Heriot watt University

School of Engineering and Physical Sciences

Multidisciplinary group project B31XP

Human Support Robot for Domestic Interaction

Authors:

Brieuc Sauzeat - H00291156

Daria de Tinguy - H00332420

Sunbul Ahmad - H00333703

Supervisor:

Dr. Mauro Dragone

April 23, 2020

Contents

Abstract

Sunbul, Daria

Consumer service robotics is a domain of home automation with an active research field and high market potential. In domestic applications, the robots are meant to help the users performing their daily routine tasks, the biggest target of this domain being the elderly people with cognitive impairments. In this report, we present an architecture implemented on the Human Support Robot (HSR) robot which can map and move around autonomously in the smart-home, recognize objects in its environment and perform a range of action according to its user's vocal commands.
Keywords: *Human Support Robot, Smart-home*

1 Project presentation

1.1 Objective

Daria,Sunbul

The goal of this project was to program a Human Support Robot (HSR) from Toyota to perform a choice of at least two of the four tasks described in the ERL competition Rulebook of 2018 [1] (those tasks will be described in section 1.2). The core idea was to prepare HSR to perform tasks in a domestic environment. Our realisation was conceived with the purpose of being modifiable by a team for an ERL competition. Therefore our objectives where:

- To understand three different modules e.g. Mapping, Speech and Object Recognition.
- To design an expandable architecture displaying an interaction between those three modules.
- To conceive at least 2 tasks in this architecture.
- To pass an exploitable code for the next team, with a complete readable.

1.2 Presentation of Home service competitions

Daria,Sunbul

At least two tournaments exist aiming to bring robots into domestic environments and support people autonomously in a home scenario. The European Robotics League (ERL) and Robocup@Home. The consumer demand for autonomous robots is rising and those leagues aim to develop the re-

search in consumer service robotics service and assistive robot technology for personal domestic applications [2].

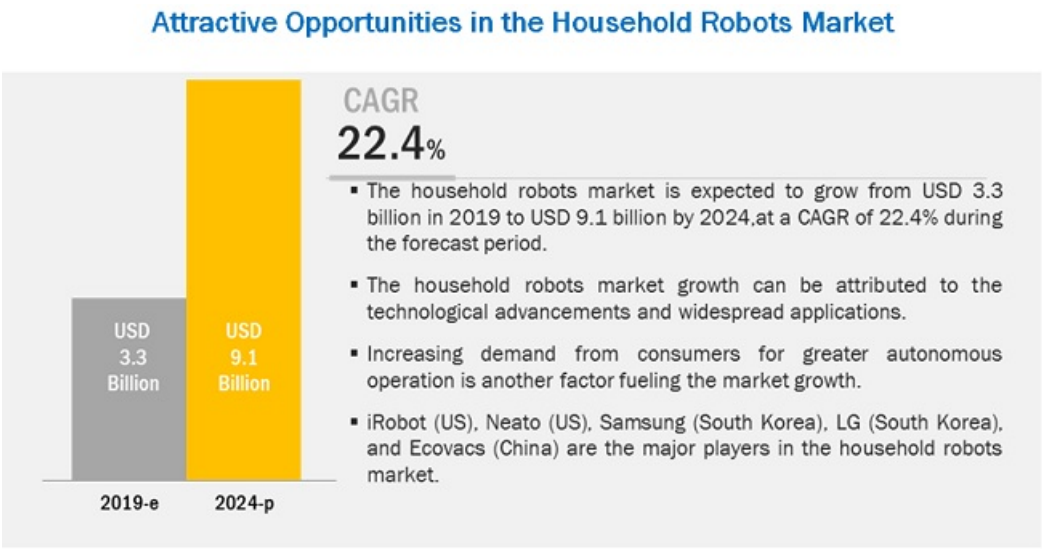


Figure 1: Attractive opportunities in the household robot market [2]

In order to work in a domestic environment a robot should be able to navigate without any assistance and know the domain, exchange and understand its owner, answer to his queries in an adapted way, and above all, be safe to use. ERL and Robocup@Home are open to any group with commercial or home made robots (in sub-platforms however) and the challenges fall into 2 categories: Basic functionalities achievement and complex tasks realisation. The basic functionalities of the robot are evaluated, as:

- Speech understanding,
- Recognizing,
- Navigating,
- Finding,
- Manipulating,
- Tracking,
- Following (a person)

The second category integrating those functionalities into more complex tasks in order to fill a human request (as, for instance, getting someone's glasses back to its owner).

The ERL competition and Robocup give 4 to 5 tasks to complete, with a set of exigency for each. The best a task is accomplished, the highest the grade. The team with most point after the execution of those tasks is rewarded as the winner. In 2019, the ERL competition had the following tasks [1]:

- **Getting to know my home**: Robot detects change in its environment.
- **Welcoming visitors**: Recognize known/ Identify unknown visitor and act with them accordingly to their identity.
- **Catering for Granny Annie's comfort**: Understanding a vocal command and be able to identify and find an object in its environment, manipulate an object, accompany a human from one point to a goal.
- **Visiting my home**: Recognize a mobile objective and track it, deal with obstacles depending on their nature (humans, objects, dynamic new obstacles). Follow an unknown individual from a point A to B then guide him back to A.

The software demonstration of the robots accomplishing those tasks is a source of improvement for research and application perspective in the field of domestic robots.

1.3 HSR Description

Daria, Sunbul

The Human Support Robot (HSR) is developed by TOYOTA. This robot is meant to assist people, and notably disabled and elderly, in everyday life, therefore, it's a perfect robot to use for a customer service competition. HSR can move around the house, keep watch over family members, and fetch objects. Its creators hope to make HSR beneficial to all people in the near future [3]. The HSR is able to self-locale, detect and avoid obstacles and plan motion. It can pick up objects or papers at different heights and has 4 cameras, 3 sensors. Those camera and sensors are [4]:

- An RGB-D sensor with a depth image and color image. It can be used to recognize objects and place them in the environment.
- A wide-angle camera with a wide-angle view image of 135° horizontally.
- A stereo camera to determine the distance information of an object from the parallax of the right and left cameras.
- An IMU sensor to recuperate the acceleration and angular velocity of HSR.

- A force-Torque sensor, to obtain the force and torque acting on the wrist.
- A laser range sensor to gather information about the environment with a horizontal scan possessing a 250°angle.

In HSR, the robot can speak by speech synthesis. The synthesized voice do not exist in the HSR simulation but it's functioning can be confirmed by text instead. Finally, the speech recognition of HSR incorporates Julius which has a dictionary composed 70% of Japanese words and 30% of English words.

2 Related work

2.1 Introduction

Daria,Sunbul

In this section we intend to briefly present our researches in order to help the next HSR team in its development. We don't intend to fully detail everything but to inform about packages existence and use.

In order to accomplish any tasks some elements are necessary. First of all a domestic robot needs to exchange with a human. It needs to understand and answer to a query. Then it needs to be able to navigate in an environment and know where he is situated or where he has to go. Finally being able to understand its environment, recognize the objects and remember their position is essential to know its home.

Therefore we will briefly present different techniques (used in domestic competitions) to map and navigate around the home, Secondly we will develop how speech recognition and speech synthesis could be developed in robotics to take vocal commands from the user. Finally, we will present how object recognition and object mapping could be achieved. Those elements are the minimum required for a robot to be of use in a home environment. However, even though face recognition and object manipulation won't be extensively presented here, they are also notable elements that would enrich the robot comportment. Once all those elements defined, it is important to know which package to use and how to structure each functionality into a complex architecture. Hence we will expose the composition of some competition winning teams architecture before presenting our own work.

2.2 Mapping and Navigation

Daria

In this section we will briefly describe some solution to map and navigate in an environment. All the solutions described here are ROS supported.

2.2.1 Localization and Navigation

In this section we only present the most overused package, SLAM and its diverse versions. The point is not to describe each of them but to inform the next team of their existence as potential tools to improve navigation.

SLAM

Simultaneous Localization and Mapping (SLAM) creates a local map of the environment and determines the robot position while moving. Current measurements and localization are changing at each displacement, therefore, in order to create a map it merges measurements from previous positions. A lot of improved versions of SLAM exist as the Iterative closest point, Probabilistic SLAM, ORB-SLAM, AMCL etc. that we won't describe here.

However it is interesting to note that a localization can be improved with the use of EKF (Extended Kalman filter) or UKF (Unscented kalman Filter), some of those versions integrate them already in their solution, others do not. The kalman filters are used to fuse data coming from diverse sensory inputs (as laser, camera, gps etc.) with the probability of their correctness and give a better estimation of the robot pose [5]. Most teams presenting their navigation system use SLAM, but most of them do not detail which particular SLAM version they use [6, 7, 8, 9, 10, 11, 12, 13].

2.2.2 Mapping

This section, as the previous one, do not intend to fully describe each solution but to offer a mean of comparison, and inform of diverse mapping alternative as a starting point for further researches.

Gmapping

Gmapping is SLAM based. It extracts data from laser (Lidar data) and odometry (motor encoder count for example) to conceive the map. The SLAM estimates two things: the map and the robot's pose within this map. Gmapping use this data and combines it with a Particle Filter (PF) to estimate the real robot's position combined with the information coming from the Lidar and the odometry of the robot. Once the map has been estimated it will be set as it is in a 2D layer [6, 7, 11, 5]. It is important to note that to use Gmapping safely you need an impeccable odometry else the map will drift (see figure 2 and (A) Figure 18).

Octomap

Octomap is based on Octree (tree data structure, often used to partition a three-dimensional space) and can create a 3D occupancy map. It uses data from the lasers (2D or 3D), the cameras, the odom-

etry and fuse all this data with a probabilistic sensor fusion to obtain the map. The environment is defined as a 3D grid with free or occupied cells. To determine the robot pose, they use a Monte Carlo localization (AMCL) based on the 2D laser range measurements. Octomap is well used to create semantic maps as it can recreate 3D objects in the map [6, 8, 14]. However, even though it is optimised, it requires much computational power.

Cartographer

Is an alternative to both Gmapping and Octomap as it uses SLAM to localize itself and create 2D or 3D maps according to the fused data sensor it can collects [10].

However Cartographer is hard to tune as it has many parameters.

Hector mapping

Then hector mapping also generates 2D maps using SLAM, but it doesn't base itself on odometry but on external sensory information as Laser and cameras. It is important to set the frame and parameters right for hector mapping to give an optimal solution (see (B) figure 18). Furthermore, in a closed environment where there is a lot of sensory data coming in, Hector mapping is interesting, however in an empty open spaces, Hector mapping won't be adequate as there will not be enough sensory inputs to localize itself in the map since odometry isn't considered [15].

Rtab mapping

Lastly Real-Time Appearance-Based Mapping (RTAB) is based on SLAM and sensory inputs from the lidars, cameras (stereo and RGB-D) to recreate a graph. it uses bag of words and need many loops closure to create a 2D or 3D map. This mapping technique is well used for semantic mapping. Rtab can squish 3D maps into 2D (e.g. a table will be a black square on the 2D map, indicating it's presence) (see figure 3 and (C) figure 18). It can be easily used and setup, however extracting the map may prove to be a bit challenging and slow [16].

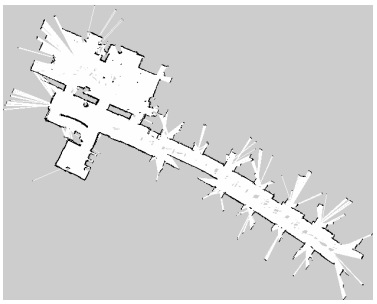


Figure 2: Gmapping map

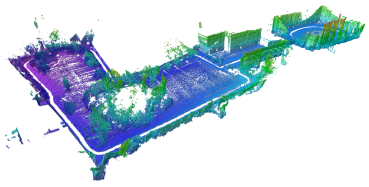


Figure 3:
Octomap map

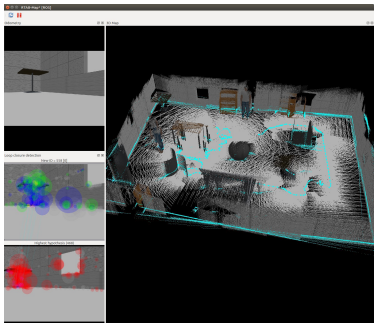


Figure 4: Rtabmap map

2.3 Speech Recognition

Sunbul

In this section, we concentrate on the speech recognition tasks that are human-robot interactive. So, this is how Speech recognition works, You speak into a microphone, either your computer's or the robot's and the computer/robot's system transforms that sound of your words into string of words that can be used and processed in any way. Then, the computer may repeat what you just said or it may give you a prompt for what you are expected to say next. This is the central promise of interactive speech recognition. Early speech recognition programs made you speak in staccato fashion, insisting that you leave a gap between every two words. You also had to correct any errors virtually as soon as they happened, which means that you had to concentrate so hard on the software that you often forgot what you were trying to say. There is not many robots in the competitions talking back to the human. Mostly the software used in different teams scenarios is Festival [17].

Some of the systems also look at whole phrases, not just the individual words you speak. They try to get information from the context of your speech, to help work out the correct interpretation. This is how they can (sometimes) work out what you mean when there are several words that sound similar (such as "to," "too" and "two".)[18].

2.3.1 Text-to-Speech, Natural Language Processing

A Text to Speech Synthesiser (TTS) software called eSpeak is used for converting the details of the detected object (in text format) to speech output. eSpeak is a compact, open source, software speech synthesizer for Linux, Windows, and other platforms [19]. It can provide many languages since it use formant synthesis method. It supports Speech Synthesis Markup Language (SSML). eSpeak helps in changing the characteristics features of voice such as pitch range, frequency and add effects such as echo, whisper etc [20].

Pyttsx [21] and eSpeak [22] collectively construct the speech processing. Ptttsx is a cross-platform Python library for text-to-speech synthesis that completely relies on the default speech synthesizer depending on the Operating Systems. For Ubuntu, it uses **eSpeak**, a compact open source software speech synthesiser [19] written in C which supports SSML (a markup language for different speech synthesis applications) and HTML [23].

In our case, Text to Speech Synthesiser (TTS) software eSpeak was used for converting the details of the detected object (in text format) to speech output. It also helps in changing the characteristics features of voice such as pitch range, frequency and add effects such as echo, whisper etc [20].

The strategy is that it first stores the text to pronounce in a SSML file (Speech Synthesis Markup Language), which allows specifying the speech rate, the voice to be used by the synthesizer, pauses,

etc. After that, it scans the SSML looking for the required keywords related to our context. The SSML file is read, and its motion sequence is computed, assigning commands according to the associated keywords. Content and non-content words are also distinguished here [23]

Code:

```
import os
os.system("espeak 'Hello Human, How are you today?'")
```

2.3.2 Speech-to-Text, Natural Language Processing

2.3.3 Google API

In order to synthesize speech into the text(STT), we used Google API [24]. Google has a great Speech Recognition API. This API converts spoken text (microphone) into written text (Python strings), briefly Speech to Text. You can simply speak in a microphone and Google API will translate this into written text. The API has excellent results for English language. This program will record audio from your microphone, send it to the speech API and return a Python string [25].

2.3.4 Text Analysis Techniques

In recent years, with the advent of Big Data and the immense amount of textual data coming from the Internet, a lot of text analysis techniques have been developed by necessity. In fact, this form of data can be very difficult to analyze, but at the same time represents a source of a lot of useful information, given also the enormous availability of data. Just think of all the literature produced, the numerous posts published on the Internet, for example. Comments on social networks and chats can also be a great source of data, especially to understand the degree of approval or disapproval of a particular topic [26].

Analyzing these texts has therefore become a source of enormous interest, and there are many techniques that have been introduced for this purpose, creating a real discipline in itself [26]. Some of the more important techniques are the following:

1 - Analysis of the frequency distribution of words

2 - Pattern recognition

3 - Tagging

4 - Analysis of links and associations

5 - Sentiment analysis

2.3.5 Natural Language ToolKit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. `https://www.nltk.org/`

2.3.6 Information Extraction

A sentence can contain information in any shape and size. It is important that we get our desired information in the right manner. For that, we tagged and tokenized the sentences using NLTK. Figure explains the simple architecture for the extraction of information.

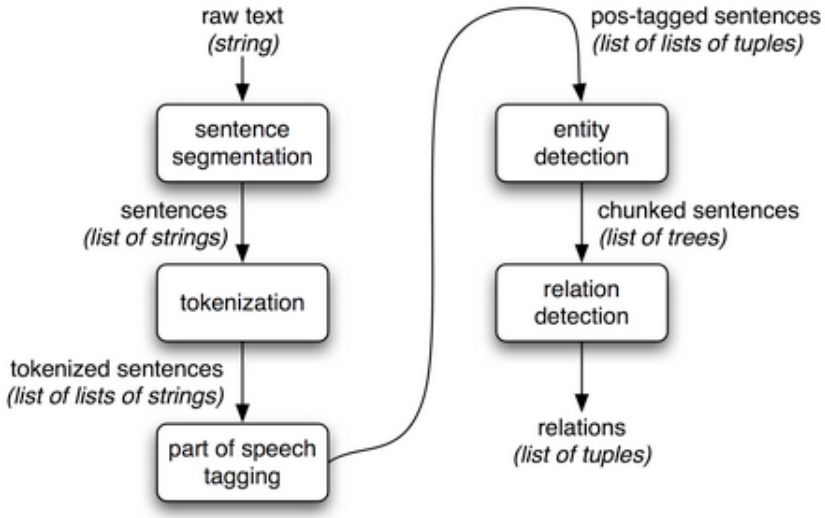


Figure 5: Architecture for an Information Extraction System

The raw text of the document is split into sentences using a sentence segmenter then each sentence is further subdivided into every single word using a tokenizer. After that, each sentence is tagged with parts of speech tags (Nouns, Pronouns, Verbs, Adjectives) , which will prove very helpful in the next step, named entity detection. In this step, we search for mentions of potentially interesting entities in each sentence. Finally, we use relation detection to search for likely relations between different entities in the text `https://www.nltk.org/book/ch07.html`. Figure shows a simple example of information extraction.

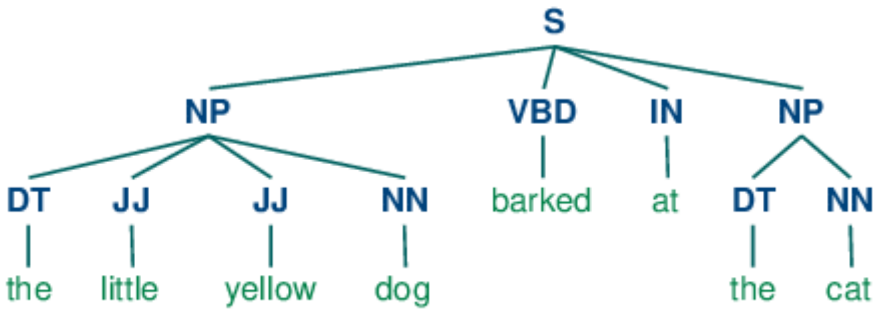


Figure 6: Information Extraction

2.3.7 Dialogue Management

To manage the The Dialogue Flow of the system, we created a simple data base with nearly all possible objects, places and tasks in the Smart-Home scenario. The end of a sentence is determined through silence. To get the intent out of the sentence is done by the Word Tagging (explained above). After analysing different sentences we realized that most of the time, The Verb used in the sentence shows the intent of the user. For example if user says, **"Please bring me the Milk"** it tokenize it in such a way that **Bring** is verb and also it is the intention of the user get the milk. It is true for nearly all the possible interactions in the smart-home context. The figure shows a simple flow chart with the Ros Topics and speech sub-system used in the project.

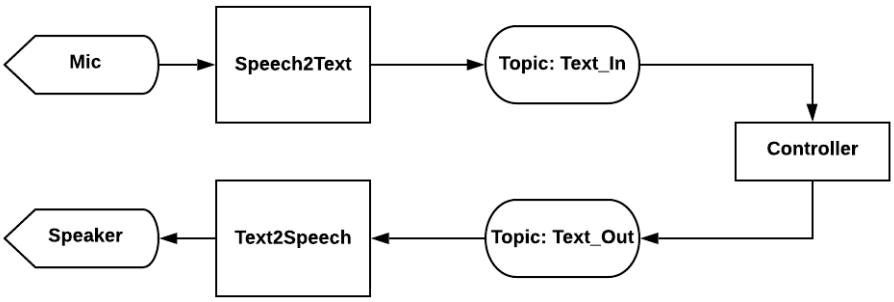


Figure 7: Speech Sub System

In the current implementation voice clips recordings are sent after every 2-5 seconds. We implemented **"STOP"** and **"DANGER"** commands in case the user gives the wrong commands by mistake or if the user wants the robot to do something urgent and stop the tasks it is currently doing. STOP will make the HSR to stop the action its currently doing and DANGER will kill all the nodes and will make HSR to stop everything. To run the file we first created a **Thread** which will keep on running at the back and will keep on listening every 5 seconds. Whenever user say

Hello it shall start interacting but due to the thread conflicts with the main architecture, we had to remove it.

Human-Robot Interaction - When robot starts interacting with the human, it first tries to be friendly and ask general question, e.g. how are you?, how is your day going? etc. Here we check the **"Similarity Index"**. So, it works by checking the probabilistic similarities between the user's answers and the data set of answers we have created in our program. It take union between the two and returns the similarity. If similarity is greater than 0.5 it responds to proceed the interaction otherwise, it asks user to please say it again because he could not understand. In next step, it asks the user about the command. When user gives the command, e.g. Please bring the pillow from the bedroom. It first confirms that the robot understood the correct sentence. Then it again checks the similarity but this time not only with the user's sentence but also with all the possible synonyms in that sentence, to avoid mistakes and then it executes the action.

2.4 Object Recognition

Brieuc

As Stated Previously in order to do the Object Recognition we will use the Package YOLO which has been given by the Toyota HSR Team.

2.4.1 What is YOLO

YOLO is an object detection and recognition algorithms, which is part of the Darknet project.

This Tool, once it has detected an object will surround it with a bounding box and will attached a label, which is the name of the detected object. This make YOLO easy to works with, furthermore as it is an open source project it possess many pre-trained weight that can be instantly use.

2.4.2 How does it works

YOLO will apply a grid of S x S to the image. Once it is done, it will look if an object is in a cell and will create a bounding box accordingly with the cell and attribute a confidence factor to each one. This confidence factor contain the accuracy that there is an object in the cell and how much the bounding box is close to the real object. it will then take the bounding box that is the most accurate. [27]

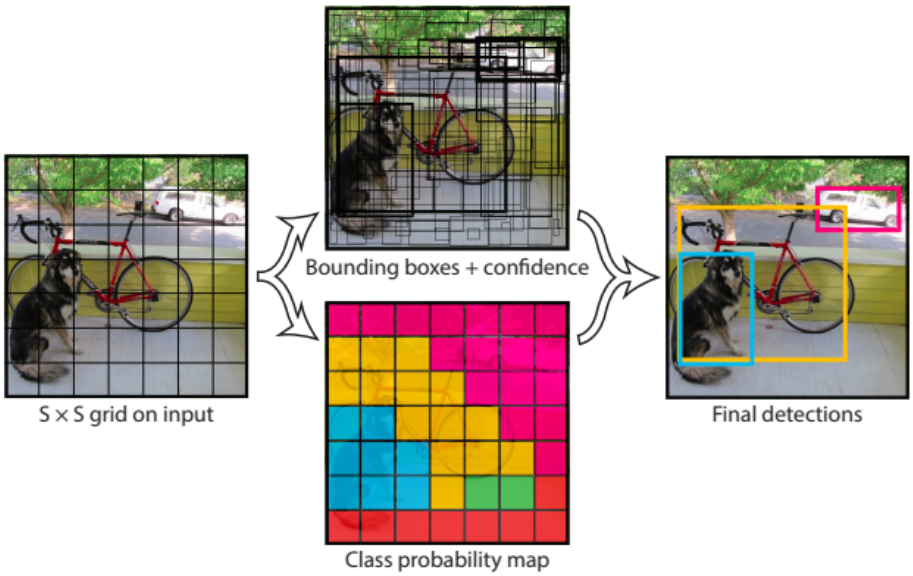


Figure 8: Explanation of YOLO

2.4.3 How to use YOLO and csv file

Our program that create the semantic map alongside YOLO works in 3 stage:

First stage the Robot will take a picture with one of his camera and YOLO will analyse all the object on this picture, Then the robot will use the 3d-position [28] to obtain the real coordinate of the object in the map, and at the and a self-made program will take the object and their position and will compare them to the object previously detected on the file, if they have been already seen by the robot, the program will only update their location, but if they are new it will add them to the semantic file.

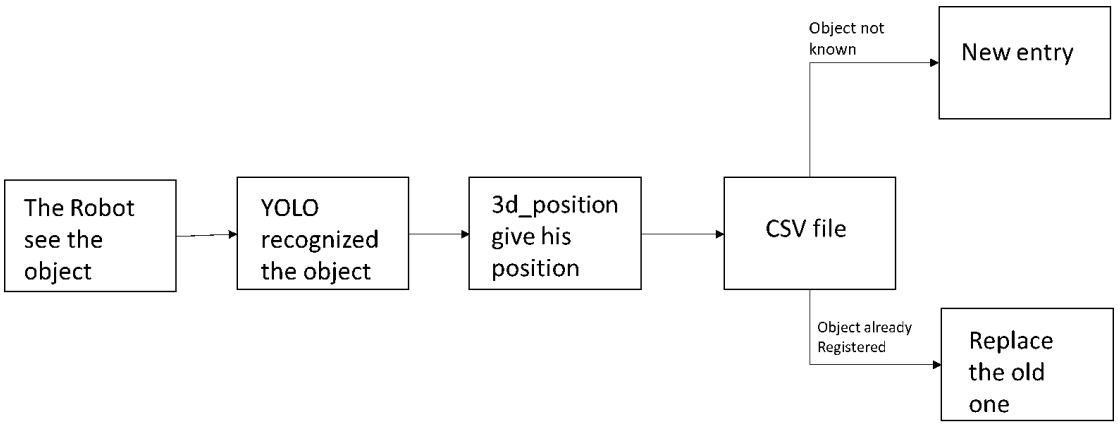


Figure 9: Schema of the entry in the csv file

The flaw of this method is that if an object is moved to far from its original position then the robot will think it is a new object and will 'duplicate' it in the semantic map. Once the semantic map is done it is very easy to manipulate as the program will transform it into a matrix we can freely manipulate.

2.4.4 How to train YOLO

The YOLO package given by HSR come with a tutorial on how to train our own weight for detection. [29] In order to do that we first choose if we want to train with the real robot or in simulation. In our case due to technical problem with the robot we choose to train them in simulation. In order to train our own weight, we need to take 50 pictures at different angle of each the object we want to detect, and put them in their own folder. For that we take a screenshot with the tool given to us in Gazebo. Once we have down that we need to specify where the object is on each picture by using the Bbox-label-tool given by HSR.

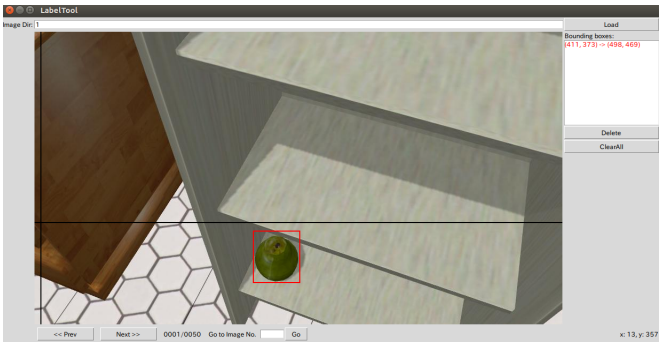


Figure 10: Bbox-label-tool

Once we have labelled the 50 images of each object, we will use a script that will process each image in order in order to artificially increase the number of samples. After the processing another script will format the samples for the training. Once this is done, we can begin the training.

Once the training is complete we have to assign a name to each object to tell what they are.

Unfortunately, we had problem with a tool needed for speeding up the training which make our weight unusable.

2.5 Teams Architectures

Daria/Sunbul/Brieuc

In this Part we will present the different architectures of the teams having previously participated, with high results, in a robot customer service competition. Those architectures were then

used as references for our own realisation.

A team architecture is a combination of functionalities organized in different actions. This architecture is what allows the robot to work and display a behaviour. The architecture, composed of diverse action is usually conceived with a state machine. A state machine is what could be called the ”brain” of the robot, it is the program skeleton, that will call the different functionalities in order to accomplish the desired goal.

Table 1 present the tools chosen by each team in order to compose each functionality (as exposed on figure 11). We can observe that each team made different choices, and they all produced a coherent architecture based on those bricks. This is mean to show that the package is not as important as the use we make of it and their integration in the architecture. However we can observe that when it come to manipulation, MoveIt! is a recurrently used tool showing good results.

Teams	**Robot**	**Speech recognition**	**Navigation**	**Face/Object recognition**	**Manipulation**
Homer [30] Sunbul	Lisa	Nuance (VoCon software)	Planar estimation	Point Cloud Library Hand-made classification Network	MoveIt! with an RRT-connect
Catie [10] Daria	Tiago	Snips, an offline, free tool	SLAM, gmapping	YOLO	MoveIt! Octomap (3D occupancygrid)
UTMIL [31] Sunbul	HSR	Google Cloud Speech API	Non specified navigation stack (ROS)	TensorFlow Keras	MoveIt!
Sspl-Lyontech [32] Brieuc	Pepper	Pepper User interaction	Pepper navigation	YOLO	OpenPose

Table 1: Teams choice of packages for each task.

Figure 11 is an example of the full architecture of a winning team in the ERL and Robotcup@home competitions (during several years in a row) presented in actions and associated functionalities. This is the core of the architecture, how we compose each package (navigation package, speech recognition package etc.) into a coherent structure.

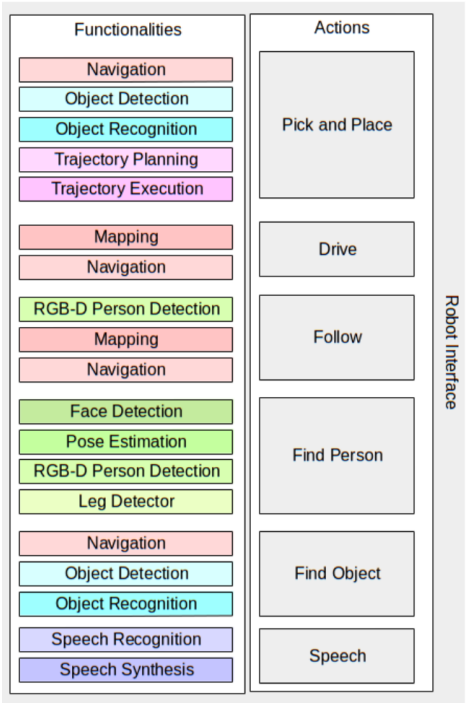


Figure 11: The Homer team used a 6 actions state machine with sub-actions (each color correspond to one category of functionnality)

We can observe that navigation is the most recurrent sub-actions in this architecture. If the robot can't understand its position and move accordingly, nothing else can be achieved.

The ROS packages/nodes used in UTMIL@Home 2018 as mentioned in Figure 12 are explained below. To Perform each core component of Robocup@home they used:

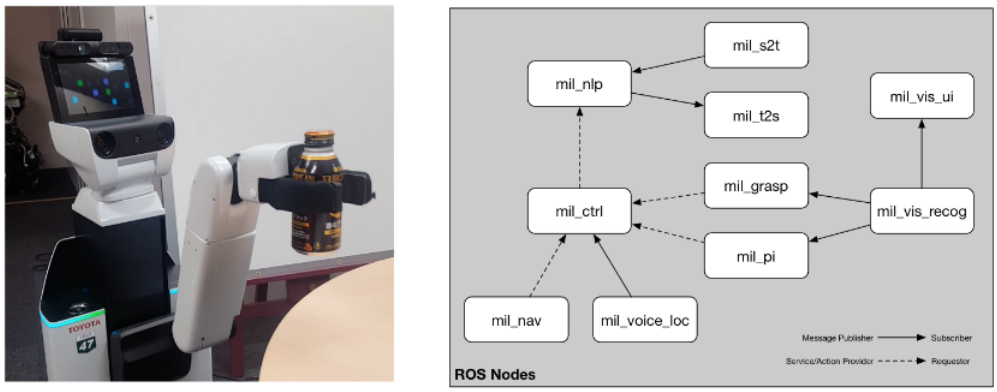


Figure 12: ROS packages/nodes in UTMIL@Home 2018 [31]

- **mils2tSpeech** to text module. HSR's internal Python interface is adopted [31].
- **milt2sText** to speech module. Google Cloud Speech API [31].

- **milnlp** Natural language processing module. It consists of various processing methods [31].
- **milctrlTask** control module. This is a group of nodes each of which is responsible for a high level task suchas “fetch me the black bottle” and “introduce me to the others” [31].
- **milnav** Navigation module. It is responsible for map construction and loco-motion of the robot [31].
- **milvoiceloc** Voice localization module. In some tasks, HSR responds to sounds(E.g., rotate to face the person giving the command) [31].
- **milgrasp** Manipulation module. We exploit the Python interface provided by Toyota to move around the end-effector and grasp/release objects once we have detected the objects’ poses [31].
- **milpi** Person identification module. Our HSR robot can perform person identification based on its vision [31].
- **milvisrecog** Visual recognition module. Rather than making a node for eachtask involving visual recognition (E.g., YOLO for object detection, FasterR-CNN for face detection, etc) [31].
- **milvisuiVisual** recognition UI module. This is simply a visualization node for debugging and behavior explanation [31].

We could observe that some applications (packages) were coming back more often than other to achieve a functionality in the competition. Based on thirteen Teams papers and papers reviews, we present here the general tendencies on the tools choice for each functionality. This layout is based on the teams describing their solution for each theme [30, 11, 32, 6, 33, 7, 8, 9, 34, 10, 12, 13, 4]:

- ROS is the middleware used by all the teams fully describing their architecture.
- **Mapping:** 42% of the participants used Gmapping, the second most used solution being Octomap.
- **Localization/Navigation:** 72% of the teams used SLAM (Iterative closest point/ Probabilistic SLAM/ ORB-SLAM/ AMCL), the use of an EKF or UKF increase the SLAM robustness. Then, on the margin, some toolkit can be used as the Mobile Robot Programming Toolkit or the Advanced Robotic Navigation and Localization System.

- **Manipulation:** 66% of the teams used MoveIt!, then rapidly exploring random tree (RRT), probabilistic rodmap (PRM) or Open Motion Planning Library (OPML) have been marginally used. None of those solutions will be further explained in this paper.
- **Object recognition:** Most of the Team were using YOLO (You Only Look Once), which is a package for object recognition that use deep learning in order to recognize the object.
- **Face recognition:** No particular package seems to be used for face recognition (when it was being realized). When a team used Face recognition it was mostly a self made program made with deep learning.
- **Human recognition:** Most of the Team used their object recognition program or face detection program for human recognition as YOLO can detect people.
- **Speech recognition:**

  The softwares used to do the speech recognition (English) by different teams were:

- Microsoft SDK,Vocon,Loquendo.
- ROS available: CMU sphinx, Android Speech Synthesis and Recognition (API).
- Google speech API.
- IBMBluemix Speech Recognition Service (Watson).
- Chatito (training data for Rasa/Snips).
- SNIPS.

3 Our Architecture

Daria,Sunbul,Brieuc

For our architecture we decided to use Google API and eSpeak for, respectively, speech recognition and synthesis. Hector mapping to map the environment and the HSR functionalities to assure a good navigation. Finally YOLO was chosen for object detection and adapted to our use. We work on ROS on Ubuntu kinetic and melodic, so all our work is compatible with both.

Google Speech-to-Text API enables the users to convert audio to text with its powerful neural network models and its not very complicated to use. The API can recognize more than 120 languages and it is very helpful with recognizing different accents.

Hector mapping was chosen for its good map accuracy and usage convenience. The navigation is based on our generated map and HSR functionalities, HSR functionnalities include object detection and trajectory recalculation.

YOLO was the package used by most of the team which had participated in a robotic competition, it is easy to use, very precise and possess pre-trained weight for object recognition. But one of the main reason is the fact that the HSR is equip with a Jetson and pre-made package to used YOLO with the HSR.

Robot	**Speech recognition**	**Speech synthesis**	**Navigation**	**object detection**
HSR Toyota	Google API	eSpeak	Hector mapping HSR navigation	YOLO

Table 2: our choice of package for each task.

Those packages take HSR data as inputs and deliver (after being filtered by our architecture) exploitable outputs as seen in figure 13.

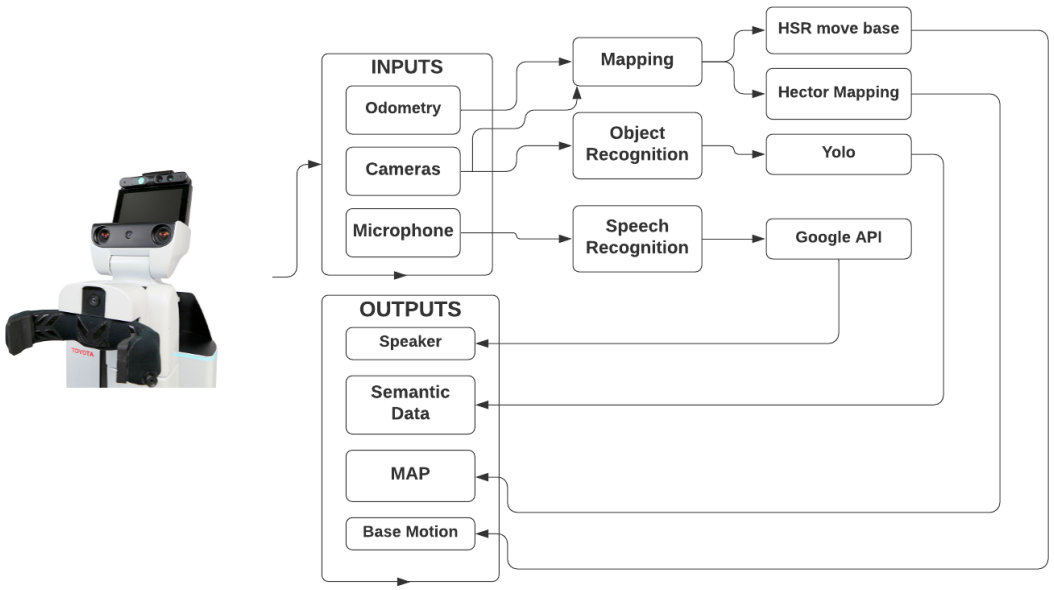


Figure 13: Inputs/Outputs map of our main functionalities

Figure 13 shows the basic interactions between the inputs and outputs of our system. It shows how they are interacting with each other.

Inspired by Homer architecture [10] we created actions and functionalities as represented in Figure 14 that are bricks of our state machine architecture.

Our architecture is composed of 3 actions : ”Mapping”, ”Get” and ”Welcome” each chosen by

vocal command.[1]

Extraction\vocal command	e.g. "Bring me a banana"	e.g. "Go check the door"
Action	Get	Welcome
Object	Banana	X
Where from	Home	X

Table 3: Instance of how a vocal command is transformed in instances (action, object and where from)

- **Mapping** does a mapping of the home (the doors must be open), object recognition and vocal command are on (to abort the mapping or even stop hsr completely). Once the process finished (success or not) the map is saved.
- **Get** searches in its semantic map if he knows this object and where it is. If yes, then it searches a free space close to it and move. If no, it searches the room concerned by the command and if the object is found the process end and the object position is returned, the closest accessible point found and the robot move.
- **Welcome** works when the user asks the robot to go and check the door. Through our Speech Recognition and Synthesis we enable the robot to welcome the visitor. The robot will recognized the uniform of the visitor and he will greet him and guide him if he is recognized.

[1]Our architecture can be found on Github: https:https://github.com/sma20/Human-Support-Robot-for-Domestic-Interaction

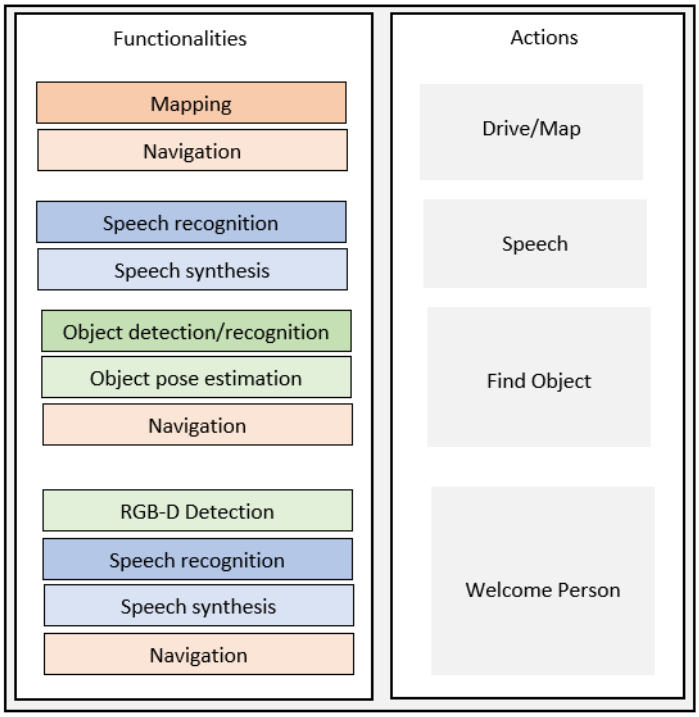


Figure 14: Our architecture in actions and functionalities blocks

3.1 The state machine

Daria

The state machine was created using SMACH [2] . SMACH is a ROS package allowing the construction and visualization of relatively complex state machines. At first we thought of using Flexbe instead, as it possesses a visual interface to construct our states, but it was not a trustworthy tool and would regularly crash. The state machine waits for an action to be given to it by the speech recognition topic, once one is recognized it goes to the sub state machine corresponding and execute each step of the process. In case the user got the wrong command or wants to give a new one a **"STOP"** command can be issued to stop the action running and go back to waiting a new order. In case we want to stop the robot in an urgent situation, the **"DANGER"** command can be issued any time and would kill all the processes.

[2]SMACH : http://wiki.ros.org/smach

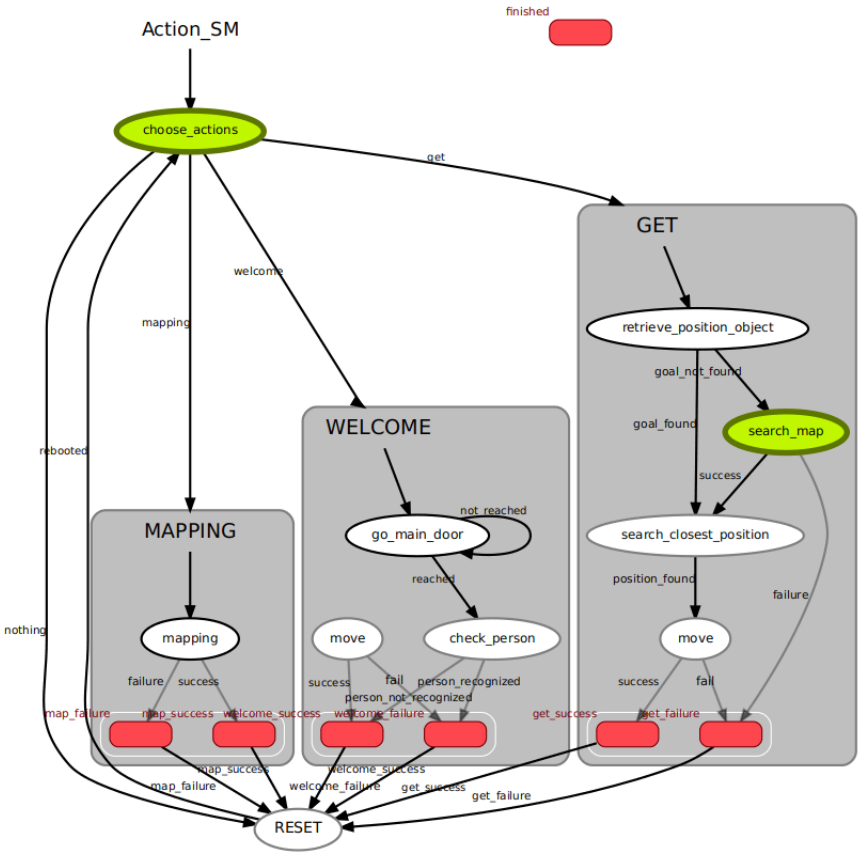


Figure 15: Our State machine and related actions on SMACH viewer

Some of the actions (as mapping or searching the map) can be quite long or get stuck in infinite loops, in those long actions a timer has been added to stop them if they run for too long.
We will here describe each and every brick of the architecture.
Let's begin with the mapping action.

3.2 Autonomous mapping

Daria

The autonomous mapping was based on an A* strategy re-adapted to our use [3].

The A* algorithm searches the less costly path from a point Start to a point Goal. It plans the most optimal path in a known map [35].

[3] Turtlebot Navigation package: https://github.com/bnurbekov/Turtlebot_Navigation

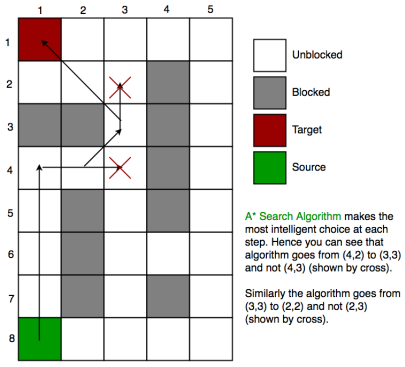


Figure 16: A* Search algorithm path planning example [35]

In our code the algorithm finds the closest unknown area and search the best path to reach a known area around this position. For our use, we just extracted the goal position of the A* search and sent it to the HSR move action, which allows us to detect obstacles as the HSR moves toward the goal. The mapping process is long because the map is re-scaled each time we search for the new closest accessible unknown area. Any map created with this mapping will be of format 2048x2048. The auto-mapping strategy follows the step of the flow chart below:

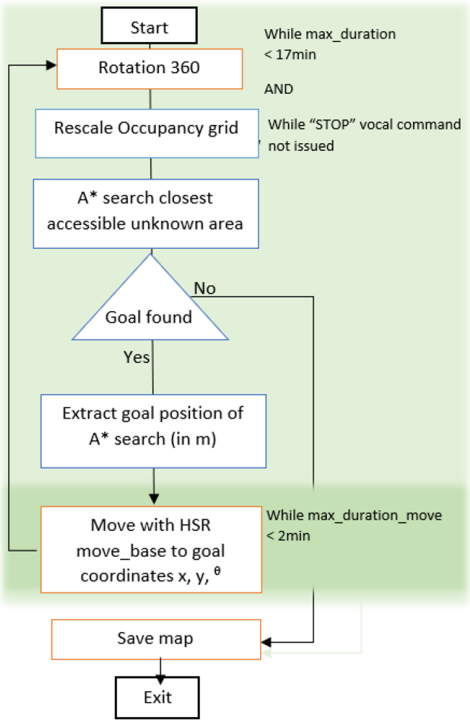


Figure 17: Flow chart of the mapping process (simplified). Orange box means the action take in control.py and Blue means it's from mapping.py

Hector mapping is used to map the house. It uses and fuse data from the cameras, the laserscan and the transformed odometry to estimate its position. This gives a good estimation of the robot position if there is a lot of landmarks to rely on, as in a home. Gmapping relying solely on odometry gave results a lot less precise (see A and B figure 18). Rtabmap was also a considered option to map, as it also relies on cameras feedback and has the advantage of converting 3D information into 2D (see estimation C in figure 18). However the map created with Rtabmap didn't save itself where intended and we didn't succeed in recuperating the 2D map. Having a map with obstacles printed on it would allow us to consider the obstacles presence more easily.

Figure 18: The results of the mapping using (A) Gmapping, (B) Hector mapping and (C) Rtabmap (estimation)

To navigate, the HSR possesses a topic and an action mode to which we give the point to reach. Both mode have object detection comprised as functionality, but the topic does not give feedback on the completion of the movement while the action does. In this package the action is being used to be able to control the action duration. Some areas are still grey on (B) figure 18 because the robot can't reach those areas. In those cases, it's highly likely that the function won't be terminated because the map is finished but because the time limit has been reached.

Amelioration for mapping strategy:

The hector mapping we are using should have some parameters tuned to be perfectly stable in Rviz. Implementing Rtabmap may give better results

3.3 Finding objects

Flowchart of the full action:

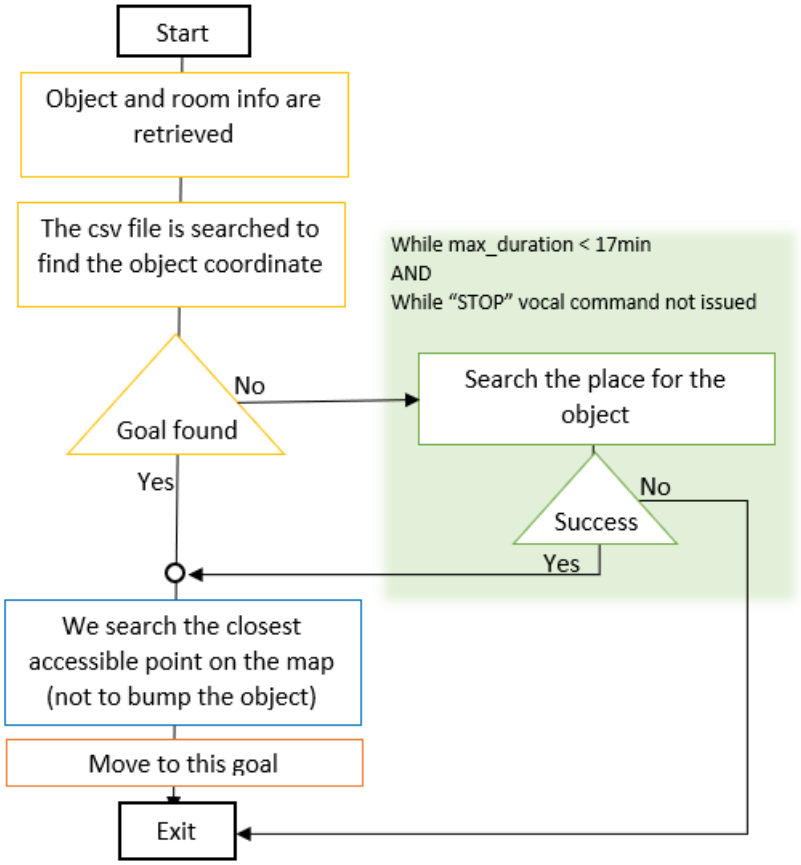


Figure 19: Flow chart of the GET action. yellow box means the action is done in the state machine file, green box means it's done in search_map.py, blue box means set_goal_v3.py and orange box means move.py

The object and the room are extracted from the speech recognition topic and used to search the csv file (semantic map) for the object in question. If the object isn't found in the list a map search is launched.

3.3.1 Searching the map for the object

The map is rescaled (note: the code is usable with any map) and transformed into a grid. We extract from a csv file the angles position of the selected room* and sample this space to extract a succession of goals following 2 rules:

- Potential goals must be 20cm away from any obstacle recorded on the map.
- Potential goals must be 50cm away from any previously saved goals.

Once the room completely sampled the positions are sent to the robot as an action to move** And after each new goal reached the robot do a 360 to avoid missing anything. If a goal isn't reachable because of the inflation of the obstacles, the robot goes to the next without turning. Each time the robot finish turning it checks if the object has been saved in the csv file and if it is the case it ends the search, send the object coordinates and we search the closest accessible point to reach this object.

Amelioration for the searching strategy:

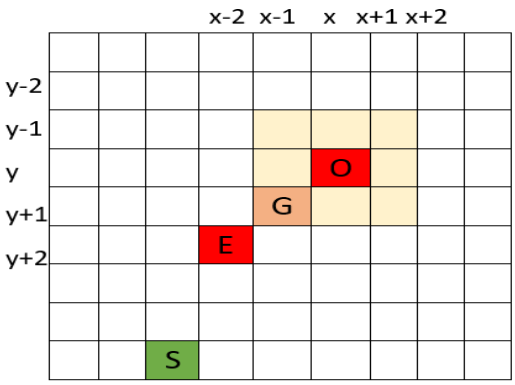
* We have to extract the rooms angles manually, making the map a bit static (the mapping has to be done from the same start position for it to correspond with those angle positions). A solution envisaged and tested is the angle extraction with the Harris Corner strategy, the problem of this technique is that any angle is extracted, not only room angles [4]

** The robot inclines the head while moving, it should be tilted a bit up so that objects could be well detected. The move is thought as an action so this order could be added to the code.

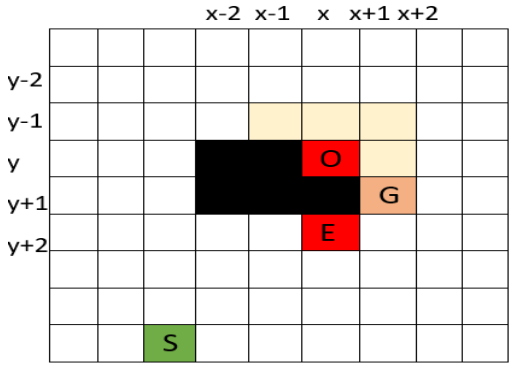
3.3.2 Establishing the closest accessible pose to the object

We have the object position we want to check if the space around it is accessible or not. A search for a free space is effectuated with the object as its centre and a radium of 20cm. The pose to reach considers the actual pose of the robot and where it's coming from and add 10 extra cm to the final position to reach to be sure it doesn't enter the object or obstacle on which is the object.

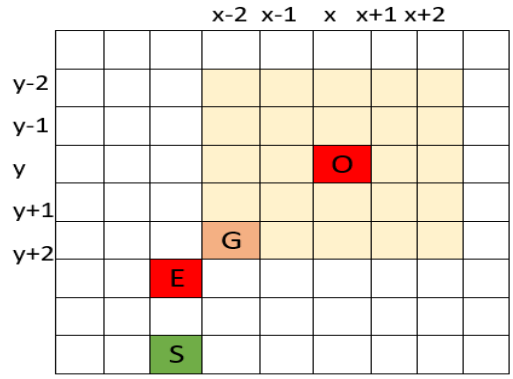
[4] See existing code in : Human-Support-Robot-for-Domestic-Interaction/architecture/src/rubbish/Harris_corner_test.py



(a) first attempt no obstacle



(b) first attempt with obstacle



(c) second attempt no obstacle

Figure 20: Demonstration of the goal set on a girded map. S: the starting pose of the robot. E: the end pose of the robot, G: the goal pose found by the strategy, O: the object pose. The black grids symbolize an obstacle while the orange zone is the zone checked for finding the goal. In the retry operation, only the outline of the square is searched. The end pose is always 10cm away from the goal to take into account the robot's circumference.

Amelioration for the goal set strategy:

Now this operation is conducted only once. While it's probable that the first tentative fails to reach the goal as it's likely that objects will be on top of tables (not seen as obstacles on the map). We should retry several times to search a new closest accessible goal if moving to the previous established one isn't possible. A solution may be to use Rtabmap which allow us to consider the table presence.

Another idea, and the actual strategy planned here (but not fully tested), is, if moving to the goal fails, we retry to search a new goal by expanding the area of research (as (C) in Figure 20). If after several tries no viable solution is found, then the action will exit with a failure. The code is ready to have this feature implemented.

3.4 Welcoming visitors

You can see in Figure 21 the flowchart of the welcome action as it has been realised for the moment in its simplified version. The robot doesn't need to hear "Welcome" specifically to understand it's the action it has to execute but all the synonyms are grouped under this action name.

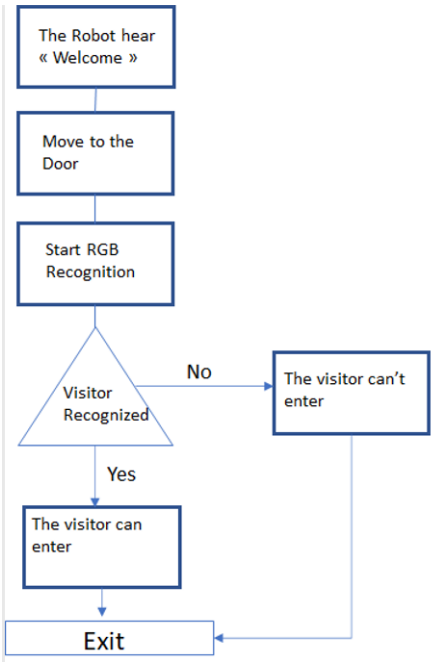


Figure 21: Welcome Flow chart

3.4.1 Speech

Sunbul

To complete this section, we created a script in which if the user says "Please go and check the door" the HSR will be able to understand that the user wants him to walk to the door. The script involves the welcome dialogues too. For example, if the robot recognizes that there is postman at the door, it'll say "Hi Postman, what have you brought for me today?" or will ask him "How are you doing today?".

The part of the project plan was to do the face recognition in which ideally, if the robot goes to check the door and it does not recognises the person. It shall not welcome the stranger or will ask him his details first.

3.4.2 RGB recognition

Brieuc

In order to recognized who is the visitor, we choose to recognized their uniform. For that we are using RGB recognition. Once the robot his in front of the visitor he will look at all the color that are associated to the known visitor and take the dominant one which take as assomption that if a visitor don't have a uniform that as a dominant color it won't be recognized.

3.4.3 HSR Motion

Daria

At the beginning of the action, wherever the robot is, it will move toward the front door, which coordinates are taken from the csv file containing the room. It will stop in the middle of the corridor 60cm away from the door. In order for it to be operable. The motion action is again called once the individual is recognized and its identity requires a motion. Both motions use the same motion principle, one with static coordinates (front door) and the second with flexible coordinates.

Amelioration of the motion functions:
The front door position is only valid for this particular map, because the parameters are coming from a csv file containing the room angles filled manually. An improvement has been proposed in the amelioration of section 3.3.1. Also in this scenario, the motion to go to the front door is repeated in case the first reaching fails. This is not optimal because if there is an obstacle in the way, it may just be blocked there forever.

4 Results

Daria,Sunbul

Our work has been entirely realised on simulation (Covid19 situation). Therefore our tests are not complete to verify the solidity of our architecture. However we tried to submit each part of our architecture to diverse situations and improve their limits.

Mapping

The mapping has been tested from several starting points (not considering the static corner list) to see if the mapping was performing correctly.
The performance was evaluated considering the integrity of the map realized. The more unknown areas, the lower the score. The percentages were estimated comparing the home size to the surface sum of the unknown areas, manually encompassed. After several mappings (starting from 0,0,0 as x,y,z position) we had an average mapping completeness of 80%. The best achieved was 90%. The

area behind the living room low table (see (B) of Figure 18 and Figure 22) was never fully mapped and the areas behind the kitchen counter and bed were rarely mapped. In figure 22 we can observe a typical map with a completion of around 75%

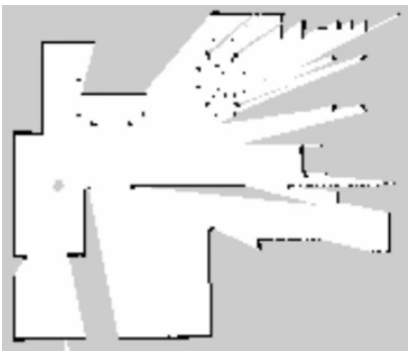


Figure 22: Typical result of the mapping process

The results are easily explained by the starting point. The (0,0,0) position corresponds to the grey circle in the corridor of Figure 22. From this position, the first rotation of the robot encompasses already around 40% of the map. From the bedroom however, the robot could get stuck at a position because of the inflated objects of the room (bed, walls, closet), explaining the 15% score we see in Table 4 (only part of the bedroom and living room were mapped, the robot didn't move).

N° of tests	Starting position (x,y,z)	Average mapping (%)	Worst mapping (%)	Best mapping (%)
7	Corridor (0,0,0)	80%	45%	90%
4	Bathroom	55%	30%	70%
3	Bedroom	60%	15%	75%

Table 4: Table of mapping result, all percentages are an estimation

Get action

The Get action was tested piece by piece, due to some functionalities only fully realized a bit late. Yolo functioning was simulated by writing an object and pose in the csv file (semantic map) for instance. The search map was tested by verifying that the robot fully re-mapped the area designed. The living-room was the less well re-mapped by this function. Half the room not being accessible by the robot (because of the detection of inflated obstacles, the paths are obstruct). Also, in the kitchen remapping objects on the counter may easily be missed for the same reason (the path toward it is impracticable and the robot faces down). The search map (see section 3.3.1) was not tested with yolo after training with the objects of the simulation. It is feared that the robot may look too much down to actually identify them, as they may only briefly appear on the camera

images.

The set of the goal (see section 3.3.2 for an explanation of this function) for the robot motion was tested by giving different positions and verifying if the solution was indeed optimal for the robot (according to the map). The results were 100% adequate. However let's nuance that if the object is in the middle of a table, the robot will have to recalculate its motion mid-way, and the re-calculation of the goal position, even though ready (see section 3.3.2 and the amelioration paragraph), has not been tested.

Speech recognition

The Speech Recognition has been tested to work according to the domestic environment. The data base we created according to the smart-home context included many of the different possible interactions that could be used between user and the robot. To see that speech is well recognised, We tested it with different voices (male, female) and three different accents.

Figure 26 explains the test results. We can observe that the program checks all the possible synonyms of intents from the user to avoid all the confusions.

```
Say something!
hello
Hello Human, how are you today
Say something!
good
i am good too, how can i help you
Say something!
can you please bring an apple from the kitchen
Did you mean can you please bring an apple from the kitchen
Say something!
can you please bring an apple from the kitchen
can you please bring an apple from the kitchen
[('can', 'MD'), ('you', 'PRP'), ('please', 'VB'), ('bring', 'VBG'), ('an', 'DT'), ('apple', 'NN'), ('from', 'IN'), ('the', 'DT'), ('kitchen', 'NN')]
apple
kitchen
verbs [('please', 'VB'), ('bring', 'VBG')]
nouns ['apple', 'kitchen']
please
bring
apple
kitchen
do the action  ['apple', 'bring'] to/from kitchen
```

Figure 23: Test Results Commands with Chinese Accent

```
Say something!
hello
Hello Human, how are you today
Say something!
I am good how are you
i am good too, how can i help you
Say something!
can you please go and check the door
Did you mean can you please go and check the door
Say something!
can you please go and check the door
can you please go and check the door
[('can', 'MD'), ('you', 'PRP'), ('please', 'VB'), ('go', 'VB'), ('and', 'CC'), ('check', 'VB'), ('the', 'DT'), ('door', 'NN')]
door
verbs [('please', 'VB'), ('go', 'VB'), ('check', 'VB')]
nouns ['door']
please
go
check
door
do the action  ['door', 'check'] to/from home
```

Figure 24: Test Results Commands on for welcoming Visitors (British Accent)

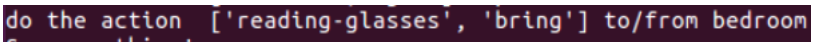


Figure 25: Test Results with Indian Accent

Challenges and Limitations: Speech Recognition is a big challenge itself.Although it has improved a lot in last decade but still speech can be difficult to understand in noisy environments or with different accents. It was observed that ”Danger” command could not be understood by the program in French Accent. Moreover, we observed that sometimes when the users says e.g. ”Bring me soda” , the program take soda as a verb not as a noun. SO the user have to put ”the” eith nouns e.g. ”Bring me the soda” and it works. In our case, the biggest challenge was the integration of architecture. After combining speech with the architecture, because it can not recognise all the objects, it just takes the coordinates and then stops. It can be seen in the figure.

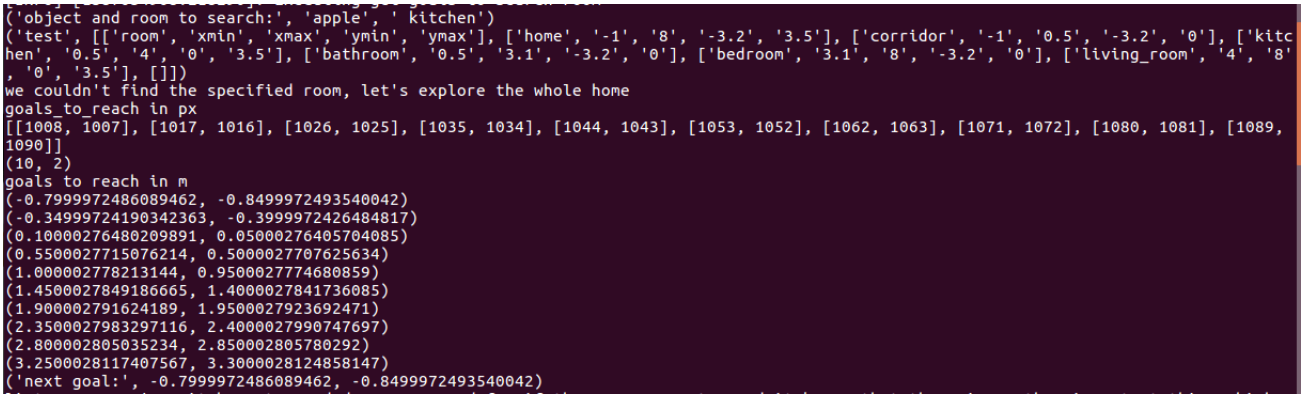


Figure 26: Test Results with Main Architecture

5 Future works

Sunbul, Daria

For future we can improve Speech Recognition by providing a noise reduction technology software. In our case, the voice output generation is mainly hard coded for simplicity reasons. However, we can make use of grammars to get full sentences too.

In parallel the Navigation and mapping could both be improved by considering more cases (we could get stuck, the robot may try to go to the same inaccessible point endlessly) and therefore improve the flexibility of the program. This also apply for each and every action, as they all have a limited range of performance. It means that if unprecedented situations arise, the program is not robust enough to propose an alternative solution.

Yolo is more adapted to the real world than to the simulation. While it can recognize objects from TV-remote and cups to bird, cats or traffic lights in the real world, it struggles to differentiate a

wall from a bird in simulation. Beside, little objects are well recognized under a 2m range approximately. This could be improved with more training.

References

[1] Francesco Amigoni al Aamir Ahmad. Erl consumer european roootic league for consumer service robots. https://www.eu-robotics.net/roboticsleague/upload/documents-2018/ERLConsumer.pdf, 2018.

[2] www.marketsandmarkets.com. Household robots market. https://www.marketsandmarkets.com/Market-Reports/household-robot-market-253781130.html.

[3] Your Guide to the World of Robotics. Human support robot. https://robots.ieee.org/robots/hsr/.

[4] Takashi Yamamoto, Koji Terada, Akiyoshi Ochiai, Fuminori Saito, Yoshiaki Asahara, and Kazuto Murase. Development of human support robot as the research platform of a domestic mobile manipulator. *ROBOMECH journal*, 6(1):4, 2019.

[5] openslam org. Gmapping. https://openslam-org.github.io/gmapping.html.

[6] Tahir Mehmood, Usman Hashmi, Arsalan Akhter, and Ammar Ajmal. Techniques and approaches in robocup@ home-a review. In *2015 International Conference on Information and Communication Technologies (ICICT)*, pages 1–6. IEEE, 2015.

[7] Fabrice Jumel, Jacques Saraydaryan, Raphael Leber, Laëtitia Matignon, Eric Lombardi, Christian Wolf, and Olivier Simonin. Context aware robot architecture, application to the robocup@ home challenge. In *Robot World Cup*, pages 205–216. Springer, 2018.

[8] Vittorio Perera, Tiago Pereira, Jonathan Connell, and Manuela Veloso. Setting up pepper for autonomous navigation and personalized interaction with users. *arXiv preprint arXiv:1704.04797*, 2017.

[9] Sansei Hori, Yutaro Ishida, Yuta Kiyama, Yuichiro Tanaka, Yuki Kuroda, Masataka Hisano, Yuto Imamura, Tomotaka Himaki, Yuma Yoshimoto, Yoshiya Aratani, et al. Hibikino-musashi@ home 2017 team description paper. *arXiv preprint arXiv:1711.05457*, 2017.

[10] Rémi FABRE, Boris ALBAR, Clément DUSSIEUX, Ludwig JOFFROY, Zhe LI, Clément PINET, Jennifer SIMEON, and Sébastien LOTY. Catie robotics@ home 2019 team description paper. 2018.

[11] Mauricio Matamoros, SEIB Viktor, and Dietrich Paulus. Trends, challenges and adopted strategies in robocup@ home. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–6. IEEE, 2019.

[12] Fabrice Jumel, Jacques Saraydaryan, Raphael Leber, Laëtitia Matignon, Eric Lombardi, Christian Wolf, and Olivier Simonin. Context aware robot architecture, application to the robocup@ home challenge. In *Robot World Cup*, pages 205–216. Springer, 2018.

[13] Cristopher Gómez, José Astorga, Nicolás Marticorena, Lukas Pavez, Rodrigo Salas, and Javier Ruiz-del Solar. Uchile peppers 2019 team description paper.

[14] octomap. Octomap- an efficient probabilistic 3d mapping framework based on octrees. `https://octomap.github.io/`.

[15] Mustafa Eliwa, Ahmed Adham, Islam Sami, and Mahmoud Eldeeb. A critical comparison between fast and hector slam algorithms.

[16] introlab. Rtab-map- real-time appearance-based mapping. `http://introlab.github.io/rtabmap/`.

[17] Tahir Mehmood, Usman Hashmi, Arsalan Akhter, and Ammar Ajmal. Techniques and approaches in robocup@ home-a review. In *2015 International Conference on Information and Communication Technologies (ICICT)*, pages 1–6. IEEE, 2015.

[18] Elaine L Appleton. Put usability to the test. *Datamation*, 39(14):61–62, 1993.

[19] espeak. `http://espeak.sourceforge.net/`.

[20] A Nishajith, J Nivedha, Shilpa S Nair, and J Mohammed Shaffi. Smart cap-wearable visual guidance system for blind. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 275–278. IEEE, 2018.

[21] Ruth Breckinridge Church, Philip Garber, and Kathryn Rogalski. The role of gesture in memory and social communication. *Gesture*, 7(2):137–158, 2007.

[22] Paul Bremner and Ute Leonards. Speech and gesture emphasis effects for robotic and human communicators-a direct comparison. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 255–262. IEEE, 2015.

[23] Laura Pérez-Mayos, Mireia Farrús, and J Adell. Part-of-speech and prosody-based approaches for robot speech and gesture synchronization. *Journal of Intelligent & Robotic Systems*, pages 1–11, 2019.

[24] James Kennedy, Séverin Lemaignan, Caroline Montassier, Pauline Lavalade, Bahar Irfan, Fotios Papadopoulos, Emmanuel Senft, and Tony Belpaeme. Child speech recognition in human-robot interaction: evaluations and recommendations. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 82–90, 2017.

[25] googleapi. `https://pythonspot.com/speech-recognition-using-google-speech-a`

[26] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[27] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection.

[28] Kazuto Murase.

[29] hsr developer guide. `https://docs.hsr.io/hsrdevelopmanualen/advance/jetsonyolo.html#id2`.

[30] Raphael Memmesheimer, Vikor Seib, and Dietrich Paulus. homer@ unikoblenz: winning team of the robocup@ home open platform league 2017. In *Robot World Cup*, pages 509–520. Springer, 2017.

[31] PYujin Tang,mes Borg, Yusuke Kurose, Francesco Savarese, TakayoshiTakayanagi, Mohammad Reza Motallebi, Antonio Tejero-de-Pablos, YingyiWen, Toshihiko Matsuura, Jen-Yen Chang, Li Yang, Yoshitaka Ushiku andTatsuya Harada. Utmil@home dspl2018 team description paper. pages 1–11, 2018.

[32] Kamerider sspl@home 2019team description paper.

[33] Jacob Menashe B, Josh Kelle, Katie Genter, Josiah Hanna, Elad Liebman, Sanmit Narvekar, Ruohan Zhang, and Peter Stone. RoboCup 2017: Robot World Cup XXI. 11175(January):45–58, 2018.

[34] Mohit Vaishnav David Weiss Malav Bateriwala Eduardo Ochoa, Roger Pi Roig. Robotic system science class project - erl competition. *no publication*, 2019.

[35] Geeks for geeks. A* search algorithm. https://www.geeksforgeeks.org/a-search-algorithm/.