CHALMERS
UNIVERSITY OF TECHNOLOGY

Genetic Algorithms for human-like expressions in an autonomous vehicle

Master's thesis in Complex Adaptive Systems

Alberto Gosálvez White, Róbert Fjölnir Birkisson

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Genetic Algorithms for human-like expressions in an autonomous vehicle

Alberto Gosálvez White, Róbert Fjölnir Birkisson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden 2024

Genetic Algorithms for human-like expressions in an autonomous vehicle
Alberto Gosálvez White, Róbert Fjölnir Birkisson

Supervisor: Johan Tenghamn, Semcon
Examiner: Krister Wolff, Department of Applied Mechanics

Master's Thesis 2024
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The autonomous vehicle Will-e with its face visible and a representation of a randomly generated Behavior Tree, having being optimized evolving with Genetic Algorithms.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Genetic Algorithms for human-like expressions in an autonomous vehicle
Alberto Gosálvez White, Róbert Fjölnir Birkisson
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Advancing AI technologies have bridged the gap between science fiction and reality, and a future where advanced human-robotic interaction is common doesn't seem far away. While AI models are getting smarter in conversation and reasoning they are still missing a crucial piece in human communication, namely facial expressions.

In this project, we investigate the prospect of using a Genetic Algorithm with Behavior Trees to create the illusion of having emotions for an autonomous vehicle. The vehicle, called Will-e, determines on its own an appropriate emotional state and then expresses those emotions using LED-matrices in the form of eyes and a mouth. We implement these algorithms, train them, and test them in a populated setting and explore how people interact with the robot.

Keywords: Behavior tree, Genetic algorithms, autonomous robot, human-robot interaction, emotions, social robotics.

Acknowledgements

We would like to express our sincerest gratitude to our supervisor, Johan Tenghamn, for his invaluable assistance and guidance. We also extend our thanks to our examiner, Krister Wolff, for his insightful advice and support.

Our heartfelt thanks go to Semcon for the opportunity to work with their robot and for providing the necessary resources we needed. Finally, we are grateful to all participants in the experiment for their valuable feedback.

Alberto Gosálvez White & Róbert Fjölnir Birkisson
Gothenburg, June 2024

Contents

List of Figures

1
Introduction

Human-robot interaction has been a large research topic in the last few years, especially with advancing Artificial Intelligence technologies. Another focus has been on Autonomous Robots which can use some human inspired features to communicate with people.

The idea of adding emotions to these robots to signal their internal states seems like a next logical step and indeed it has been implemented and researched in various areas, such as in the service industry [2] and in human-robot collaboration [3]. Designing robot interaction patterns can however be tedious work and potentially prone to many biases or faulty ideas on how we humans want to interact with robots. An interaction pattern or design that evolves from human input could therefore be of great value by minimizing these biases.

1.1 Purpose & Objective

The primary objective of this project is to develop and test an algorithm that enables a small autonomous vehicle, *Will-e*, to simulate human-like emotions through its interactions. The main goal of the algorithm was to learn which expressions to display on the front of the vehicle making it fun for people to interact with it.

The algorithm we decided to create is based on the use of Behavior Trees (BTs) [4] in combination with Genetic Algorithms (GAs) with an objective of attracting people to the robot and keeping them engaged with it for as long as possible. The successful implementation of the algorithm demonstrates that it is possible to simulate engaging human-like emotions in autonomous vehicles, thereby validating our initial hypothesis that GAs combined with BTs can enhance Human-Robot Interaction. Other simpler decision-making problems could be solved using an approach similar to the one explored here.

1.2 Background

Human-Robot Interaction is a relatively new field of research, encompassing diverse areas such as perception methods, motion planning, cognitive models, and coordination strategies [5]. Emotions are particularly significant in deepening communication between robots and humans [6].

As already mentioned, the key part of our project was the algorithm that determines the expressions shown, which was based on BTs and GAs, but we also had to overcome other imposing challenges. We structured it so that we could separately handle the perception of the environment, the actions of the robot and the decision-making process.

A Behavior Tree (BT) is an extension of a Finite-State Machine and a mathematical model of a plan execution. They can create very complex tasks by switching between and combining simple tasks and so can simulate complex behaviors while still being easy to understand.

Genetic Algorithms (GAs) are learning algorithms, part of Evolutionary Algorithms [7], used for optimization of complex problems. They use biologically inspired methods such as *mutation*, *crossover* and *selection* to generate solutions which get iteratively higher in quality.

1.3 Context & Scope

This thesis was suggested and supervised by the company Semcon who had already modified a small one-person vehicle into a autonomous vehicle they called *Will-e*. They wanted to incorporate some facial features into Will-e in order to *"give life to it"*. We decided that an achievable goal for Will-e was to attract people and interact with them in places such as conferences and fairs.

As our thesis focuses on software, we constrained ourselves to only implement the absolutely necessary hardware changes needed for this project. Additionally, because training and testing with real people takes a long time, we had to implement some special strategies in order to shorten the training times.

1.4 Thesis Contribution

Our findings show people's reaction towards an autonomous vehicle showing expressions with various emotions. The method used to determine these expressions is an uncommon approach of combining BTs and GAs together, which can be used in other contexts if needed but are used here as a novel method of determining expressions and emotions for a robot.

2
Theory

Our project covers many different topics and is influenced by many diverse fields. Since it is about the development and experimentation of a social robot, it is composed of crucial elements which necessitate explanation, such as aspects of human-robot interaction, emotional psychology and the algorithms used for the creation of a sensible decision-making module.

2.1 Human-Robot Interaction

Human-Robot Interaction (HRI) is a multidisciplinary field that studies the relation and interactions between humans and robots. HRI research is comprised of contributions from diverse sources such as human–computer interaction, artificial intelligence, robotics, natural-language processing, design, psychology and philosophy.

Some of the general aspects of HRI research relate to methods for the robots to perceive humans or surroundings, e.g. through the use of cameras, speech recognition with natural-language processing models or any type of physical sensor used for the purpose of understanding their environment [8].

HRI has a large effect on various robotic technologies including industrial collaborative robots (COBOTS) [9] working with humans; autonomous driving; exploration, search and rescue missions; medical applications such as surgical robots [10], for rehabilitation [11] [12] or healthcare [13]; social applications like therapy robots [14], educational [15] and robotic toys that play with you and show emotions [16].

(a) Davinci, a surgical robot

(b) A COBOT, UniversalRobots

(c) Paro, a social robot

(d) Cozmo, a toy robot

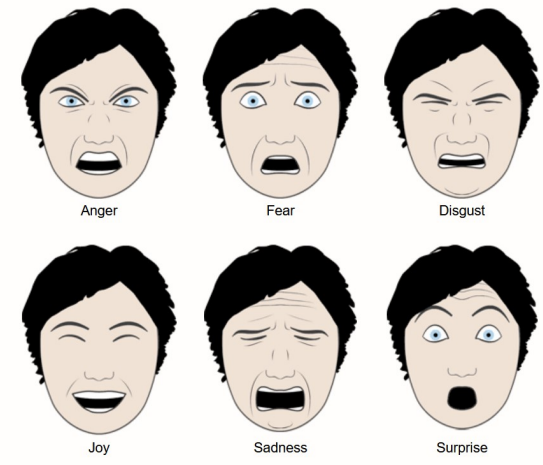
Figure 2.1: Examples of HRI applications in different robots. These robots either enhance human work or communicate with humans.

Social robots are autonomous robots that communicate and interact with humans, often mimicking human social expressions such as emotions, body language and behaviors. A large body of research has gone into exploring the interaction between humans and social robots, especially related to emotions [16].

2.2 Emotions in robots

Humans have a strong propensity to anthropomorphize anything, especially animated entities, therefore exploiting this bias for robotics is potentially very powerful. The ability to convey emotions convincingly could serve as a way to enhance human-robot interaction [2] [6]. The capability of robots to express emotions through facial expressions, body language, and other non-verbal cues is a growing area of research within the fields of robotics, artificial intelligence (AI), and human-computer interaction (HCI) and could help with human-robot collaboration in the future [3].

Expressions play a fundamental role in human communication, serving as a primary medium through which emotions are conveyed and social bonds are formed. Generally it is accepted that six basic facial expressions of emotions are universal: anger, fear, enjoyment, sadness, disgust and surprise. [17]. In theory all emotions can be formed as a combination of these basic ones.



(a) *Image taken from The Grimace project, an experiment to analyze how well cartoon expressions show their intended emotion.* [18]

Figure 2.2: The six basic facial expressions of humans that are commonly considered to be universally recognized no matter the culture.

Providing robots with the ability to express emotions, fundamentally, has the goal of affecting humans. Examples of reasons for simulating emotions in robots can be to warn humans about something, to catch their attention, to lift their mood in

an empathetic way, to joke when working on boring tasks or to induce a desired behavior the robot may want for the human [19]. An undesirable consequence of robotic emotions is the *uncanny valley effect* [1]; the hypothesis that the likeability of something resembling a human depends on how closely it does and how there is an area where the likeability drops sharply.

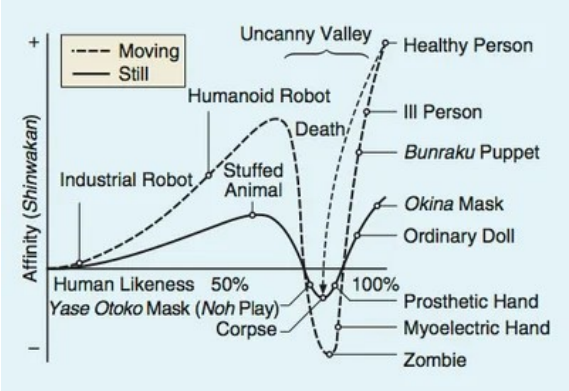


Figure 2.3: Original representation of the uncanny valley effect (Masahiro Mori, 1970)[1]. People are unnerved by things that resemble a human but are distinctly not human, like zombies or creepy dolls.

Relevant to this project, we implemented the basic universal faces on Will-e with addition of two more, a *neutral* face and a *euphoric* face which is more happy than *joy*. However, after implementing the faces we noticed that *fear*, *disgust* and *surprise* were all ambiguous i.e. easily misinterpreted, and so they were not used for this project. The reason for adding the *euphoric* face was to balance out the positive and negative emotions so that we had the same number of each, namely two negative, two positive and one neutral.

(a) Anger **(b)** Sadness **(c)** Neutral **(d)** Happiness **(e)** Euphoria

Figure 2.4: The facial expressions of Will-e used in this project. These expressions were created by combining different eye and mouth configurations and were essential for testing the response of human subjects.

(a) Surprise **(b)** Disgust **(c)** Fear

Figure 2.5: Unused faces of Will-e. When tested, people felt they were too ambiguous for our experiments.

2.3 Behavior Trees

A Behavior Tree (BT) [4] is a structured graph that determines when an autonomous agent should switch between tasks. BTs are an efficient way of creating complex behaviors that are modular and reactive, which are crucial properties in many applications. These properties have allowed BTs to spread to many branches of Artificial Intelligence (AI) and Robotics.

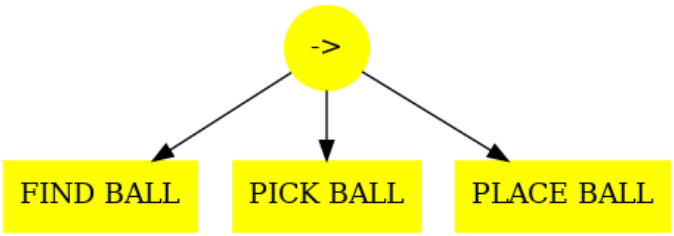


Figure 2.6: An example of a simple high level BT for a robot to carry out a task consisting of first finding, then picking and finally placing a ball.

2.3.1 Formal definition

A BT is a directed rooted tree where internal nodes are called *control flow nodes* and leaf nodes are called *execution nodes*. All control nodes have at least one child and all nodes have exactly one control node parent, except the root. A BT starts its execution by sending execution signals, called *ticks*, from its root node to its children at a given frequency. A node is executed only if it receives this tick. The child immediately returns *Running* to the parent, if its execution is under way, *Success* if it has achieved its goal, or *Failure* otherwise. In the classical formulation, there exists

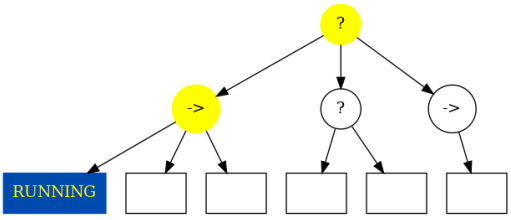
four categories of control flow nodes but only two will be explored here, *Sequences* and *Fallbacks*, and two categories of execution nodes, *Actions* and *Conditions*.

Sequences send ticks to their left-most child first and if the child returns *running* or *failure*, the Sequence returns the same. Otherwise, if the child returns *Success*, the Sequence sends ticks to the next child. In other words, a Sequence only returns *success* if **all** its children return *success*, *running* if any of the children return *running* and *failure* otherwise.

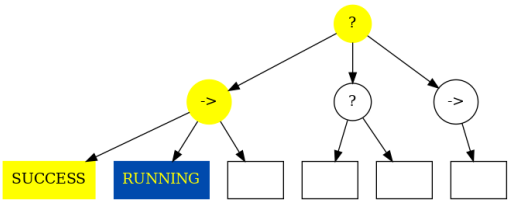
Similarly, *Fallbacks* send ticks to their left-most child first but return *success* if **any** of their children return *success*, *running* if any of their children are running and *failure* otherwise (if all their children return *failure*).

When *Actions* receive ticks they run some specified code, often performing operations or functions. They return *running* if the action is currently being performed, *failure* if the action fails for some reason and *success* if the action succeeds.

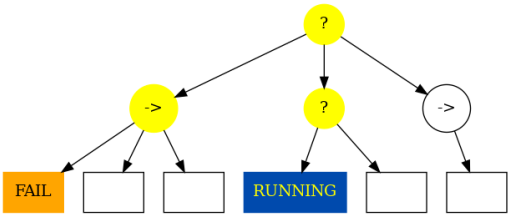
Conditions nodes immediately check a condition as soon as they receive a tick and therefore can only return *success* or *failure* if the condition is true or not respectively. Usually, the *Condition* nodes check the state of the total environment state, which can change quickly due to outside factors. These *Conditions* allow the BT to be highly responsive to its environment.



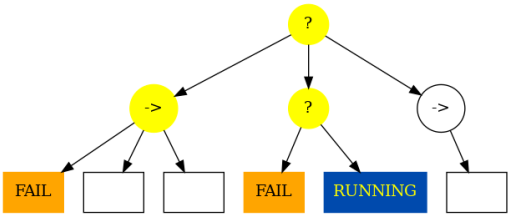
(a) One example of an execution of a particular tree where the first *action node* visited returns *running*, therefore it stops ticking there.



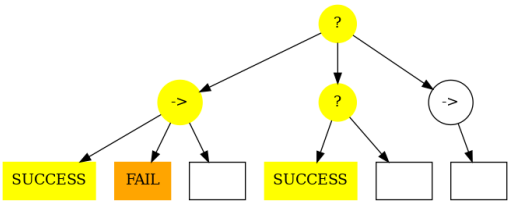
(b) Another execution on the same tree where the first *action node* is now a *success* and as its parent is a *sequence* it continues to the next sibling.



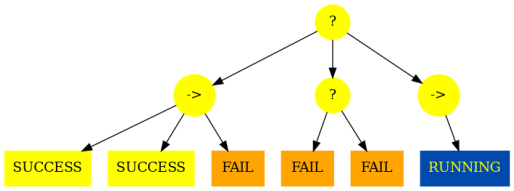
(c) Now the first *action node* was a *failure*. As its immediate parent is a *sequence* the children of this node are not visited but as its previous parent is a *fallback* the next child branch of this node is visited.



(d) Again, the first action *failed* in this execution and the action under the middle *fallback* node was visited. This time it returned *failure* and since its parent is a *fallback* node it continued to its sibling.



(e) In this execution, the first action *succeeded* and the second *failed* meaning the next *action node* visited was the one in the middle branch. That one succeeded and because all its parents are *fallbacks* the entire tree returns *Success*. The tree stopped executing even though no *running* node got visited. This evaluation of the tree didn't return any *action*.



(f) In this last example of an execution all the nodes were visited. The first branch ends up *failing* and that gets transmitted all the way to the root node. As this one is a *fallback* it visits the next branch and as this branch also returns *fail* it visits the next one. Finally the tree visits a *running* action node, so this evaluation returns the last action.

Figure 2.7: Examples of a tree evaluated at different times i.e. with different environment states. The evaluations do not depend on what a previous evaluation obtained, they only depend on the state of the environment at that moment.

2.3.2 Relevancy for this project

As stated before, Behavior Trees can produce complex and responsive behaviors, even with a low number of available actions such as with emotions in this project. Additionally, they provide a clear explanation for why a certain action was taken for an outside viewer, especially relevant when behaviors are being learned from scratch. In comparison, using a Reinforcement Learning algorithm for a similar objective produces hard to explain behaviors.

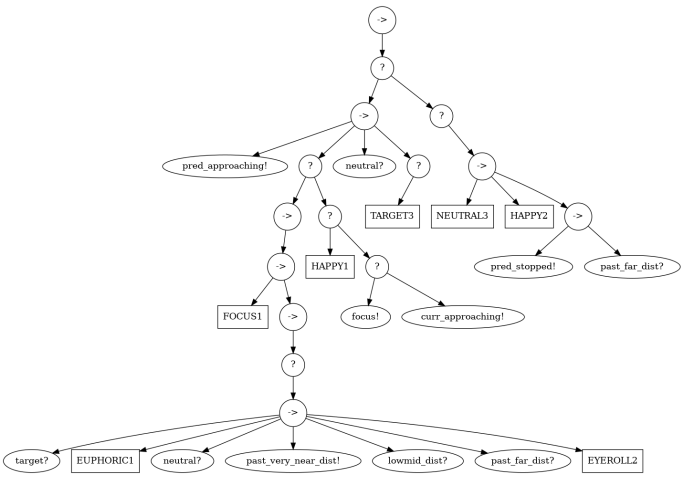


Figure 2.8: Example of a relatively complex Behavior Tree generated during one of our training sessions. The trees generated were often large with some unreachable branches so some pruning could be used for optimization.

2.4 Genetic Algorithms

Genetic Algorithms (GAs) are learning algorithms, part of Evolutionary Algorithms [7], used for optimization of complex problems. They use biologically inspired methods such as mutation, crossover and selection, i.e. heredity, to generate solutions which get iteratively higher in quality.

The process of GAs involves generating random solutions to a problem called *individuals* and checking how well they solve the problem. The individuals that solve the problem well i.e. have a high fitness, are then selected to create new individuals with some slight changes. This process has some degree of randomness to it and can be thought of as a type of stochastic gradient ascent with some clever biologically inspired methods to make the convergence to the best solution faster.

2.4.1 Components of GAs

GAs implement the process above using *chromosomes*, which are comprised of *genes*, to encode different individuals. These *chromosomes* are often programmed as arrays of numbers or characters where each element is a *gene* and corresponds to some element or input of the fitness function, depending on the problem. A GAs runtime is split into several distinct parts:

- **Fitness checking:** Each individual is checked for their fitness, this can be through simulation, a specific function or any process that generates a number that determines how good an individual is at solving the problem.

- **Selection:** After getting each individuals fitness, a selection is performed to get some of the better ones to repopulate. Common selection methods include Tournament selection and Roulette Wheel selection. An important note is that the selection is random and biased towards better fitness but not deterministic.

- **Crossover:** After selection, new individuals are formed through reproduction of the previously selected individuals. To do so, new individuals are formed by mixing two chromosomes through a process called crossover. A part of chromosome A is switched out for a part of chromosome B, leading to two new chromosomes that are a combination of both A and B.

- **Mutation:** In nature, mutation is an error while copying chromosomes but these errors play a crucial role in providing evolution with new material to work with. Therefore, with a small random chance, some of the newly formed chromosomes are changed a little bit with completely random (but also valid) elements.

- **Replacement:** The final step in the GA procedure is replacement where the new individuals are added into the population and the old population possibly taken out. Then the whole procedure can start again with the new population

until a good enough individual, and therefore solution, is found.

2.4.2 Benefits of GAs in our context

Genetic Algorithms have been shown to combine well with Behavior Trees [20] and limit the search space of BTs somewhat, especially where BTs are being learned from scratch. The methods of GAs with the node structure and modularity of BTs allow for extensive exploration and fine tuning in a large problem space without the search space becoming excessively large.

Additionally, using GAs with BTs allows for easy analysis of the generated BTs meaning they do not act as a black box and can easily be modified if needed to better suit the objective. By allowing BTs to be evolved using GAs it is possible to create complex behaviors faster than by hand.

Comparing this to learning BTs through classical Reinforcement Learning where BTs are learned through trial and error, the search space becomes increasingly large over time and requires more computing power. Of course, Reinforcement Learning can be used with BTs to great effect when the problem space is more limited but that does not apply in this project.

2.5 Communication

Many parts of the software were split up over a few different computers, each having their specialized domain, e.g. one computer handled image processing and detection, another handled control flow for the vehicle itself and a few micro-controllers handled their respective hardware components were also included. Two different communication protocols were used, namely ROS and MQTT.

2.5.1 Robotic Operating System (ROS)

ROS [21] is a set of software libraries and tools that make it easier to build robot applications. Namely it helps run separate processes or microservices and makes it simple to communicate between them, whether they are running on the same machine or not.

ROS implements this process by having one main information stream that all data is sent to and programs can listen for specific data. This made transferring large and small data between different programs (or "nodes") trivial and enabled us to create programs that focused on one single thing, e.g. one program focused on retrieving the newest frame from a web-camera and publishing it to the data stream.

2.5.2 Message Queuing Telemetry Transport (MQTT)

Classically, MQTT [22] is a standard messaging protocol for the Internet of Things (IoT). It is a lightweight publishing/subscribing messaging transport that is ideal for small embedded devices. This made it perfect for the various micro-controllers present in the project, which listened to commands from the control programs on the larger computers. In addition, MQTT also allowed us to send signals from other devices, such as buttons or phones, to the larger computers to control Will-e or provide feedback.

2.6 Related work

Emotions in robots has been a popular subject of studies since robotics was in its infancy and their effect on humans in a service setting was studied in the paper by Staphanie Hui-Wen Chuah and Joanne Yu, 2021 [2]. There they investigated how emotional robots influence potential consumers' feelings by using sentiment analysis and machine learning and they found that the development of emotional robots had implications for human-robot communication. The robots could give non-verbal cues and signals to human to communicate something or change the context of a statement.

Additionally, they found that some emotions, like surprise or some instances of happiness, were often misunderstood by human participants which shows that conveying accurate emotions is difficult and often very subjective. Another study by Di Fu, Fares Abawi and Stefan Wermter, 2023 [3] further showed that this was the case but additionally showed that when working together to solve a task, humans performed better when working with a robot that showed emotions than the same robot without emotions. All this is to say that adding emotions to a robot that will interact with humans could be beneficial if done right.

As autonomous vehicles get more popular, more research has gone into communication between self-driving cars and pedestrians e.g. when a pedestrian needs to cross a street. A paper by Chia-Ming Chang, Koki Toda, Daisuke Sakamoto and Takeo Igarashi, 2017 [23] studies how adding eyes to a car affects interactions between pedestrians and self-driving vehicles. Based on their evaluation, pedestrians could make faster decisions whether to cross the street and felt safer when the autonomous vehicle had eyes and made eye-contact with them.

3
Methods

A number of features were implemented for this project ranging from hardware modules which had to be assembled and integrated to software algorithms with complex structures.

3.1 Implemented features on Will-e before project

Will-e had a number of features already implemented from previous projects, such as a LIDAR sensor, controllers for the car features, microphone and integration with Chat-GPT among others but only features relevant to this project will be discussed here.

(a) **(b)**

Figure 3.1: (a): Ruidi Cabin Car X9, the mobility scooter that Will-e is based on. (b): The LED matrix used as Will-e's mouth. It was attached into the grill of the mobility scooter.

A number of computers where integrated into Will-e, communicating using ROS (Robot Operating System) and performing various tasks depending on their position in the vehicle and sensors attached to them. One of these sensors was a web camera positioned and oriented to look through the window and onto the environment in front of the vehicle. The camera streamed flat images to one of the main computers, an Nvidia Jetson edge computer, where the detection algorithms were applied to the images.

Another important feature was the LED mouth that is integrated to the front of Will-e. It is a 64x32 LED matrix connected to an ESP32 micro-controller that has been programmed to display a few different expressions (see Figure 2.4 and Figure 2.5). It runs a MQTT subscriber that listens to commands coming from the other parts of the system.

These features provided a good base to start the project on and allowed for easy expansion in other features.

3.2 Installation of eyes

Similar to the mouth, two 64x64 LED matrices were provided to be used as eyes but they had not been programmed or installed into the vehicle yet. Another ESP32 was used as a microcontroller and a program for the eyes was written in the C-based Arduino programming language.

The graphics for the eyes constitutes of a white background, two overlapping circles resembling the iris and the pupil, and two black areas resembling the higher and lower eyebrows. These features are parametrized and controlled via a MQTT client on the ESP32 to change gazing direction, color, eyebrow angle and more, allowing them to display a wide array of expressions and providing personality to Will-e.

Figure 3.2: The LED matrices used as eyes for Will-e. Two 3D-printed brackets were used to attach them to where the headlights were. For each eye, we used two concentric circles to represent the iris and pupil and a line to represent en eyelid serving as an eyebrow.

The eyes are often referred to as the "windows to the soul" and therefore they are an essential component for the communication of emotions [23]. Just adding this simplified design we opted for, gave Will-e a noticeable increase of appeal. The inclusion of eyes capable of focusing and following a person also allowed it to have a way of indicating attention, showing that it is engaged in the interaction, something crucial as a lack of eye contact might suggest distraction or disinterest.

In addition to focusing and following people, we created a series of small actions and animations for variety in the experiment. We observed how small subtle movements helped increase the attention people pay to Will-e, such as looking around when not focusing on anyone, but it was clear that staring at people was the most effective

way to maintain the attention as most people thought Will-e was malfunctioning when it was not looking at them.

3.3 Person and pose detection

In order for Will-e to react to and interact with people, it was necessary to implement a robust method not only capable of detecting humans but also their poses. We chose YOLOv8n-pose [24] because it is specifically tailored for identifying and mapping human body keypoints in images. The YOLOv8 Pose models are pretrained on the large COCO dataset.

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle

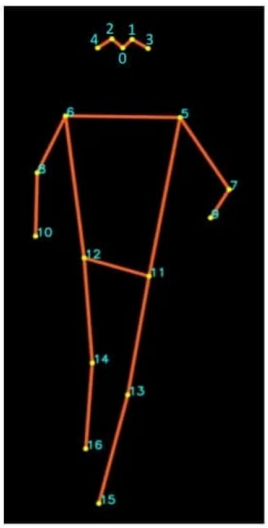


Figure 3.3: Keypoints given by YOLOv8n-pose. The returned matrix included these keypoints and a confidence score for each one. The keypoints were vital for the distance estimation and gesture detection in Will-e.

The output of the pose estimation is a set of points that represent the keypoints, as seen in Figure 3.3, on the objects in each frame along with a confidence score for each point. To keep track of the detected objects we implemented a simple centroid-tracking algorithm which allows us to apply a unique ID to each tracked object as they move around in a video.

For every frame in our video stream we compute an object's centroid and then determine if we can associate the new object's centroid with an old object's centroid. To accomplish this process, we compute the Euclidean distance between each pair of existing object centroids and input object centroids.

The primary assumption of the centroid-tracking algorithm is that a given object will potentially move in between subsequent frames, but the distance between the centroids for frames F_t and F_{t+1} will be smaller than all other distances between

objects.

Therefore, by choosing to associate centroids with minimum distances between subsequent frames we track those objects. In case that there are more objects detected than tracked, we simply register these new objects by assigning them a new ID and adding them to our list of tracked objects. When a tracked object cannot be matched to any detected object, it is removed from the list of tracked objects, i.e. when a person leaves the frame.

In addition to the tracking, we also predict where some key features of the person will be in the next frames. This is accomplished by fitting the data of the positions of those key features in a few previous frames and predicting where it will be in the next frames.

As we only had access to a 2d camera for the detection, we had to estimate how far in the depth axis the persons are. To achieve this estimation in a sensible, fast and accurate enough manner for our objectives, we look at how wide the head of the person is; the wider the head is, the closer to the camera we assume it is. We decided to use the keypoints on the head, namely the eyes, the ears and the nose, because the maximum distance between the visible features barely changes except when moving closer or farther from the camera.

All this information acquired during the detection process serves to create snapshots of the state environment, where attributes are recorded for a behavior tree to check when it is run, such as how far a person is from a desired target position, their direction of movement, if it is looking at the robot and if the person is gesturing.

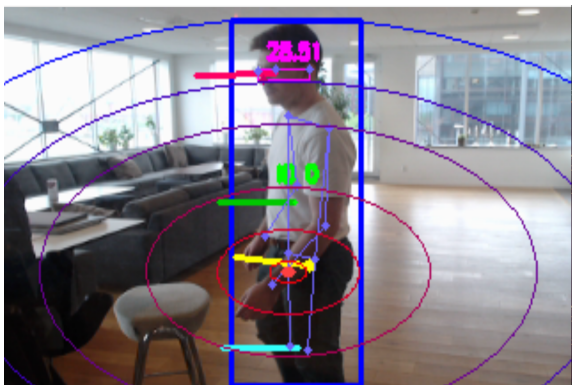


Figure 3.4: Example of our detections, showing the bounding box, keypoints of the subject, predictions for the movement of some features and the control point calculated using the width of the head. The concentric ellipses are the limits of the different area levels used for the decision-making process of the robot, their center is a target position where we desired the subject to go.

3.4 Genetic algorithm with Behavior Trees

In order to use Behavior Trees (BTs) with a Genetic Algorithm (GA), the structure of a single BT had to be encoded into a simple chromosome for the GA and suitable algorithms for crossover, mutation and the fitness had to be found.

3.4.1 Behavior Tree encoding

To get a chromosome representation of a BT, each node of the BT was encoded as a small string and parentheses were used to denote the parent-child hierarchy [25]. Each of the node representations were then used as the genes for the chromosome, which was represented by a list of these genes. The following encodings were used for each of the different nodes:

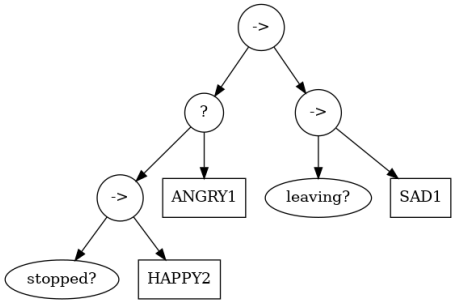


Figure 3.5: An example of a tree with the encoding: `["s(", "f(", "s(","c_stopped?", "HAPPY2",")", "ANGRY1", ")", "s(", "c_leaving?", "SAD1", ")", ")"]`

- **Control nodes:** Sequence nodes were represented by the string `"s("` and Fallback nodes by the string `"f("`. To complement them, a closing bracket `")"` is required in order to have a valid chromosome. The first element of a chromosome is always one of these control nodes.

- **Condition nodes:** There are a few different types of condition nodes that check different aspects of the environment. One of the types are **distance** conditions that check the distance from the agent to Will-e, e.g. `c_far_dist?` and `c_mid_dist?`. Another type checks the agent's current **direction**, if it is approaching or leaving and another checks the agent's predicted direction, e.g. `c_curr_approaching?` and `c_pred_stopping?`. The last type of conditions check the current **state** of Will-e and what emotion he is performing, e.g. `c_happy?` and `c_focus?`. Additionally, condition nodes have a decorator at the back of their respective string, which by default is the question mark `"?"`. However, they can also have the exclamation mark `"!"` which means the entire condition is flipped, e.g. `c_far_dist?` (is agent far away) vs. `c_far_dist!` (is agent not far away).

- **Action nodes:** Action nodes are denoted by the name of an action and a duration (either "1", "2" or "3"), e.g. `ANGRY2`, `TARGET3`. When activated by a behavior tree, the named action is performed for a specific amount of time denoted by the duration decorator.

3.4.2 Crossover and mutation

The tree structure of a BT meant that crossover and mutation in a GA had to have special cases. For **crossover**, a non-length preserving crossover was used where a random node (except the root node) was selected in two trees and then those nodes were switched, along with all of their children if they had any.

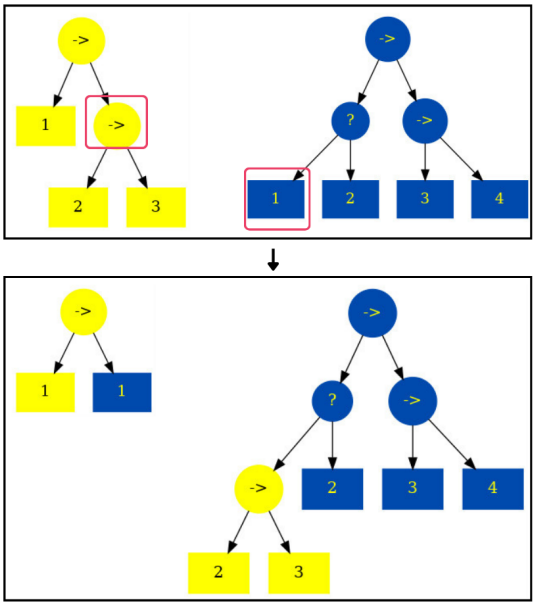
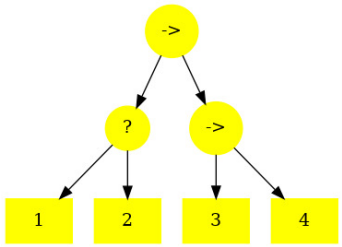


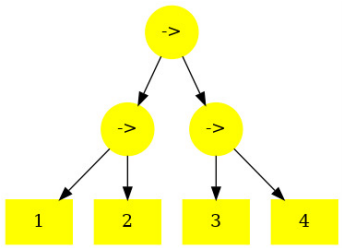
Figure 3.6: An example of the non-length preserving crossover that was used in the Genetic Algorithm. Two random nodes (except the root nodes) are selected in the trees, then they and all their children are switched.

There were a few different types of **mutation** to account for different changes:

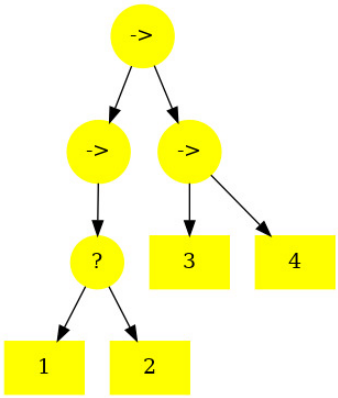
- **Adding control node:** Control nodes can be added anywhere and they change their siblings into their children.
- **Adding leaf node:** Leaf nodes can be added anywhere.
- **Deleting control node:** If a control node is deleted (except the root node) their children go up in the hierarchy.
- **Deleting leaf node:** A special case happens if a leaf node is deleted and it's the last child of a control node and the control node is deleted as well.
- **Changing operators or durations:** Operators on condition nodes and action nodes can be changed to any other valid operator.
- **Nodes changing to a node of the same type:** Control nodes can change into other control nodes and leaf nodes can turn into other leaf nodes.



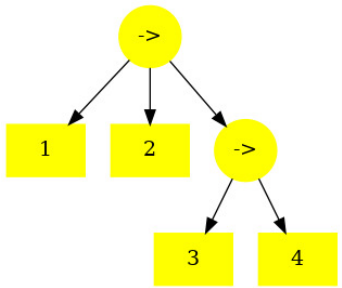
(a) Original behavior tree



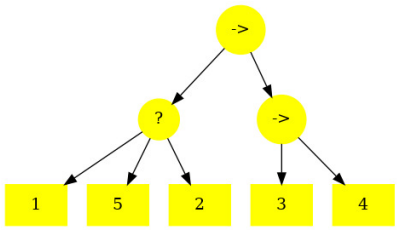
(b) Fallback node switched to a sequence node



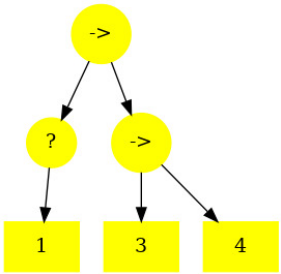
(c) Add control node



(d) Delete control node



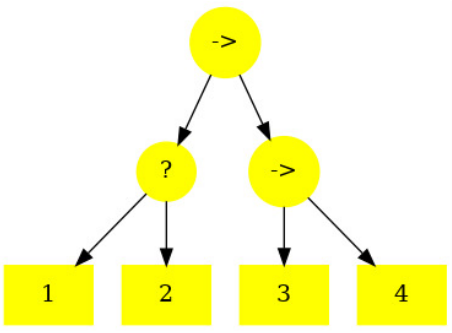
(e) Add leaf node



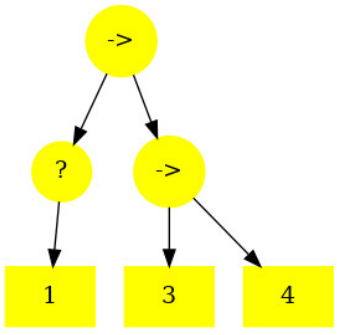
(f) Delete leaf node

Figure 3.7: Examples of various mutations being applied to the original tree in Figure 3.7a. Having a mutation like this that can slightly change the structure of the tree is important for exploring the whole solution space.

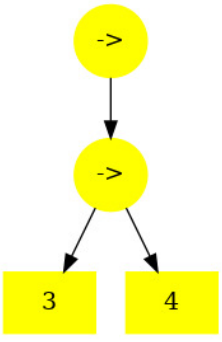
In figure 3.8 we can see an example of a special case where all the leaf nodes of a control node are deleted and therefore the control node is deleted as well.



(a)



(b)



(c)

Figure 3.8: Example of all children of a control node being deleted. In 3.8a we see the original chromosome, in 3.8b we see that leaf node #2 has been deleted and in 3.8c leaf node #1 has been deleted as well as the parent fallback node.

3.4.3 Additional methods and parameters

In addition to crossover and mutation we used a *Tournament selection* method and *Elitism*. The Tournament Selection involved randomly selecting some amount of individuals (Usually 4 for training in the simulations and 2 for training with humans because of population size) and allowing the individual with the highest fitness can win according to a probability (e.g. 60% of the time). If that individual doesn't win, the individual with the next highest fitness can win using the same probability and so on and so forth until the last individual, which always wins if it gets to it. The winner of this tournament is selected for the crossover process.

Elitism involves taking the individual with the highest fitness in a generation and inserting it unchanged into the next generation when repopulating. This allows for smoother change and helps retain good individuals when the fitness is non-deterministic, as is the case in this project.

3.4.4 Fitness function

The objective we wanted to achieve for this project was that the Will-e robot would learn to attract people to itself and keep them interacting for as long as possible. For this purpose the fitness function chosen prioritizes the length of interaction and that the person is located close to a desired position.

Each chromosome is evaluated on a set number of agents, each providing a score calculated by adding together a few different rewards and penalties based on the performance. The final fitness for the chromosome is then the mean of the scores obtained for each agent. To calculate the score for each agent, a number of factors had to be considered, including the agent's position and duration of interaction, number of actions sent from the BT, length of the chromosome and feedback from human agents.

To reward BTs that managed to make the agents stay in the target area, the field of view was split into different areas based on the distance to the target center. Detecting the agent position on every time step, a certain number of points is rewarded depending on which area level it is, the closer to the target the higher the reward. When leaving the field of view a penalty is considered, so that an agent starting close to the target area and then leaving immediately gets a worse score than one starting further away but staying in the field of view for a considerable long amount of time.

When an interaction session ends, either because the agent leaves the field of view or the time limit is reached, the accumulated number of points per levels is divided by the duration of the session to get an average score for each time step in the session. The decision to use this average number of points instead of just the accumulated points was taken to balance the scores of agents starting in different areas.

In addition, the level that was visited the most by the agent in the session also

contributes to the fitness score. Without taking this in consideration, the evolved BTs would attempt to make the agent stop in place no matter where it started just to maximize the time spent in the field of view, but when adding a considerable reward for spending the most time in the closer levels to the target, the evolved BTs would instead want the agent to move to the target as fast as possible and then stay there in place.

Another important factor for the fitness is related to the number of different actions the BT decides to realize. This number is tracked and then divided by the runtime in order to get an average frequency of changes during the interaction session. That frequency is then mapped to a penalty where too low and too high of a frequency gives a high penalty, as can be seen in equations 3.1 and 3.2. The reason for this penalty was to find a suitable frequency of changes in emotions, so that they were not changing every single frame or not changing at all.

$$P_f = 16(f - 0.5)^4, f > 0.5 \tag{3.1}$$

$$P_f = (f - 1)^{50}, f <= 0.5 \tag{3.2}$$

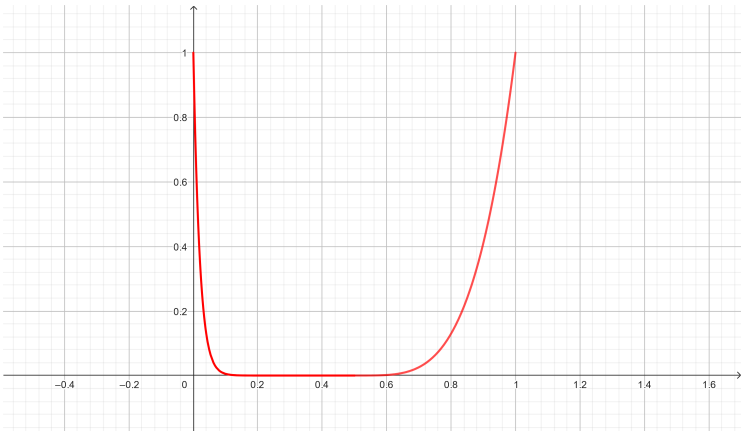


Figure 3.9: Graph of equations 3.1 and 3.2 for the frequency penalty. Note the large spikes when approaching a ratio of 0 (when there's no change in the face) and a ratio of 1 (when the face changes every tick).

When training in real life the possibility of using more factors was taken into account. Humans would sometimes unpredictably attempt to interact with Will-e gesturing with their hands. Counting the duration of these gestures was a good way to measure the engagement during a session, and as such it was included when calculating the fitness. The feedback humans would give was also valuable information and it was collected using an array of buttons where they could easily rate their interaction. The feedback contributes considerably to the score but without completely overshadowing the other passive factors.

Lastly, two penalties are applied, one if the length of the chromosome being tested is longer than a 100 nodes and the other if the agent leaves the interaction before a certain amount of time.

Putting everything together, we get the formula:

$$F = R_l + R_c + P_f + R_g + R_h + P_l + P_e \tag{3.3}$$

with:

$$\begin{aligned} R_l &= \text{Average reward for the distance levels} \\ R_c &= \text{Reward for the most common distance level} \\ P_f &= \text{Penalty for the frequency of changes} \\ R_g &= \text{Reward for the duration of gestures} \\ R_h &= \text{Reward from the feedback of humans} \\ P_l &= \text{Penalty for the length of the chromosome} \\ P_e &= \text{Penalty for agents leaving early} \end{aligned}$$

3.5 Simulation of expected human behavior

Since evaluating every chromosome in real life is an extremely time consuming task and humans are prone to fatigue, especially when testing bad or boring trees, we created a simulator to help us train some preliminary chromosomes that would later be introduced in the population when training in real life.

The simulator tries to replicate how a person would react to the different expressions that Will-e is capable of doing. When starting the simulation, a particle is spawned in a randomly uniformly distributed position inside the canvas workplace. The particle starts with a random speed, direction and attention level and for every time step of the simulator it has a chance of changing these parameters.

When evaluating a chromosome, for every step of the simulation, a snapshot of the environment state in the simulator is taken and then fed into the BT encoded in a chromosome to get the action Will-e should do. That change in the state of the robot is then passed to the simulator, so that the particle can notice it and react according to a probability depending on its attention level.

The simulation is similar to a random walk but with some key differences. Different behaviors for the particle are defined and for each of the expressions that Will-e can have, a list of expected behaviors is coupled. Depending on different probabilities, when an expression is detected by the particle, it will use the expected behavior or not. These behaviors consist of making changes to the speed, direction and attention.

- When the particle is on *standard* behavior, it will usually keep its speed and direction because the changes are calculated using a random normal distribution, centered on zero with a standard deviation of a third of the maximum expected change for speed and direction respectively.

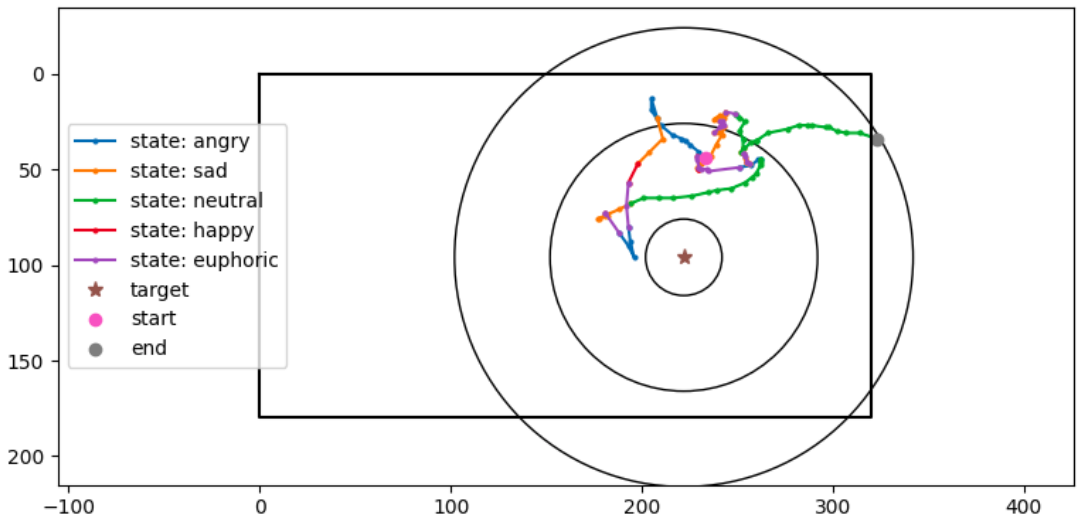
- When on the *stopping* behavior it will change the speed by subtracting from it a value calculated by a random normal distribution centered on its current speed value. In effect, the particle will slow down while still changing the direction normally.

- Other behaviors make the particle be attracted to certain specified positions, such as *target* and *middle*. The changes are done to the direction by calculating the angle difference between its current direction and the vector towards the objective, then using this value as the center of a random normal distribution to get the change of direction.

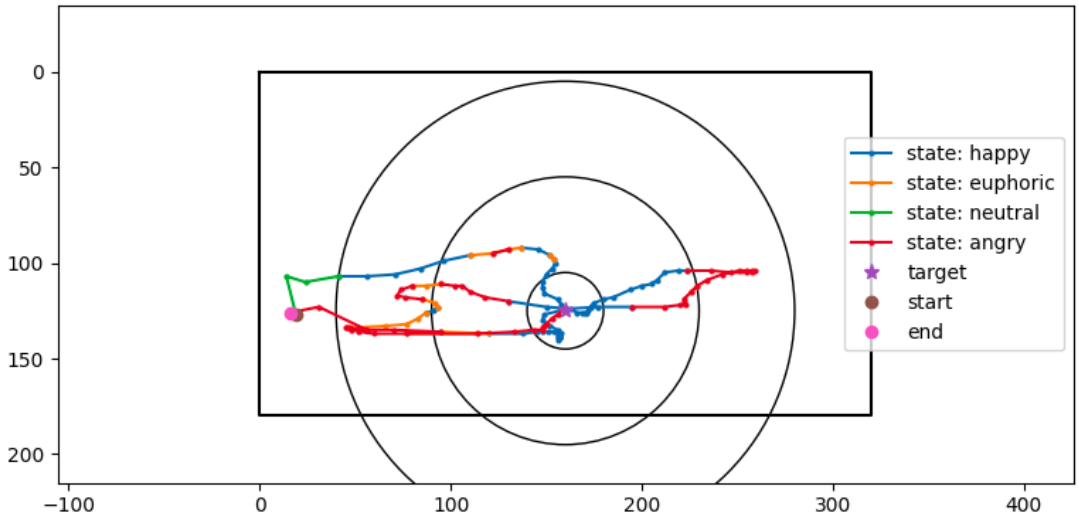
- The *opposite* behavior acts in a similar way to *target* and *middle*, by centering the random distribution at 180 degrees but only the first time the behavior is activated when a new detection is noticed, afterwards it just does the standard behavior. This effectively reverses the direction of the particle.

The attention of the particle is also calculated in the simulation, it may increase when changes in the state of the robot happen or when certain behaviors get activated e.g. eyes focusing on the agent. The robot making too many state changes in a short span or not making enough will decrease the attention of the particle. This reflects how in real life the robot needs a balanced approach when making changes so that the people don't get fatigued or bored too soon.

As the reaction of a person wouldn't be immediate from noticing an action of Wille, the simulator also has a delay when changing the behavior of an agent after it notices a change in the state of the robot.



(a) Example of a trajectory of a random simulation. Notice how when the state of the robot changes the particle may not necessarily do the expected behavior.



(b) Example of a trajectory made by a person during a real interaction with Will-e. Notice how the simulation is similar to the behaviors in real-life.

Figure 3.10: Comparison of a simulated agent to the reactions of a real person when interacting to Will-e. We can observe how both move in similarly random ways and how the changes of the robot affect them in a similar manner. In these examples we can clearly see that when Will-e is *angry* both the human and the simulated agent change their direction after a delay. In addition, it can also be observed that when the human was paying attention and Will-e was *happy*, the movement slows down, as we predicted for the expected behaviors for the simulator.

4
Results

4.1 Evolution with the simulator

The Genetic Algorithm (GA) provides a finite population of chromosomes for each generation. Each of the chromosomes has to be evaluated for the selection process and the repopulation of the next generation to take place. This evaluation is the most challenging part of the project, as it depends on the unpredictable nature of human actions. The simulation of human behavior was created with the intent of providing an educated representation of how a real person is expected to act, to be able to evaluate a Behavior Tree (BT) on many different artificial persons or agents.

4.1.1 Setup and training with the simulator

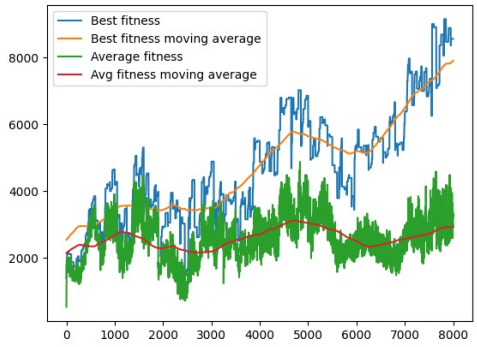
When evaluating the BT encoded in a chromosome we test it using a large number of different agents, with initial positions randomly uniformly distributed around the workspace area. Each of the agents will move by itself and react to the actions the tree decides to perform in each step of their simulation. Every simulation provides information that is used to determine the fitness of the chromosome, such as how many times the robot performed actions, the places the agent visited and for how long it was there. The final fitness of each chromosome is the mean of the scores of all agents tested on it in this generation as explained in Section 3.4.4.

To be able to objectively compare the chromosomes, we use the same set of agents for all the chromosomes in each generation. Each agent in the set has a random seed for their random number generator, and the initial conditions of each agent are the same for the chromosomes in a generation, including the speed, direction, position, attention and behaviors. As the simulations run for each chromosome, the same agent will behave differently, unless the visited parts of the trees are identical between chromosomes.

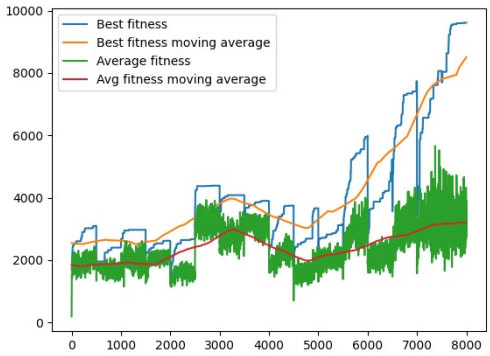
As the set of agents has to be finite, we run into the problem that eventually the GA will start overfitting to these particular agents. To solve this problem, it's needed to diversify the agents used for the evaluation. We achieve this by replacing some agents in the set with new ones having different attributes at a regular interval of generations.

The effects on the fitness of a chromosome when testing on different sets of agents can be very noticeable, e.g. the fitness of a chromosome that was large with a cer-

tain set of agents can sink by the exclusion of a single agent. While this method of changing the agents can seem aggressive and defeating the purpose of the GA by creating fluctuations on the best fitness per generation, we observe how the overall trend of the best fitness keeps increasing in the same manner as in a traditional GA.



(a) Graph of a training session where 20% of the agents used for evaluation are changed every 50 generations



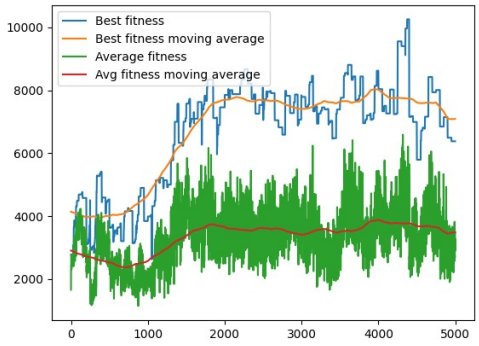
(b) Graph of a training session where all of the agents used for evaluation are changed every 500 generations.

Figure 4.1: Graphs of the best fitness, the average fitness and rolling averages of both for two simulation training sessions with different agent-rotation parameters. Note the large dips every 500 generations in the graph on the right.

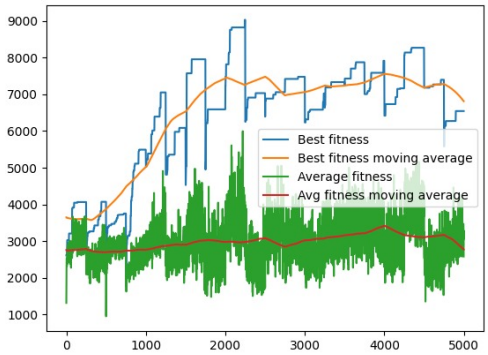
4.1.2 Results using the simulator

Looking at the results from the training sessions in the simulator, the best generated BTs mostly shared some characteristic behaviors although with completely different tree structures. They usually displayed positive expressions when the subject was close and in the middle of the frame and negative emotions when the subject was too far or was moving away, in addition they didn't change too fast nor too slow. This was in line with our fitness function as the BTs got a higher reward for successfully guiding agents to the middle of the screen, therefore this step helped in finding adequate BTs with which to continue training in the real world.

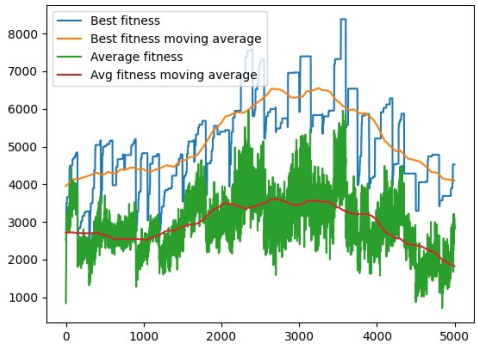
As for the training sessions themselves, it is interesting to analyze how the evolution progress differs when using modified strategies. The amount of test agents and how frequently they are rotated are particularly influential in the evolution progress. As mentioned in Section 4.1.1, the rotation is necessary so that the training doesn't overfit to a specific set of agents, but a rotation too frequent or too aggressive (changing too many agents at the same time) can also have detrimental effects. Having too frequent rotations meant that the GA did not have time to fit to the agents and as such, did not learn anything at all. Too aggressive rotations introduced large dips in the fitness and often the better chromosomes were lost to these large dips, meaning that these training sessions often got stuck in local optima.



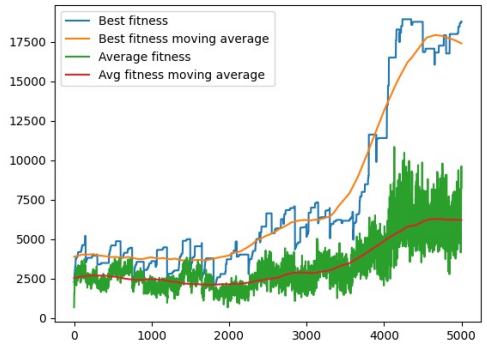
(a) A typical training run. Notice the steep incline at the start and then the long plateau, common in Genetic Algorithms. 20% of the agents were rotated out every 50 generations.



(b) Another typical training run but with all agents being rotated every 250 generations. Notice how the fluctuations are less frequent but the dips are more noticeable.



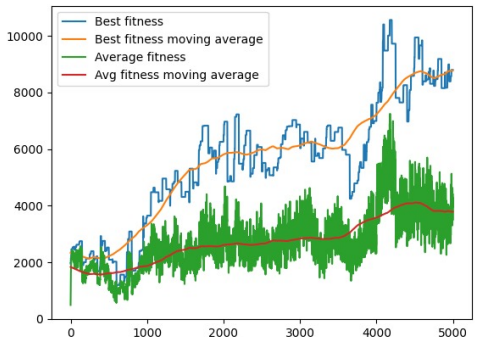
(c) A failed training run. Notice the big drops when rotations happen, they were being too frequent and aggressive and eventually the better chromosomes were lost to bad agents.



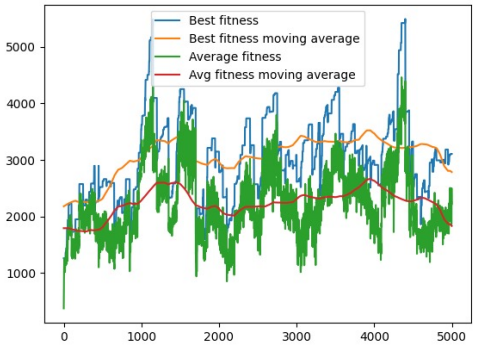
(d) A training run that got one of the highest fitness. Notice the rapid escalation when good chromosomes where created and how the rotation didn't destroy them.

Figure 4.2: Various training runs with different parameters. Most of the graphs tend to go up but are heavily affected by the parameters of each run. The magnitude and frequency of the fluctuations also vary depending on these parameters.

In addition, the training runs were heavily affected by the randomness of the agents they were tested on. Testing each chromosome on a large number of agents largely eliminated these factors at the cost of computing time so a middle ground had to be found. For most training sessions, we chose to evaluate each chromosome on 100 different simulated agents per generation, this mitigated the randomness of the agents while still having a high performance, with each training run of 5000 generations taking around 8 hours. However, in some training runs, the random nature of the agents in some generations changed the fitness so much that bad chromosomes, that for example were largely unchanging and only displayed one expressions, dominated and pushed the other chromosomes out.



(a) A graph showing the typical jumps and plateaus often observed in GAs when a good chromosome is first generated.



(b) A graph showing a failed run where the randomness of the agents did not allow for an escalation on the fitness to occur.

Figure 4.3: An example of how randomness affects the outcome of the training. The outcome in figure 4.3a was a common occurrence in our training runs because of the high number of agents used but because this number is finite, edge cases such as the one shown in figure 4.3b can happen. The smaller number of agents the more likely these edge cases are to appear. Both training runs used the same parameters where they rotated 20% of the agents every 50 generations. Note the scale of the fitness in both graphs.

The crossover and mutation parameters in the GA are also factors that noticeably affected the performance of the evolution. Higher crossover and mutation encouraged exploration of new behaviors but at the cost of possibly destroying good chromosomes. A balance between exploration and exploitation is necessary for a successful training run.

For the simulated training sessions, a crossover rate of 60% and a mutation rate of 10% were used. A greater emphasis on exploration was put in the simulated environments due to the possibility of evaluating on a greater number of agents while being able to compare different chromosomes with the same agents, as opposed to real life scenarios where these options are impossible.

4.2 Evolution in real life

As stated earlier, being able to evaluate the many different BTs the evolution creates is challenging, but especially so in a real-life scenario. Humans behave in unpredictable ways, the noisy environment is uncontrollable, opinions when rating a session of interactions are completely subjective and can be contradictory, but the main problem is the high number of evaluations needed to achieve objective and fair fitness scores for the chromosomes.

4.2.1 Setup of the training in real-life

Will-e was tested in two main scenarios, one at the Semcon headquarters and the other at the VECS fair in Gothenburg. The first scenario was in the main hall of the Semcon headquarters, where many people passed every day. In addition to the passive ways that the fitness was calculated there was also an array of five buttons installed in order to allow people to rate their interactions and affect the training. People would then come and play with Will-e and rate how they felt it performed.

Finally, as this was a high traffic area, a lot of people didn't interact with Will-e and walked past and so we developed a functionality that disregarded these agents in the training, even though Will-e would still attempt to attract them. As an example, after a week long training we had around 700 interacting humans, with only 35 giving feedback using the buttons and around 1500 passers-by.

Figure 4.4: The setup for training at the Semcon headquarters. Note the array of five buttons for feedback from human agents next to Will-e, this feedback function was vital when training with people.

The second scenario was at the VECS fair, a very populated conference, where Will-e was at the Semcon booth in the exhibition hall for two days. Although many people interacted with Will-e, most people passed by and some stopped in front to speak to other people without paying attention to Will-e, which heavily skewed the training.

Because of the high variance of the human agents each of the chromosomes needed to be tested on several agents to be evaluated fairly. We explored two different approaches. The first involved testing the same chromosome a few times in a row

before using the next one but we observed that seeing the same behavior promoted fatigue in people, especially when evaluating a subpar tree. In an attempt to remedy this, the second approach was tried, where each of the BTs is evaluated with the same amount of people as before but the trees are rotated on every interaction session, so that no chromosome is used twice in a row. Although this introduces an issue where a person can get two very different behaviors from Will-e if they leave its field of vision and come back, which could be problematic when giving feedback for the interaction.

4.2.2 Results from real-life scenarios

At the Semcon headquarters, several long training sessions were run, around a week long each. The data recorded included the location of the control point of the main subject, the current expression and eyes state of Will-e, locations of various key-points on the person and any feedback received with the buttons.

Initially, before there was a way for human agents to provide direct feedback, we observed that Will-e got stuck in a local optima. The better scoring trees mainly displayed the sad emotion and did not react sufficiently to changes in the environment because people were more likely to stay in front of Will-e in an attempt to lighten his mood. However, even though this accomplished the objective of attracting and keeping people in front of Will-e, it wasn't really satisfactory behavior, especially since people wouldn't want to interact with Will-e again afterwards.

Once the feedback buttons were installed Will-e's behavior became more complex and gratifying. Now that the human agents could give their opinions, better interactions that felt more real were rewarded and simple interactions were punished. In addition, interactions were overall more positive, the BTs that resulted mostly in negative-feeling emotions got worse scores than without this feedback. These effects were felt even though only 4.5% of interacting agents gave feedback using the buttons.

Figure 4.5: The button array used to send feedback to Will-e during training. Five Shelly Button 1 were used with a custom 3d printed case. They sent a message through MQTT to the training software.

We can see how the different expressions of Will-e affected the human agents when reading the recorded data. To analyse it, we checked the position of the control point of each person that interacted with Will-e and we looked at how it changed

in the following frames every time Will-e changed expressions. By observing the actions of the humans in a sufficient number of frames we can deduce how they reacted to an expression change.

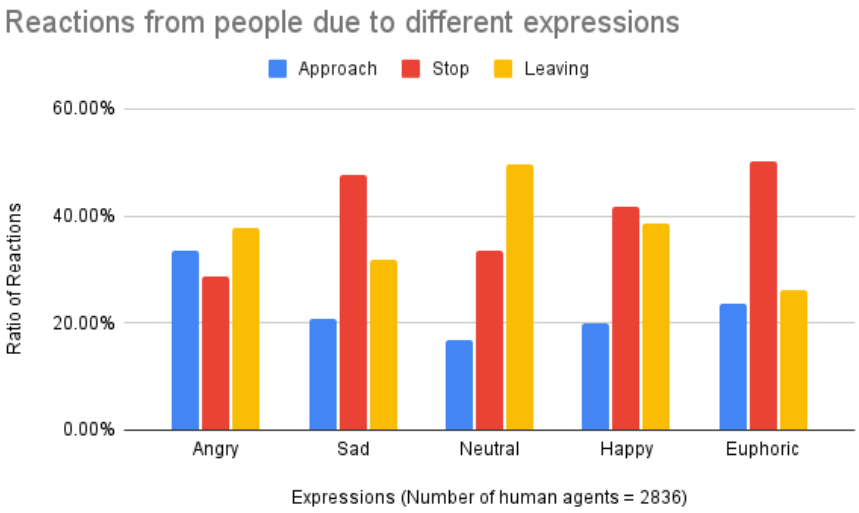
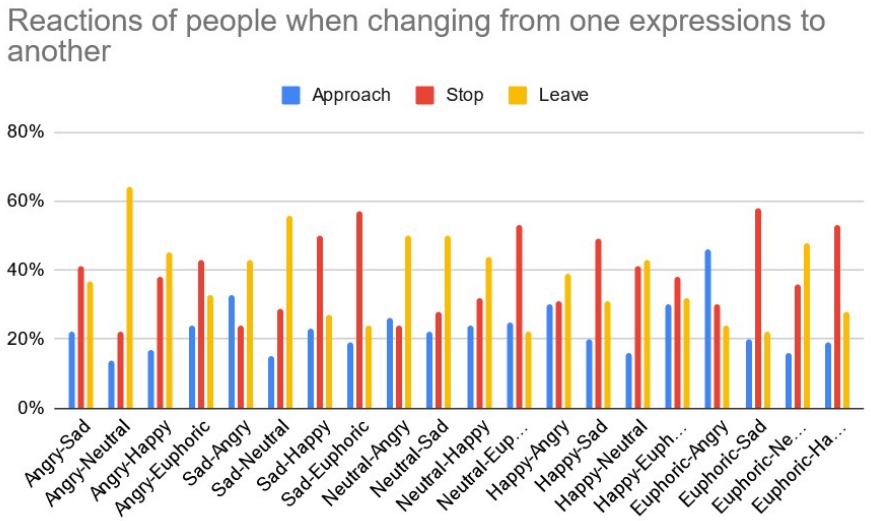


Figure 4.6: Ratio of reactionary movement due to different expressions. After an expression change the next 10 movement states were counted and then divided by the number of total states after the new expression.

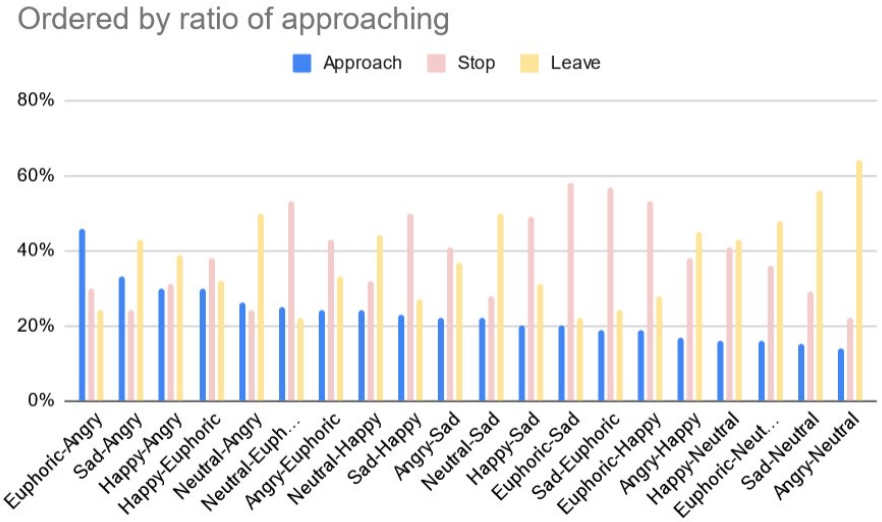
Looking at Figure 4.6, after Will-e became *angry*, people were more likely to approach him compared to the other emotions. It also was the expression that made people move around the most, either approaching or leaving. A possible explanation is that being angry could be making people want to approach and explore trying to help Will-e improve his mood or to see what is affecting him, as this is a very powerful emotion not often seen in robots.

Another interesting thing of note is that it seems the emotions *sad* and *euphoric* evoked similar responses in humans, mainly favoring stopping. The reasons for this could be many but our hypothesis, based on our observations, is that people stop when Will-e is sad because they think they did something wrong but they also stop when Will-e is euphoric because they feel they did something right or are standing in the correct place. In addition, it seems the more negative of the two emotions was slightly more likely to make people leave.

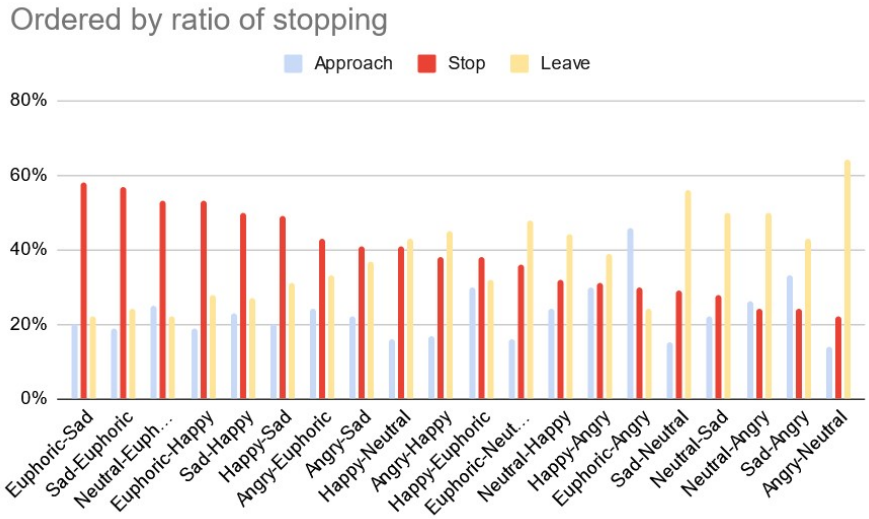
In contrast, the emotion *happy* does not seem to evoke as strong of a response from human agents. In fact, we believe it acts as a middle ground between *neutral* and *euphoric*. While the response to *neutral* is mainly to ignore it and leave, people are slightly more likely to stop with *happy*, but not as overwhelmingly as with *euphoric*. We theorize that *happy* acts more as a *neutral* face because people are used to seeing happy faces on robots, with that being one of the reasons the response to *sad* is more noticeable than that of *happy*.



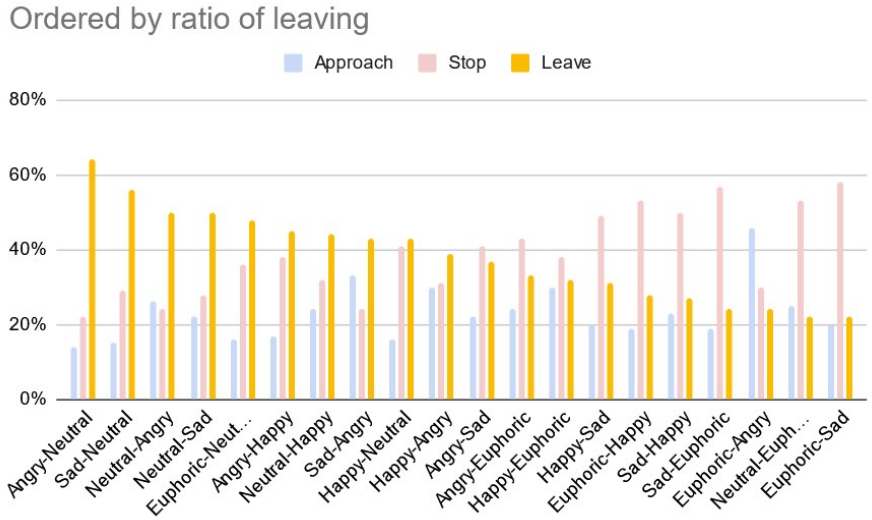
(a) A graph using the same data set as in Figure 4.6 plotting the ratio of movement reactions of people but with an additional breakdown of the previous state of the robot. It's difficult to get to a conclusion when the data is as dense as this, so we shall focus on each reaction separately in the following graphs. When ordering the state changes by the ratio of the human reactions it is easier to visualize the data.



(b) On one hand, we can observe that it was more probable for people to *approach* when changing from any expression to *angry* than any other expressions, although it was only the most probable response for the extreme case of going from *euphoric* to *angry*. On the other hand, changing to *neutral* rarely made people approach. This seems to suggest that big displays of emotions attract people to Will-e.



(c) Looking at the columns where *stopping* is the most likely reaction, we can see that it happened when changing to *sad*, *euphoric* and *happy* but changing from *euphoric* to *sad* and vice versa elicited the strongest stopping responses. This seems to support our hypothesis that going from a positive emotion to *sad* seems to signal that something is wrong and so people will stop and going from *sad* or *neutral* to a positive emotion, like *happy* or *euphoric*, seems to signal that the person did something right and so they will also stop, thinking Will-e wants them where they are now.



(d) When ordered by *leaving*, we can see that *neutral* is commonly involved when a person is leaving. The *neutral* face was used as the default face when no one was interacting with Will-e, so it could be that people felt that when Will-e changed to neutral that the robot was disinterested in them or that it was ignoring them.

Figure 4.7: An in-depth analysis of how Will-e changing from an expression to another evoked different reactions in people. We can clearly observe that the different emotional changes evoked distinct kinds of reactions.

5
Conclusion

5.1 Project conclusion

The goal of our thesis was to prove that our method is a viable way to create solutions for even an ambiguous objective such as *giving life* to a robot. For Will-e we wanted to provide a way for it to have a personality that manages to attract and make people want to interact with it. Our chosen method to solve this task was an interesting take using Genetic Algorithms (GAs) and Behavior Trees (BTs). The results were quite good, managing to complete our goal while also providing a modular method that can be easily adapted to other objectives.

Figure 5.1: Will-e interacting with a human. People liked to come up, move around and use various gestures like waving or crouching down. Even with simple expressions and gestures people were invested with Will-e's behavior.

The combination of BTs and GAs produced responsive and novel behavior in the autonomous vehicle Will-e. A special characteristic of BTs is that they can be read by people unfamiliar with the system and show that there is a complex algorithm behind Will-e's behavior. In contrast to other algorithms, BTs do not act as a black box and they can be easily analyzed and modified if needed, allowing for flexibility

to achieve a desired outcome. By allowing the trees to be evolved with GAs it is possible to construct complex behaviors faster and more reliably than by creating trees by hand, which depending on the objective can be clunky and not match their expected function. GAs are very useful to optimize a problem and find a more optimal solution, and although a definite objective was impossible for our case study it was still possible to achieve optimized solutions that behaved reasonable well.

As previously mentioned, an early problem we ran into is the scope of the solution state for this project, which made training with only humans unfeasible because of time constraints and human decision fatigue. We created a simulator that replicated simple human behavior and trained thousands of generations in order to get adequate BTs which we then used as starting individuals when training with humans. This was a success as the pre-trained individuals already exhibited complex and adequate behavior. Further evolution with them proved difficult as we trained in a highly populated and uncontrolled area but allowing people to provide direct feedback to the training helped immensely. The final behaviors of Will-e didn't have a big improvement from the pre-trained ones with the simulator because the evaluations take too much time in real life and many are needed so that they are fair. Despite that, we saw slight optimisation which implies that with more time better behaviors could be found.

To reach our conclusions, we collected data of the humans' reactions to the various expression of Will-e. We observed some interesting and unexpected patterns that challenged our preconceived assumptions. Some examples being how when expressing an *angry* emotion, people were more likely to approach Will-e compared to other emotions, how being *sad* made people more likely to stop, or how *happy* is often interpreted as a *neutral* state for a robot.

All in all, using GAs with BTs produced behavior that completed the objective we set out in the beginning, we managed to create a method for a robot to be able to learn by itself how to attract and keep people engaged. This kind of approach can be used for a lot of diverse tasks and objectives to help find optimal results.

5.2 Future work

By design, we limited the amount of faces to a few distinct emotions in order to simplify the training and avoid ambiguity. The expressiveness of the robot could easily be expanded by adding more faces, physical actions such as wiggling the wheels and moving back and forth, more eye animations or other types of communication, such as sound or text.

In addition, longer training could allow for more optimized results, creating BTs that interact better with humans. More data could also allow for a more exhaustive analysis of human-robot interactions, which is interesting in a lot of applications.

Furthermore, the technology developed here could be used to solve other problems,

for example, using the faces to guide people to specific places or do some tasks, communicating with pedestrians and catching their attention in a non-intrusive way. Problems not involving the face or humans could also be solved, such as navigating and exploring an area. This method is very modular and can be made to suit any number of different problems.

Bibliography

[1] Masahiro Mori, Karl F. MacDorman, and Norri Kageki. The uncanny valley [from the field]. *IEEE Robotics Automation Magazine*, 19(2):98–100, 2012.

[2] Stephanie Hui-Wen Chuah and Joanne Yu. The future of service: The power of emotion in human-robot interaction. *Journal of Retailing and Consumer Services*, 61:102551, 2021.

[3] Di Fu, Fares Abawi, and Stefan Wermter. The robot in the room: Influence of robot facial expressions and gaze on human-human-robot collaboration, 2023.

[4] Michele Colledanchise and Petter Ögren. Behavior trees in robotics and AI: an introduction. *CoRR*, abs/1709.00084, 2017.

[5] Cakmak Maya Mutlu Bilge Huang, Chien-Ming. Adaptive coordination strategies for human-robot handovers. *Robotics: Science and Systems*, 2015.

[6] Jaime Banks Brad A. Haggadone and Kevin Koban. Of robots and robotkind: Extending intergroup contact theory to social machines. *Communication Research Reports*, 38(3):161–171, 2021.

[7] Matthias Wahde. *Biologically Inspired Optimization Methods*. WIT Press, 2008.

[8] Ciro Natale. *Physical Human-Robot Interaction*, pages 1716–1724. Springer International Publishing, Cham, 2021.

[9] Abderraouf Maoudj Abdelfetah Hentout, Mustapha Aouache and Isma Akli. Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017. *Advanced Robotics*, 33(15-16):764–799, 2019.

[10] Patrick Probst. A review of the role of robotics in surgery: To davinci and beyond! *Missouri medicine*, pages 389–396, 2023 Sep-Oct.

[11] O'Kane J. Aggogeri F, Mikolajczyk T. Robotics for rehabilitation of hand movement in stroke survivors. *Advances in Mechanical Engineering.*, 2019.

[12] Edwin Daniel Oña, Juan Miguel Garcia-Haro, Alberto Jardón, and Carlos Balaguer. Robotics in health care: Perspectives of robot-aided interventions in clinical practice for rehabilitation of upper limbs. *Applied Sciences*, 9(13), 2019.

[13] MacDonald B. Broadbent E. Robinson, H. The role of healthcare robots for older people at home: A review., 2014.

[14] Serna R.W. Yanco H.A. Begum, M. Are robots ready to deliver autism interventions? a comprehensive review. *Int J of Soc Robotics*, page 157–181, 2016.

[15] Tony Belpaeme, James Kennedy, Aditi Ramachandran, Brian Scassellati, and Fumihide Tanaka. Social robots for education: A review. *Science Robotics*, 3(21):eaat5954, 2018.

[16] Hannah R. M. Pelikan, Mathias Broth, and Leelo Keevallik. "are you sad, cozmo?": How humans make sense of a home robot's emotion displays. In

Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, HRI '20, page 461–470, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Paul Ekman. An argument for basic emotions. *Cognition and Emotion*, 6(3-4):169–200, 1992.

[18] The grimace project. `http://experiment.grimace-project.net/`. Accessed: 2024-05-29.

[19] Verena Nitsch and Michael Popp. Emotions in robot psychology. *Biological Cybernetics*, 02 2014.

[20] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. A survey of behavior trees in robotics and ai. *Robotics and Autonomous Systems*, 154:104096, 2022.

[21] Ros - robot operating system. `https://www.ros.org/`. Accessed: 2024-04-03.

[22] Mqtt: The standard for iot messaging. `https://mqtt.org/`. Accessed: 2024-04-03.

[23] Chia-Ming Chang, Koki Toda, Daisuke Sakamoto, and Takeo Igarashi. Eyes on a car: an interface design for communication between an autonomous car and a pedestrian. In *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '17, page 65–73, New York, NY, USA, 2017. Association for Computing Machinery.

[24] Ultralytics pose estimation. `https://docs.ultralytics.com/tasks/pose/`. Accessed: 2024-04-03.

[25] M. Edin. Learning stationary tasks using behavior trees and genetic algorithms. `https://www.diva-portal.org/smash/get/diva2:1449937/FULLTEXT01.pdf`, 2020.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY