Interactive. Published by Electronic Arts, Activision, Sega. URL: https://www.totalwar.com. 2000-2001, 2002-2004, 2005-2022.

[13] *Unity Real-Time Development Platform*. URL: https://unity.com.

[14] *Unreal Engine 5 - Unreal Engine*. URL: https://www.unrealengine.com/en-US/unreal-engine-5.

[15] Xu Ming-Liang, et al. "Crowd simulation and its applications: Recent advances." In: *Journal of Computer Science and Technology* 29.5 (2014), pp. 799–811.

Appendix - Contents

Appendix A

Survey Details

Below is the survey as the participants would view it, albeit in a slightly different format than the native Google Forms format.

'Natural' navigation behaviour of pedestrians in a crowd.

Hello, and thank you for taking the time to answer this survey. It will ask of you to rank different collision avoidance behaviours (or lack there of) of pedestrians as more or less 'normal' or 'natural', while navigating a crowd of people as a pedestrian, like a plaza or a four-way crossing of shopping streets. It is important to remember that there are no vehicles involved, ever, and therefore there are no roads for such vehicles either. Some situations may sound silly to you, but please treat them as serious as the rest. The entire survey should take about 5 minutes.

Your response is used to compose simple different behaviours in a steering algorithm in a virtual environment, which can then be blended together to make a much more complex behaviour emerge. This algorithm will then be compared to more-established variants, to see how it holds up. Your response is completely anonymised and won't be used for anything else. The collective result of this survey may be presented in a research article. That said, by submitting this form, you implicitly agree on your answers being used for this explicit purpose.

Thank you in advance!
//Filip

Instructions

Some of these instructions may seem obvious, but in case you become confused during the survey, you can always look back here for clarification. If this section doesn't help you with some confusion, I'd love to hear about it in the comment section at the end.

You will consider different scenarios involving collisions and attempting to avoid them. In these scenarios, you will be the color blue, while the other will be the color black. The direction of each agent can be interpreted by looking at the triangle within each circle as a shaftless arrow, with the pointiest corner being 'forward'.

For some of the collision avoidance scenarios, dotted lines will describe the paths taken without any action, and where those lines cross is where the collision is bound to be.

For all types of scenarios, a red arrow will describe the action being considered in an appropriate manner according to the text of the question. A red arrow from the front of your agent adjusts your rotation, and if it points in the same direction, there is no change. A red arrow from the center of your agent adjusts speed, and in this case only slows down with an arrow point backwards.

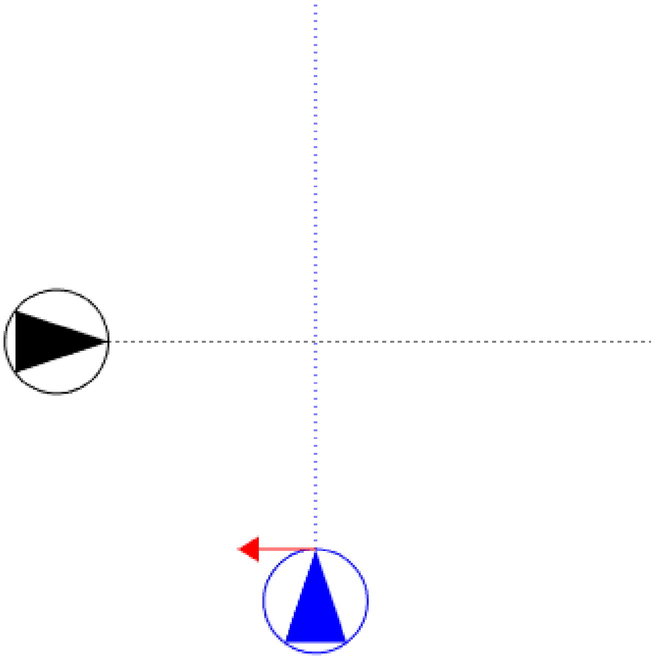
The structure of each question is different. Some may require you to finish a sentence, while others will simply want an answer to a question.

Do not consider what anybody else might think; I'm interested in what *you* think. This is important, because your response will determine the behaviour of an individual virtual agent, which is supposed to reflect the behaviour of a single person.
Some scenarios may also be influenced by whether or not you're in a hurry, or you know where you're going, etc. Try to choose the factors that would result in less traveling time, but wouldn't be considered as 'stressful' situations to you.

Avoiding Collision

This section treats only behaviours which involves avoiding collision.

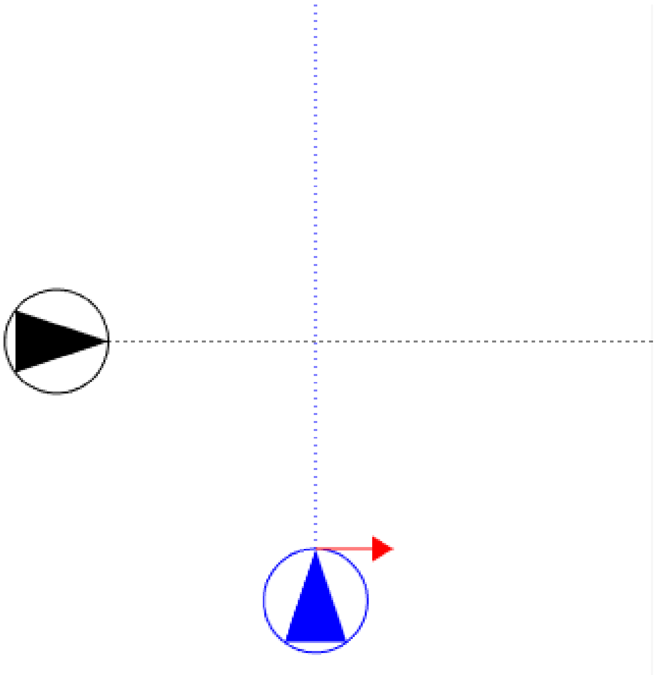
1. In a situation of suspecting collision with another crossing my intended path in a perpendicular manner (90 degrees), I turn slightly towards the point they're coming from. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

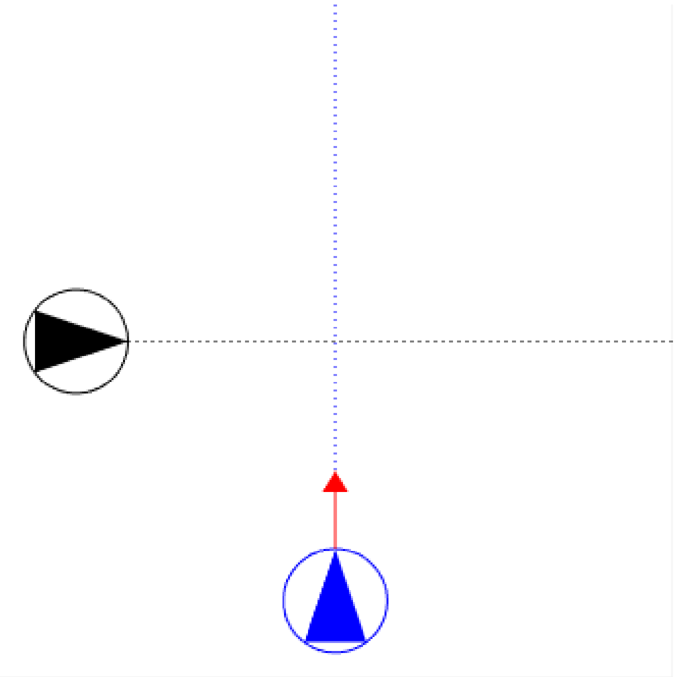
2. In a situation of suspecting collision with another crossing my intended path in a perpendicular manner (90 degrees), I turn slightly towards the point they're going to. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

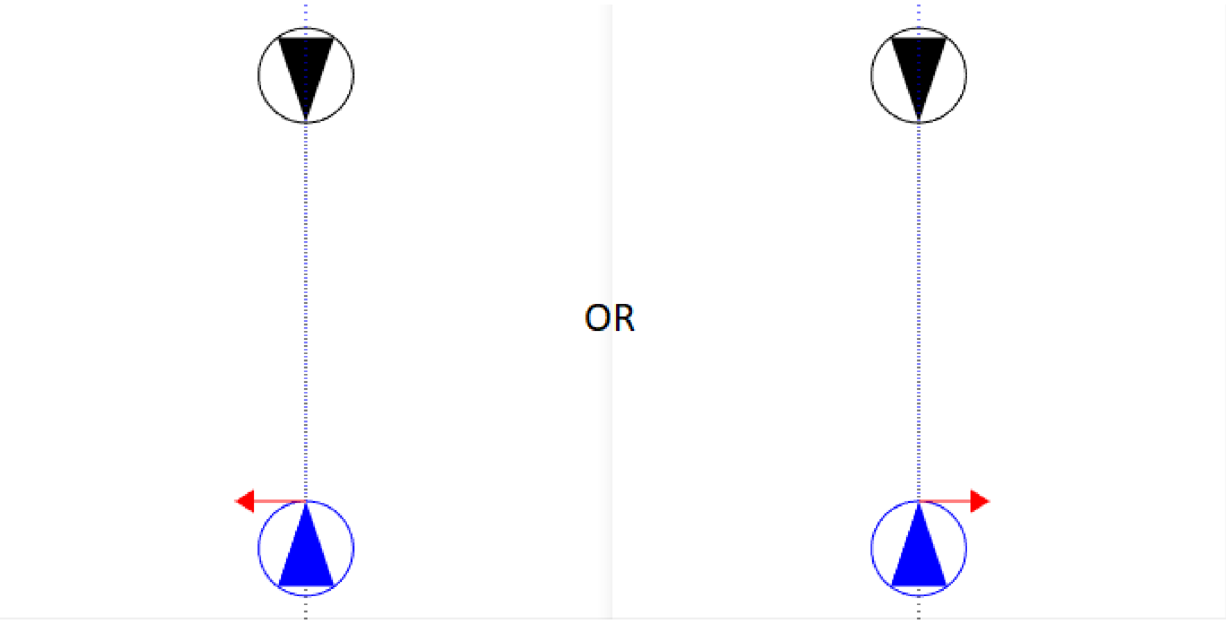
3. In a situation of suspecting collision with another crossing my intended path in a perpendicular manner (90 degrees), I keep going in my intended direction, hoping the other to avoid colliding with me. This behaviour is...



Mark only one oval.

- ( ) Absurd
- ( ) Non-ideal
- ( ) Neutral
- ( ) Reasonable
- ( ) Expected

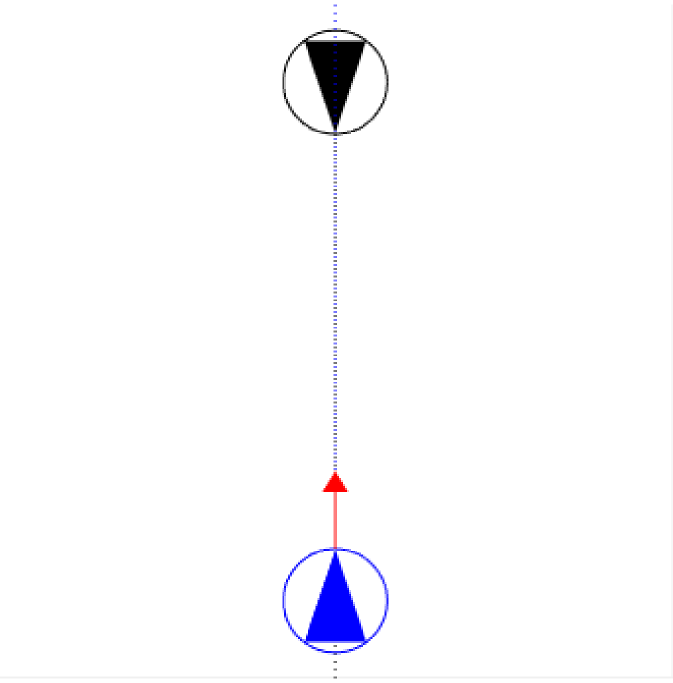
4. In a situation of suspecting a frontal collision with another heading towards me directly, I turn slightly towards the right or left, if possible. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

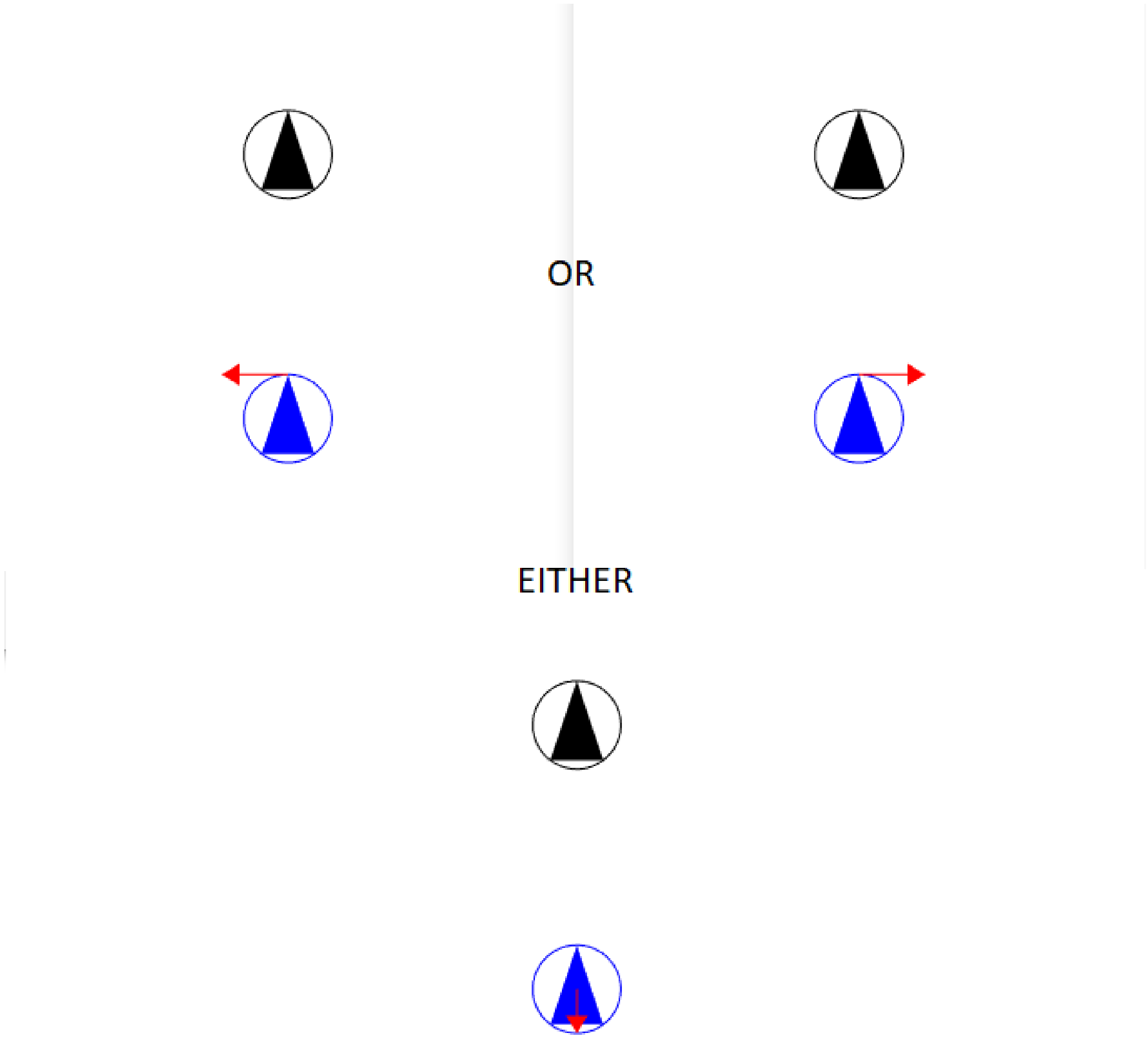
5. In a situation of suspecting a frontal collision with another heading towards me directly, I keep going my intended direction, hoping the other will avoid colliding with me. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

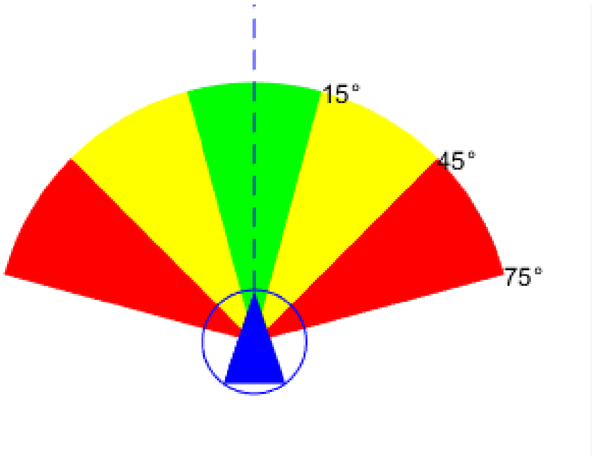
6. Noticing someone ahead of me moving slower than I do, going in the same direction, I would rather...



Mark only one oval.

- [ ] Keep my speed, moving either right or left to bypass this person eventually.
- [ ] Slow down, and keep myself behind this person.

7. How many degrees of deviation from your desired direction is too much, at which point you slow down or stop to await a clearing? Choose the closest alternative (choose the lower if directly inbetween).



Mark only one oval.

- 15 degrees
- 45 degrees
- 75 degrees

8. Adjusting my speed up and down frequently is frustrating and/or tiresome, and I prefer not doing it too often.

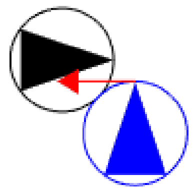
Mark only one oval.

- Yes
- No

Resolving Collision

This section treats resolving collisions after the fact that they have occured. Note that you do not need to physically collide with another person or obstacle for it to be considered a collision. Rather, you may have stopped, because going further *would* result in a physical collision.

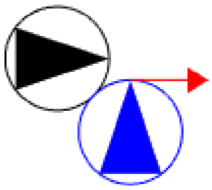
9. In a collision scenario where the one I have collided with was moving approximately in the perpendicular direction (90 degrees) relative to mine, I turn significantly towards the direction the other came from to get out. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

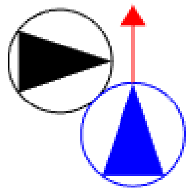
10. In a collision scenario where the one I have collided with was moving approximately in the perpendicular direction (90 degrees) relative to mine, I turn significantly towards the direction the other is going towards to get out. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

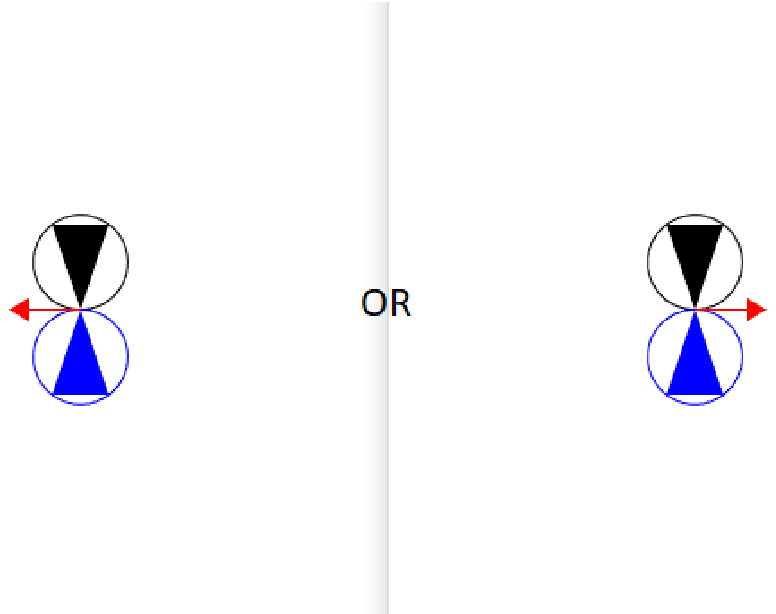
11. In a collision scenario where the one I have collided with was moving approximately in the perpendicular direction (90 degrees) relative to mine, I stay put until the other person has moved out of the way. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

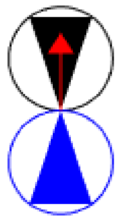
12. In a collision scenario where the one I have collided with was moving approximately in the opposite direction relative to mine, I turn significantly towards the right or left to get out. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

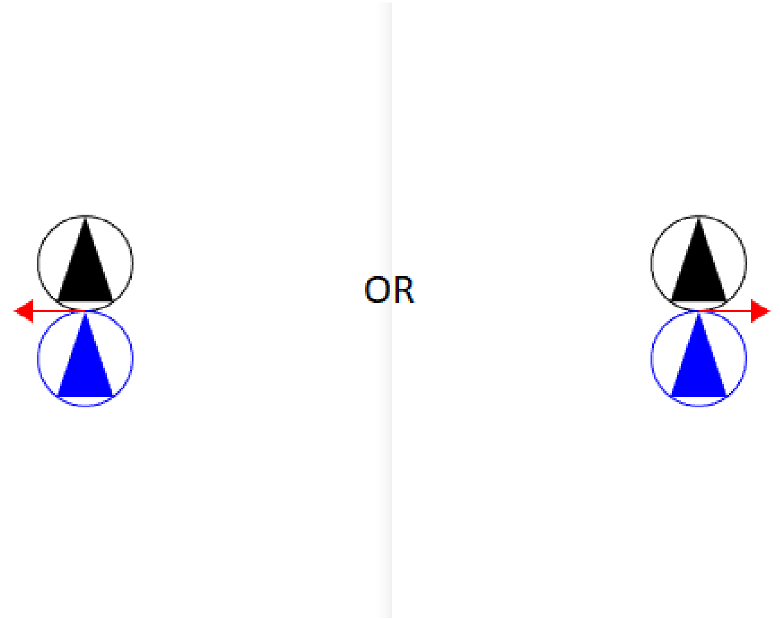
13. In a collision scenario where the one I have collided with was moving approximately in the opposite direction relative to mine, I stay put until the other person has moved out of the way. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

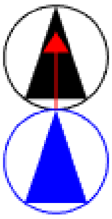
14. In a collision scenario where the one I have collided with was moving approximately in the same direction relative to mine, I turn significantly towards the right or left to get out. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

15. In a collision scenario where the one I have collided with was moving approximately in the same direction relative to mine, I wait until the other starts moving again. This behaviour is...



Mark only one oval.

- Absurd
- Non-ideal
- Neutral
- Reasonable
- Expected

Comments

16. Any comments you may have are welcome and appreciated. If you had trouble with a specific question, it would be very helpful to receive feedback on the matter. If you don't have any or don't want to, leave blank.

Appendix B

Detailed Specifications

B.1 The DevVec method

Because this is the first documentation for the DevVec method, it is described in closer detail in this section. Each variable that is used in Algorithm 1 is explained in table B.1.1.

Variable	Description
rotationFactor	Determines how significant an agent *a*will rotate
proximityVector, *frontVector*, *peripheralVector*	Avoidance vectors generated by corresponding detectors
avoidCongestion	A boolean which activates significant avoidance
speed	The speed of the agent
maxSpeed	The comfort speed of an agent *a*.
incSpeed	A small fraction of maxSpeed. Used to alter speed
wallDetected	A boolean which activates avoiding walls.
action	An enumerable. The values it can take are ”avoid”, ”avoidTowards”, ”avoidFrom”, and ”noAction”
threatObject	The active threat obstacle.
newDirection	The direction to rotate towards

Table B.1.1: Table of variables, along with appropriate descriptions.

Algorithm 1 The DevVec method

```
for all agents a do
    rotationFactor ← 1
    proximityVector, avoidCongestion ← ProximityDetector.Update()
    if ||proximityVector|| > 0 and speed > 0 then
        speed ← speed − incSpeed
        rotationFactor ← 3.0
    else if speed < maxSpeed then
        speed ← speed + incSpeed
    end if
    frontVector, wallDetected ← FrontDetector.Update(avoidCongestion)
    peripheralVector ← PeripheralDetector.Update()
    if ||frontVector|| > 0 then
        action ← peekAction(”front”)
        if !wallDetected then
            threatObject ← FrontDetector.getActiveFrontThreat()
            if a.name < threatObject.name then
                action ← cooperate(threatObject)      ▷ changes action if necessary
            end if
        end if
        if action = ”avoid” then
            newDirection ← rotateTowards(frontVector, rotationFactor)
        else
            newDirection ← followVectorField(rotationFactor)
        end if
    else if ||peripheralVector|| > 0 then
        action ← peekAction(”periphery”)
        threatObject ← PeripheralDetector.getActiveFrontThreat()
        if a.name < threatObject.name then
            action ← cooperate(threatObject)
        end if
        if action = ”avoidTowards” then
            newDirection ← rotateTowards(peripheralVector, rotationFactor)
        else if action = ”avoidFrom” then
            newDirection ← rotateTowards(peripheralVector, -rotationFactor)
        else
            newDirection ← followVectorField(rotationFactor)
        end if
    else                                              ▷ No threats detected
        newDirection ← followVectorField(rotationFactor)
    end if

    if !checkMaxRotation(newDirection) then           ▷ Does exceed the limit
        newDirection ← followVectorField(rotationFactor)
    end if
    applyRotation(a, newDirection)
    moveForward(a)
end for
```

B.2 The Gradient method

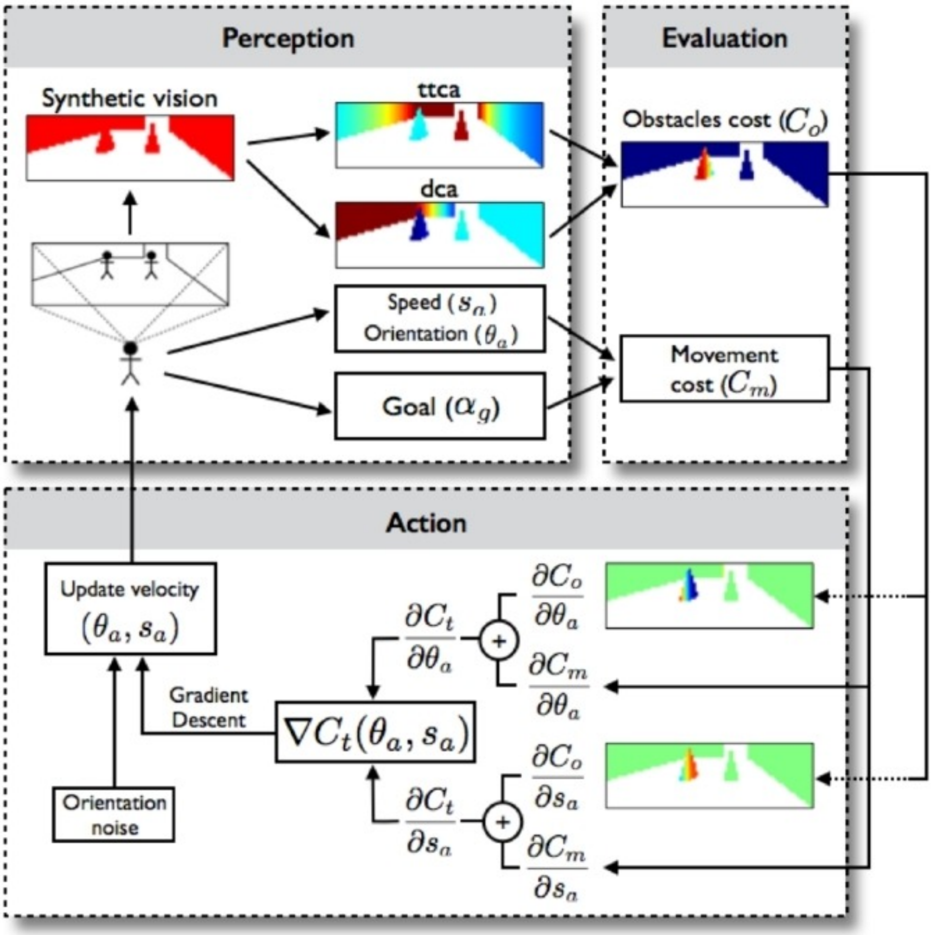


Figure B.2.1: A diagram of how the method operates, provided by the authors. More details can be found in the paper itself[5]

Symbol	Description
$\mathbf{p}_a$	Current position of agent a
$\mathbf{v}_a$	Velocity of agent a
s_a	Speed of agent a ($s_a = \|\|\mathbf{v}_a\|\|$)
θ_a	Orientation of agent a
α_g	Bearing angle with respect to goal g
$\mathbf{p}_{o_i}$	Current position of obstacle o_i
$\mathbf{v}_{o_i}$	Velocity of obstacle o_i
$\mathbf{v}_{o_i\|a}$	Velocity of obstacle o_i relative to agent a
$\mathbf{p}_{o_i\|a}$	Current position of obstacle o_i relative to agent a
$\text{ttca}_{o_i,a}$	Time to closest approach between agent a and obstacle o_i
$\text{dca}_{o_i,a}$	Distance between agent a and obstacle o_i at the closest approach

Table B.2.1: Table of symbol notation. Note that some from the original were omitted, due to being used in lower levels of abstraction than presented here.

To explain it briefly, the Gradient method[5] uses a perception phase, an evaluation phase, and an action phase, according to figure B.2.1. What the symbol notation means is explained in table B.2.1. With these clarifications given, the method works according to algorithm 2. Because of the more detailed description in the paper, that is all that will be provided here to give a overview of how it works..

Algorithm 2 The Gradient method

for all agents a **do**
    $\mathbf{p}_a, \mathbf{v}_a \leftarrow$ get_state(a)
    $camera \leftarrow$ set_up_camera($\mathbf{p}_a, \mathbf{v}_a$)
    $perc\_space \leftarrow$ render_environment()
    for all pixels $p_i \in perc\_space$ **do**
        if has_visible_obstacle($\mathrm{p_i}$) **then**
            $o_i \leftarrow get\_obstacle(p_i)$
            $\mathbf{p}_{o_i}, \mathbf{v}_{o_i} \leftarrow$ get_motion(o_i)
            $\mathbf{p}_{o_i|a}, \mathbf{v}_{o_i|a} \leftarrow$ relative_motion($\mathbf{p}_a, \mathbf{v}_a, \mathbf{p}_{o_i}, \mathbf{v}_{o_i}$)
            $ttca_{o_i,a} \leftarrow$ compute_ttca($\mathbf{p}_{o_i|a}, \mathbf{v}_{o_i|a}$)
            $dca_{o_i,a} \leftarrow$ compute_dca($\mathbf{p}_{o_i|a}, \mathbf{v}_{o_i|a}, ttca_{o_i,a}$)
            $(\frac{\partial C_{o_i,a}}{\partial s_a}, \frac{\partial C_{o_i,a}}{\partial \theta_a}) \leftarrow$ grad_pixel_cost($ttca_{o_i,a}, dca_{o_i,a}$)
            $perc\_space(p_i) \leftarrow (\frac{\partial C_{o_i,a}}{\partial s_a}, \frac{\partial C_{o_i,a}}{\partial \theta_a})$
        end if
    end for
    $\nabla C_o = (\frac{\partial C_o}{\partial s_a}, \frac{\partial C_o}{\partial \theta_a}) \leftarrow$ grad_obstacles_cost($perc\_space$)
    $\nabla C_m = (\frac{\partial C_m}{\partial s_a}, \frac{\partial C_m}{\partial \theta_a}) \leftarrow$ grad_movement_cost(Δ_s, α_g)
    $\nabla C_{t,a} \leftarrow$ grad_cost($\nabla C_o, \nabla C_m$)
end for
for all agents a **do**
    $\mathbf{v}_a \leftarrow$ adapt_agent_motion($\nabla C_{t,a}$)
end for

TTRITA-EECS-EX-2022:620