Linköping University

Advanced Project Course - AI and Machine Learning

TDDE19

Autonomous Station Master

Authors:
Oscar Melin
Elin Thorgren
Rasmus Rynell
Chetan Kandya
David Norell

January 5, 2023

1 Introduction

This paper explores one possible way of creating what we call an "interactive intelligent system". To achieve this type of system, we combine a set of artificial intelligent techniques in such a way that enables the system to not only interact with a person, but also to do so in what we consider an intelligent way. The project that this paper is based on was conducted at Linköping University as a collaboration between Linköping University's department of "Artificial intelligent and interacting computer systems" (AIICS), and the Swedish transport administration.

To consider the system an "interactive intelligent system", it should be able to perform some set of actions autonomously, in response to some set of interactions. The following actions are considered to be indicative to this end:

1. Detect if someone is interacting with the system.
2. Identify who (previously known or unknown) is interacting with the system.
3. Performing certain actions based on the interaction.

To ease the development, the focus was turned towards a more concrete implementation, a project called the "Autonomous station master". This system was to be implemented on the physical TurtleBot 4 robot, with the capabilities to help people at train stations. Implementing the proposed solution on a physical robot is very time-consuming however, and due to time constraints, the implementation was done in a simulated environment instead. Nevertheless, it was concluded that such a system should be able to act a certain way and perform certain actions, such as:

1. Greeting people interacting with the system, either by name if the person is already known or more generally if the person is unknown by the system
2. Give answers to common pre-defined questions, such as "Where are the toilets located?"
3. Perform a physical action, such as giving people a tour of the train station, in response to some interaction

To build the station master with these capabilities, four areas were identified that can be solved with relevant AI techniques. These areas, and the specific technique used, are as follows:

Decision-making: To implement a system with logic in order to both know what to do and how to act in certain scenarios, a behavior tree was used [1]. The behavior tree models the scenarios the system can find itself in and describes how to act in those scenarios.

Detecting and identifying people: In order for the system to detect who it is interacting with, the technique of facial detection was used. To then identify who it is, and if the detected face is known to the system, facial identification was implemented.

Understanding questions: For the system to analyze and understand the questions it may receive, a specific Natural Language Processing (NLP) technique called "intent recognition" was

used [2]. An intent recognition model was realized, that takes in a sentence and predicts one of several pre-defined intents, such as a greeting, a goodbye, or a question.

Performing tasks: Performing tasks such as answering questions or showing people around was done by communicating through text and driving the robot. To drive the robot, the official navigation stack for ROS 2, Nav2 was used [3]. Nav2 allows the robot to avoid obstacles and plan the path between two points.

2 Methods

This chapter presents the theory and the methods used for the different AI techniques used.

2.1 ROS 2

ROS 2 [1] (Robot Operating System 2) is a set of software libraries and tools used to build different robot applications. It provides a wide range of functionality, including support for communication between processes, low-level device control, implementation of commonly used functionality, and much more. To communicate, ROS 2 uses a publish-subscribe model and in this model, nodes (representing individual processing units) exchange messages with each other by publishing and subscribing to topics. This allows for flexible and scalable communication between different parts of a robot system.

2.2 Behavior Tree

A behavior tree is a technique used for strategically making decisions [1]. The tree is traversed from top to bottom, starting at some root node, followed by its child nodes. Each node in the tree can either be a leaf node, representing a specific action or decision, or a branch node, representing a condition or decision point. As the tree is traversed, the agent makes decisions based on the current state of the system and the available options. The technique allows the agent to make complex, realistic decisions in real-time, based on the current situation and the goals of the system. In Figure 1 a sample tree is displayed, where the colors blue, yellow, and red, display paths in the tree which represent sequence, decision, and action nodes respectively.

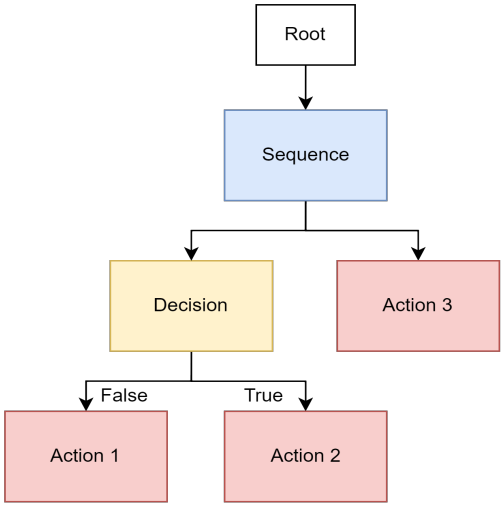

Figure 1: A sample behavior tree

[1] `https://www.ros.org`

2.3 Path Planning and Movement

Path planning is an essential task that must be done before navigating in complex environments. It is the task of calculating the shortest path between an initial position and a goal position. Furthermore, the planned path should avoid collisions with obstacles. There exist two types of path planning, global path planning, and local path planning. Global path planning can compute the shortest path from an initial position to a target position, and is based on the current position of the robot and its surroundings. The global path planning can be slow and computationally hard when many dynamic objects occur in the map, since the global path planner needs to re-calculate a new path. Local path planning is responsible for detecting dynamic obstacles using available sensors. If a dynamic obstacle would block the planned path, the local path planner adjust the velocity to avoid these obstacles [4].

Nav2 is the official navigation stack for ROS 2 [3]. In this navigation stack, A-star is used for global path planning, and for local planning, the successor of Dynamic Window Approach (DWA), DWB is used [5]. To move the TurtleBot 4, its creators have developed their own navigator package, which adds functionality such as docking and navigating to a position using the Nav2 stack.

2.4 Facial Detection and Identification

Today there are a lot of different ways of doing facial identification. One pipeline used to combine detection and identification is presented in Figure 2, it consists of 4 different stages to first detect faces, and then potentially linking that face to a known person.

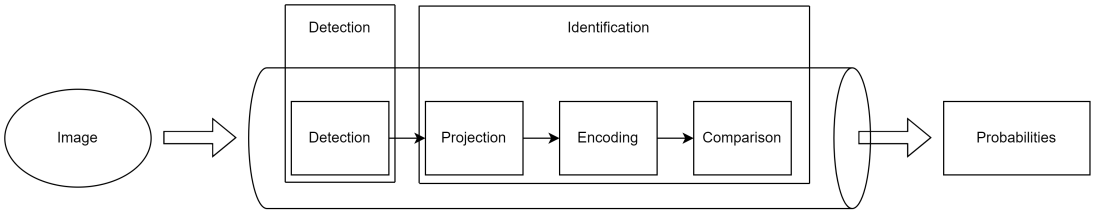

Figure 2: A simple visualization of a face recognition and identification pipeline

The first step in the pipeline is to detect potential faces in the selected image. Many types of algorithms can be used for this [6], in their paper "Blazeface: Sub-millisecond neural face detection on mobile gpus", *Bazarevsky et al.* [7] describe one way of using optimized convolutional neural networks for both detection and identifying 6 different facial key points such as eyes, ears, mouth nose, and so on, These key points are then used to detect where potential faces in the image might be located.

To then identify if the person whose face has been found is known to the system or not, one needs to have a way of comparing whether facial features are similar or not. In order to do this comparison between faces, the faces must first all be aligned in the same orientation. *Vahid Kazemi* and *Josephine Sullivan* [8] shows that this can be done by finding some facial features using an ensemble of regression trees. They do this by first warping the images to the mean

shape and then, for all splits in each regression tree, compare the intensity between two pixel pairs in order to predict an update vector for the shape parameters. The estimation introduces a form of geometric invariance which gives more certainty that a feature is detected. Once the facial features in each image are detected using *Vahid* and *Josephine's* algorithm, the next step is to apply image transformations called affine transformations, to shear, rotates, and scale the images in order for all the different facial features to be aligned. Once the faces have been aligned, the next step in the pipeline is to encode the faces into smaller representations to make it computationally easier to compare the faces. To do this, a technique described in the paper "FaceNet: A Unified Embedding for Face Recognition and Clustering" [9] can be used. It reduces the faces inside the image all the way down to a single point in a 128-dimension space. The authors *Florian et al.* describes that this is successfully done by minimizing a so-called "triplet loss" to learn how to best represent the face in this smaller space. This way, faces of different people produce radically different encoding and faces from the same person produce encoding closer to one another. Or as *Florian et al.* [9] describes the process "The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity". Lastly, to compare between the faces represented in the 128-dimension encoding space, a simple SVM-classifier can be used. This produces a confidence-value of how closely this particular face matches the known faces.

2.5 Natural Language Processing

Natural Language Processing is the task of mapping natural spoken language of humans into some internal representation that can be understood by machines. This is often done by using a machine learning model. One of the more prominent types of models is based on transformers, which transforms natural language input via several layers of encoders and then produces a response via several layers of decoders using attention [10]. One of the best performing transformer-based model today is the Bidirectional Encoder Representations from Transformers (BERT) model [11], which is a pre-trained model that can be fine-tuned for specific tasks, such as intent recognition.

2.5.1 Intent Recognition

Intent recognition is a subfield of NLP, and it is the task of classifying the intent of a sentence. As an example, take the two sentences:

1. "I am hungry"
2. "I have not eaten in days"

It is easy for humans to understand that the intent behind the two sentences is that a person is hungry. However, a computer does not understand the intent of different sentences like humans. The problem is that an intent can be specified in many ways and it can be difficult for a computer to capture all of them. There has however been some success in this area. *Hassan et al.* [2] succeeded in classifying various emotional intents with an overall accuracy of 76% in a study where they tried to detect if a person was suicidal based on conversations.

A high performing intent recognition can also be achieved by fine-tuning a BERT model for that task [12]. In a study by *Huggins et al.* [13], they create an intent recognition model using BERT to achieve an accuracy of 94%, using only 25 training examples per intent. These results are quite promising, as not only does the model achieve state-of-the-art accuracy, but it does so by training on a minimal amount of training data. This is important, since new training data is required to train a intent recognition fit for this project, and finding (or creating) enough data can sometimes be difficult.

3 Results

The results chapter explains how the different AI techniques were implemented and combined.

3.1 Simulation

The initial plan for the project was to implement all the AI techniques into a physical robot called TurtleBot 4[2]. But due to some unforeseen off-topic problems, our effort was shifted to instead work on a simulated version of the robot. For this task, the Ignition Gazebo[3] simulator was used.

Ignition Gazebo is an open-source 3D robotics simulator that supports simulation of sensors and actuators. It can create a realistic 3D environment and sensors that correctly interacts with the simulation as if it was in the real world. For this project, Ignition Gazebo was configured to spawn the TurtleBot 4 in an environment representing a simple storage space, with some crates and other obstacles lying around.

3.2 System Architecture

To structure the code as to be usable on the TurtleBot 4 both physically and in simulation, ROS 2 nodes were used. More specifically, to ensure that each part of the system could be built and tested separately, 6 different ROS 2 nodes were created that all run simultaneously and communicate via the built-in publish-subscribe model. Figure 3 shows how the different nodes interact with each other, the figure presents that the "Behavior tree"-node is central to the implementation. The other nodes, "Input", "Output", "Facial identification", "Intent recognition" and "Movement" all correspond to different parts of the system that are used by the behavior tree node to decide what actions to perform in the different scenarios.

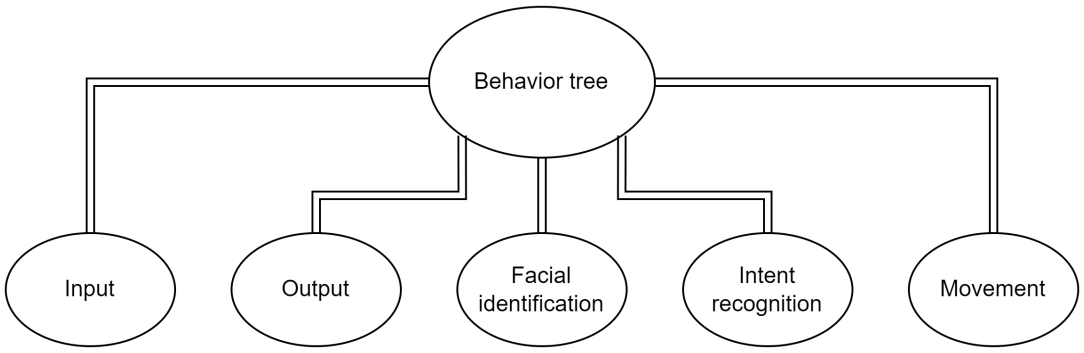

Figure 3: The ROS 2 Node structure

[2] https://www.turtlebot.com/
[3] https://www.gazebosim.org

3.3 Movement

When starting the simulation, the robot spawns in the middle of the room, undocks from the docking station, and navigates to the pre-defined starting position. When the robot is in the starting position, it is ready to answer commands from persons. The two commands resulting in movement of the robot are if the person asks for a tour or where the toilets are located. Upon receiving a request to show where the toilets are located, the robot will plan a path using the TurtleBot 4 Navigator, and then start moving towards the location. When receiving the tour command, the robot will move towards three pre-defined goals. After the robot has reached the goal location, it moves back to the initial position, awaiting new commands.

3.4 Interaction Mediums

For a person to communicate with the system, an input medium in the form of a terminal was used. The terminal allows the system to be provided with text inputs such as greetings, common questions, or requests to be shown something. Another terminal was used as an output medium to output text to a person in order to, for example, answer questions or let them know they have reached their destination. To simulate a person interacting with the system in the simulator, an external camera was used.

3.5 Facial Detection and Identification

To implement the different steps in the facial detection and identification pipeline, a set of python libraries were used. For detecting faces, Google's "MediaPipe"[4] was utilized, which provides an API for using the face-detector "BlazeFace". BlazeFace was originally created to be used on the mobile platform and therefore, it also provides very good performance on a PC. For identifying if the detected person is known by the system, another python library called "face_recognition"[5] was applied to implement all three of the remaining steps of the pipeline. The library provides an API for automatically projecting, encoding and comparing a face to a set of reference faces.

To ensure that faces both close, less than a meter, and faces further away could be detected, two different models from the "mediapipe.solutions.face_detection.FaceDetection" from the MediaPipe library was used on each image. The known faces were saved in a folder containing only known faces, with the name of the file being the name the person inputted on creation. These images and names were then imported on initialization of the facial identification node to be compared against in each frame.

To lower the strain on the system (as to be able to run on a Raspberry Pi in the physical robot), the facial detection node only has to update the behavior tree with a frequency of about 1Hz.

[4] `https://google.github.io/mediapipe/`
[5] `https://github.com/ageitgey/face_recognition`

3.6 Intent Recognition

To give the robot the ability to reply to a user or perform a certain task, the intent recognition model by *Huggins et al.* [13] was fine-tuned to classify intents required in this project. To fine-tune the model for this task, a new dataset was manually created that contained the following 6 intents:

1. Greeting
2. Goodbye
3. The person wants to find the restroom
4. The person wants a tour
5. The person wants to know the time
6. The person wants to be added to the database

In the manually created dataset there are 45 example sentences for each intent, 25 of which were used for training, 10 for validation, and the remaining 10 for testing. When training the model with this data, a similar approach to *Huggins et al.* [13] was adapted, in that different number of epochs were tested and used to find the best validation- and test accuracy. Table 1 shows the different number of epochs that were tested: [1,5,10,...,60], and the resulting validation- and test accuracies. Epochs 20, 30, 35, 50, 60 all achieves the highest validation accuracy of 0.967. These five models were then evaluated on the test data, and the results shows that the model trained trained in 20 epoch achieves the highest test accuracy of 0.969.

Number of epochs	**Validation accuracy**	**Test accuracy**
1	0.279	-
5	0.687	-
10	0.835	-
15	0.935	-
20	0.967	0.969
25	0.951	-
30	0.967	0.958
35	0.967	0.958
40	0.951	-
45	0.951	-
50	0.967	0.948
55	0.935	-
60	0.967	0.958

Table 1: Resulting validation- and test accuracy for all the training runs on the intent recognition model, using different number of epochs.

A feature was also added that allows the model to output an "unknown intent", for when the model is less confident in its prediction. The reason for this is to mitigate the model from giving the wrong replies, and instead encourage the client to reformulate the question, by answering for

example: "Sorry, I did not understand that". The confidence of the prediction is measured from the "logit"-values for each intent that is intially returned from the model. If the highest logit for all intents does not reach a certain threshold, that means that the model is not very confident in its answer. So instead of returning the intent that the logit corresponds to, the model returns the unknown intent.

3.7 Behavior Tree

To combine the different AI techniques in one unified system, a behavior tree was used. Figure 4 contains the root node as well as the individual behaviors that the agent can make, while Figure 5, Figure 6, and Figure 7 describe the behaviors in more detail.

Since the ROS 2 nodes both communicate and run asynchronously, a state system was implemented that the behavior tree node uses to execute the correct actions according to the behavior tree. Since the current tree is relatively small, the states only need to be updated by the facial identification node and the intent recognition node. As shown in Table 2, there exist 3 different states: "empty" when there is no face in the image, "unknown" when there is a face but of whom is unknown, and "name" when someone (name) has been recognized in the image.

For the intent recognition node, shown in Table 3, the possible states are "Greeting", "Goodbye", "Bathroom", "Tour", "Question" and "AddToDb". The corresponding action that the robot should perform for each intent is described in the "Action" column. To then use these different states and intents in the behavior tree, a simple "if-statement"-design is used.

State	**Description**
Empty	No face in the image
Unknown	There is a face, but the person is not known to the system
Name	When someone is identified by the system

Table 2: Facial identification states

Intent	**Action**
Greeting	Say hello
Goodbye	Say goodbye
Bathroom	Show directions to the bathroom
Tour	Give the person a tour around the building
Question	Tell the current time
AddToDb	Add the person to the database
Unknown	Ask for clarification

Table 3: Intent states

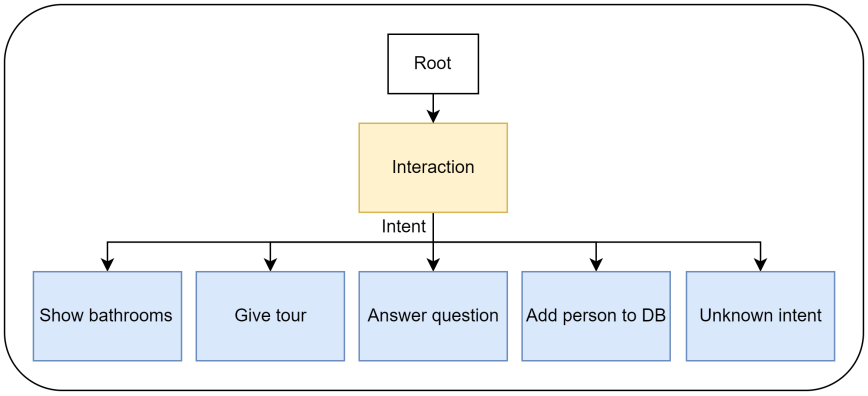

Figure 4: The start of the behavior tree

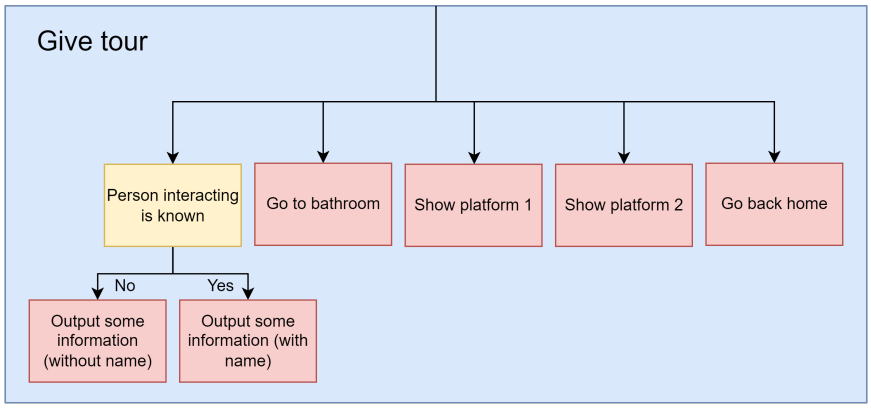

Figure 5: "Give tour" in the behavior tree

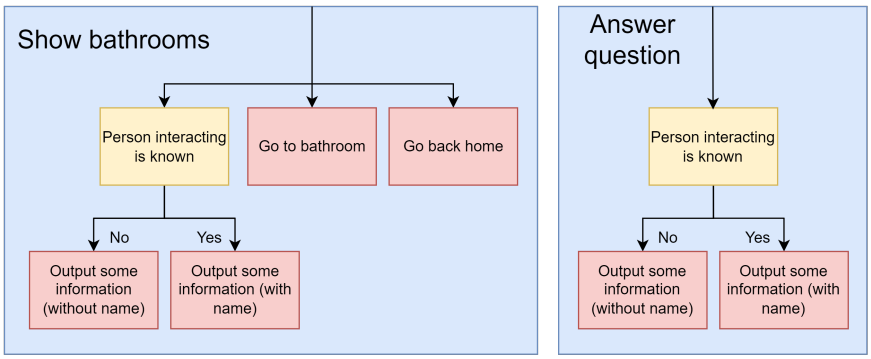

Figure 6: "Show bathrooms" and "Answer question" in the behavior tree

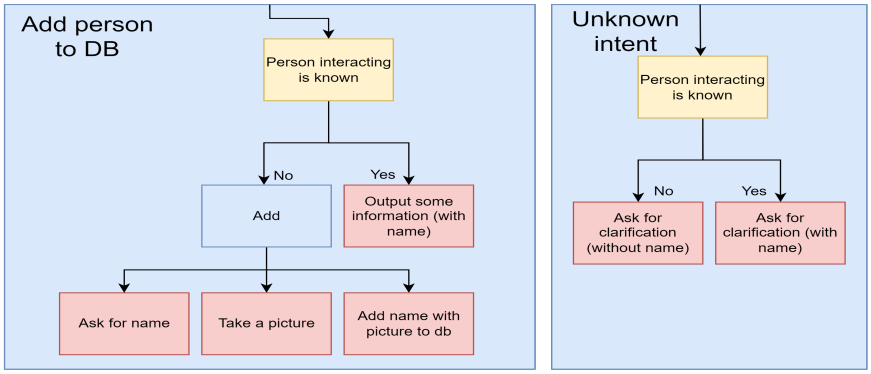

Figure 7: "Add to db" and "Unknown intent" in the behavior tree

3.8 Example Scenarios

The following are some example scenarios that can occur when a person interacts with the system. Examples of the interactions are illustrated in Figure 8.

Answers by name: As the system is able to recognize people interacting with it, answers in the output terminal can be personalized. If a person is detected and known by the system, the message in the output medium will answer the person by their name. On the contrary, if a person is not detected or a person is detected but not known by the system, the answers will not be personalized.

Leading the way: In case the person interacting with the system asks: "Where are your bathrooms?" or something similar, indicating that they want to know where the toilets are, the robot can lead the person to the toilets. During the time the robot navigates to the destination, nobody is able to communicate with the robot, such as asking common questions. Once the robot has arrived at the toilets, it outputs some text, and then returns to its initial position.

If a person instead is unfamiliar with the train station, the robot can give a tour of the place. For each of the stops on the tour, the system will output a message to the output medium letting the person know where they are. Once the tour is done, the robot returns to its initial position again.

Add person to the system: If the person interacting with the system is not known, the person can be added to the database and recognized in the future. If this is desired, the system will ask for the person's name and take a picture of the person at the end of a countdown. However, if the person is already known, the system will recognize this and not add the person.

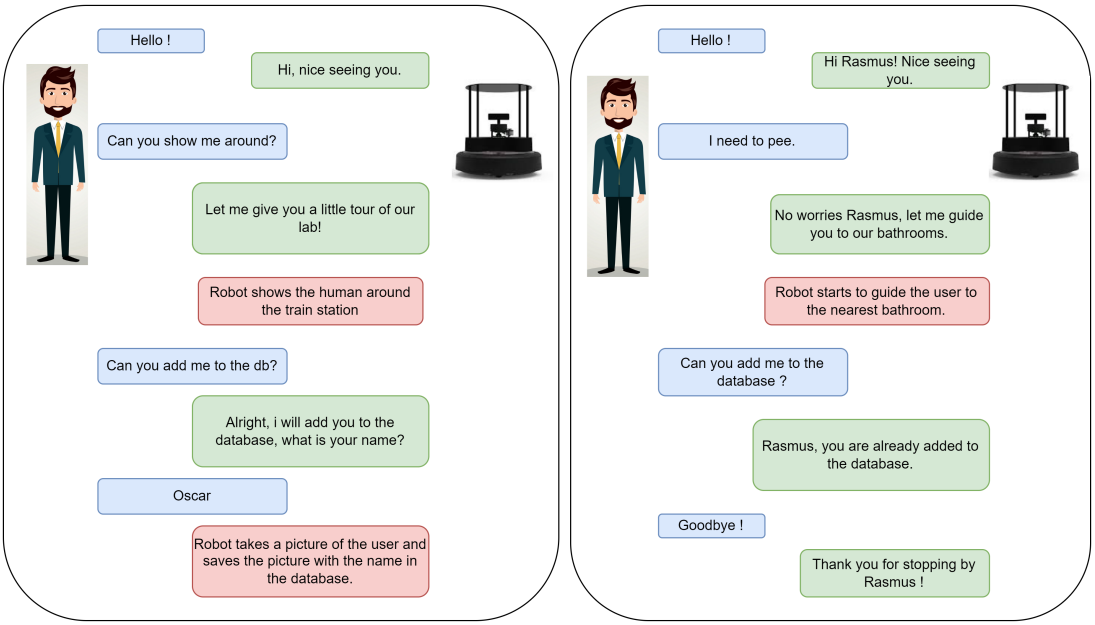

Figure 8: Interaction examples

4 Discussion

This chapter discusses the implementation, results, and potential future work.

4.1 Simulation and Robot

The final product was a simulated robot that can take commands from a textual prompt and make an informed decision in the form of actions, such as giving a guided tour, leading the way to the toilets, or just responding to common questions and greetings. Although for this project the simulated robot was a better option compared to the physical robot, it too had its problems. For example, it ran very slowly since we had to run it in a docker container. In the future it would be interesting to see how the system performs on the physical robot, TurtleBot 4, since moving away from the simulation should be relatively painless with the current implementation.

4.2 Interaction Mediums

The system currently uses two different terminals as input and output mediums to communicate with a person. The two mediums are each implemented in their own ROS 2 nodes. Although this implementation is relatively simple, it allows the possibility of changing the nodes in the future to, for example, use text-to-speech or speech-to-text as additional interaction mediums.

4.3 Behavior Tree

In this particular implementation, the behavior tree worked well. However, the tree's structure could be improved by upgrading from simple "if-statements" implementation to a more scalable solution using either a python or ROS 2 package. Although when doing this it's important to keep in mind that the system states may not always be 100% up-to-date if the current state system is still being used, as in the case of the 1Hz facial detection update frequency for example.

4.4 Facial Detection and Identification

Currently, the facial detection and identification models are only updated with a frequency of 1Hz. One reason for this was that the library used for facial identification, "face_recognition", takes an exceptionally long time in comparing a new face with known ones. Although it was noted that a higher frequency than 1Hz did improve interactions between the system and a human, it is decently an area that could be improved on in future work.

As a webcam was used as input to the facial detection pipeline, this too could be an area of improvement when working on the physical robot. We tried using a 3D camera that could be placed on the physical robot, but since it has a very distorted lens as to keep all information in one image, it did not work very well with the facial identification pipeline. In future work, some time should be dedicated to figuring out what camera one wants to use and if it requires some special facial detection models.

4.5 Intent Recognition

The intent recognition model that was created in this project achieved a validation accuracy of 0.967 and a test accuracy of 0.969. This is notably higher than the test accuracy of 0.943 achieved by the BERT model for intent recognition by *Huggins et al.* [13] for 25 training examples. Although these models cannot be compared as they are trained on different datasets, the reason for our high accuracy may be that the data used for this project is simply easier to make predictions on. For example, in their paper, *Huggins et al.* trains their models on two different datasets, one containing 7 different intents, and another that contains "several" intents. One can imagine that it is more difficult to predict correctly when there are more intents to choose from. The intent recognition model in this project only used 6 intents, and these were also quite distinct from each other, which could make them easier to distinguish. This might indicate that the accuracies achieved during testing cannot be trusted. However, the intent recognition model has performed well when implemented with the rest of the system.

Finally, the intent recognition model was trained to recognize six intents. Future work could therefore include more intents to make the robot handle more unique cases, and compare how more intents would affect the accuracy of the model.

References

[1] S. Sitanskiy, L. Sebastia, and E. Onaindia. Behaviour recognition of planning agents using behaviour trees. *Procedia Computer Science*, 176:878–887, 2020. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020.

[2] S.B. Hassan, S.B Hassan, and U. Zakia. Recognizing suicidal intent in depressed population using nlp: a pilot study. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0121–0128. IEEE, 2020.

[3] S. Macenski, F. Martín, R. White, and J. Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[4] Y. Zhuang, Y. Sun, and W. Wang. Mobile robot hybrid path planning in an obstacle-cluttered environment based on steering control and improved distance propagating. *Int. J. Innov. Comput. Inf. Control*, 8:4095–4109, 2012.

[5] S. Macenski, F. Martin, R. White, and J. Gines Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2020.

[6] A. Jadhav, S. Lone, S. Matey, T. Madamwar, and S. Jakhete. Survey on face detection algorithms. *International Journal of Innovative Science and Research Technology*, 6(2), 2021.

[7] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus, 2019.

[8] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 1867–1874, 2014.

[9] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[11] J. Devlin, MW. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Q. Chen, Z. Zhuo, and W. Wang. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*, 2019.

[13] M. Huggins, S. Alghowinem, S. Jeong, P. Colon-Hernandez, C. Breazeal, and H.W Park. Practical guidelines for intent recognition: Bert with minimal training data evaluated in real-world hri application. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pages 341–350, 2021.