The process of design and implementation of an artificial intelligence for a domestic multitask robot

Arnau Canyadell Miquel

June 15th 2018

Director: Luis Sentis
The Human Centered Robotics Lab
Department of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin

Bachelor Degree in Informatics Engineering
(Mention in Computing)

Facultat d’Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech
# Index

1. Abstract 3

1. Resum (en català) 4

1. Resumen (en español) 5

2. Introduction 6

2.1. The Toyota HSR Robot 7

3. The design of the top-level Artificial Intelligence 8

3.1. Requirements of the AI 8

3.2. Development framework 8

3.3. Design of the AI 9

3.4. Subtasks and modularity 10

3.5. Knowledge representation 11

4. Perception algorithms 13

4.1. Motivation. Object manipulation 13

4.2. HSR’s sensors 14

4.3. Tabletop segmentation 14

4.3.1. Assumptions 14

4.3.2. Algorithm steps 15

4.3.2.1. Euclidean Cluster Extraction 15

4.4. Object identification 17

4.2.1. Retraining the YOLO neural network 18

4.2.2. Using YOLO detections with the tabletop segmentation algorithm’s output 18

5. Robocup@Home 2018 tasks high-level design and performance 21

5.1. Dishwasher task high-level planner 21

6. Conclusions 23

7. Bibliography 24

8. List of abbreviations 25
Special thanks

Before starting this report I want to thank everyone that made this project possible.

First of all, I want to give my most sincere thanks to all the people in my university UPC that gave me the chance to do the “projecte de final de grau” abroad: from CFIS, Eduard Alarcón and Miguel Ángel Barja for finding the UT for me and the rest of the team for their incredible job; from FIB all the workers in International Relations Department for making everything easy; and from the funding entity Fundació Privada Cellex that makes these kind f projects economically possible.

Secondly, the Generalitat de Catalunya for their economical support via MOBINT grant.

Thirdly, my deepest thanks to my tutor Luis Sentís for taking me in, supporting me, counseling me and teaching me so much.

In forth place, to Alex Rodríguez for discovering Texas to me.

And last but definately not least, I thank a lot my lab mates and project mates in UT for all the knowledge shared and experiences lived together, specially the coworkers Minkyu and Nico, from whom I have learnt so much, and Jason Hart, who has been a dear and necessary leader for the Austin Villa 2018 project.
1. Abstract

Home assistance is a classical objective of robotics. With the current technology in both software and hardware performing some of the basic tasks can be achieved even by a team of undergraduate and even high school students. This project follows the artificial intelligence developed by a team of students and researchers from the University of Texas at Austin for the Toyota HSR robot and the contributions made by the author of the report. This artificial intelligence will compete at the Robocup@Home Robotics competition in Montréal on June 2018.

The contributions of the author discussed in this report are the integration of two perception algorithms and the design of a top-level state machine for the “dishwasher task”. The perception algorithms are the tabletop segmentation using Euclidean cluster extraction and object identification using a neural network.
1. Resum (en català)

L’assistència domèstica és un objectiu clàssic de la robòtica. Amb la tecnologia actual tant en software com en hardware dur a terme algunes tasques bàsiques ho pot aconseguir un equip d’estudiants de grau o fins i tot de secundària. Aquest projecte segueix la intel·ligència artificial desenvolupada per un equip d’investigadors de la University of Texas at Austin pel robot Toyota HSR i les contribucions fetes per l’autor de la memòria. La intel·ligència artificial en qüestió competirà a la competició de robòtica Robocup@Home a Montréal el juny de 2018.

Les contribucions de l’autor discutides en la memòria són la integració de dos algoritmes de percepció i el disseny de una màquina d’estats de primer nivell per la “tasca del rentaplats”. Els algoritmes de percepció són la tabletop segmentation (segmentació dels objectes sobre la taula) utilitzant Euclidean cluster extraction (extracció de clústers euclideans) i la identificació d’objectes utilitzant una xarxa neuronal.
1. Resumen (en español)

La asistencia doméstica es un objectio clásico de la robótica. Con la tecnología actual tanto en software como en hardware llebar a cabo algunas tareas basicas lo puede conseguir un equipo de estudiantes de grado o incluso de secundaria. Este proyecto sigue la inteligencia artificial desarrollada por un equipo de investigadores de la University of Texas at Austin para el robot Toyota HSR y las contribuciones hechas por el autor de la memoria. La inteligencia artificial en qüestión competirá en la competición de robótica Robocup@Home en Montréal el junio de 2018.

Las contribuciones del autor discutidas en la memoria son la integración de dos algoritmos de percepción y el diseño de una máquina de estados de primer nivel para la “tarea del lavaplatos”. Los algoritmos de percepción son la tabletop segmentation (segmentación de los objetos enima de la mesa) utilizanto Euclidean cluster extraction (extracción de clústeres euclideanos) y la identificación de objetos utilizanto una red neuronal.
2. Introduction

Domestic tasks have traditionally consumed a lot of human time. Also, elders and people with disabilities usually have difficulties when trying to perform activities such as clearing the table, doing the laundry, cleaning or storing groceries. The generalized use of home appliances such as washing machines has helped a lot to increase human productivity. More recently, cleaning robots have made their way into the markets. However, domestic chores are very broad and current solutions are designed to perform just one task. Furthermore, traditional appliances (and cleaning robots) do not solve the problem of assisting handicapped people.

A multi-purpose domestic robot faces a lot of technical challenges. The robot needs to interact with the environment (navigating and avoiding obstacles, grasping and putting down different kinds of objects, ...) and people (communication, giving and receiving objects, ...) and is also needs to make decisions and plans. In order to achieve that a robust AI that integrates and uses a wide set of packages is needed.

The packages or services that this whole AI is composed of should perform the robot’s usual tasks. A non-exhaustive list of these tasks is:

- Speech recognition and order recognition
- Navigation avoiding obstacles
- Person detection and identification
- Object detection and identification
- Object grasping
- Object placing
- Object handover (to and from a human)

Another important part of the AI is the knowledge representation: a database that stores all the knowledge that is either pre-known or gathered by the robot’s sensors such as the map of the house, the objects locations, the known humans or the orders given by humans.

The recent development of free frameworks for robotics programming has made it possible for non-specialized developers to program a whole robot by joining different pieces. The Toyota Human Support Robot (HSR) is an example of a robot that can be programmed using ROS and a simple API.
2.1. The Toyota HSR Robot

The HSR is a robot designed to be used as an domestic robot and used mainly for research and educational purposes. On its head, it has an omni-directional base and one robotic arm with one hand. Its main sensors are a microphone, a 3D distance sensor and a stereo an a wide-angle cameras, an inertial measurement unit (IMU); a hand force sensor and a hand camera; and finally on its base it has a laser range sensor, a bumper safety sensor and a magnetic sensor.

![Figure 1. Toyota HSR and its units.](image-url)
3. The design of the top-level Artificial Intelligence

3.1. Requirements of the AI

The Encyclopaedia Britannica defines artificial intelligence as “the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings”.

In this project, the aim is to design a software able to perform general tasks related to home activities. The project is focused on the Robocup@Home competition, but the tasks for the competition are a good representation for tasks that a domestic robot is supposed to do (a more detailed description of the robot’s performance with regard to the specific tasks is given later). In general, we want the robot to be able to:

- Transcribe and understand commands given by humans
- Answer to questions related to known topics or topics whose answer can be of general knowledge
- Map the shape of a flat and navigate around it (including tasks such as opening doors or removing small objects from the path)
- Recognize humans and identify those who have been seen previously
- Perceive and identify the objects and elements that surround the robot
- Manipulate small objects: grasp, transport, drop.
- Make decisions regarding the tasks given and act in case of failure

The concept of artificial intelligence is very broad, just as the concept of intelligence itself. However, it is generally accepted that a piece of software that has some degree of complexity, has to perform different tasks and has to make decisions regarding how to proceed can be considered an artificial intelligence. Whether this assumption is correct or not, in this project any software able to perform the previous set of tasks will be referred to as an artificial intelligence.

3.2. Development framework

ROS (Robot Operating System) is a very widespread framework for robotics. It handles message transmission from sensors, to actuators, and between different pieces of software. It also eases the task of integrating different pieces of software (packages). In spite of its name, ROS is not an operating system, but a framework or middleware. It is distributed under a free software BSD license (Berkeley Software Distribution), and it was mainly developed in its beginning by the company Willow Garage in 2007, fulfilling the need of a well-tested implementation of a framework for creating flexible, dynamic software systems intended for robotics use.
Since then, ROS has become a chance for many to develop robotics AIs. Many robot manufacturers have included ROS natively in their products (being this the case for Toyota HSR).

The ease of learning, its modularity and abstraction, and the ease of integration with the HSR are the main reasons why this framework was chosen for this project. The availability of many online resources for learning ROS are also a pro for ROS in projects in which many students with different levels of knowledge take part.

However, ROS framework has limitations which make it a not so optimal tool for certain purposes. ROS encapsulation of messages adds a big overhead in message transmission, which might be critical in many robotics applications, specially of the computing units are external to the robot itself and the sensors’ information needs to be transmitted over the network.

3.3. Design of the AI

When thinking about the tasks that need to carried by the AI, the concept of state is naturally brought to light. If the robot has grabbed an object, it is now in a state such that it will not proceed to grab another object without first dropping the one that is being held (assuming we are working with a one-handed robot such as the HSR). Moreover, a state machine is a very helpful tool to describe the behaviour of an AI.

For instance, a solution to the task of grabbing an object from a cupboard can be described as follows:

1. Navigate to a position in front of the cupboard.
2. Scan the cupboard for objects.
3. Identify the (or decide which) object to grasp.
4. Grasp the object.
5. Place the robot’s arm back to a standard position while holding the object.

This would be the linear code to do the task. However, considering that each of these tasks can finish in a state different that success, different actions can be taken. For instance, in task 4, if the object is not successfully grasped then the robot could return to task 2 and scan the cupboard again.

In the ROS framework, there is a well-known state machine implementation called SMACH (which stands for State MACHine) which allows high-level abstraction of tasks, just as described previously. This tool allows fast prototyping of complex state machines, and many tools for introspection. Its high-level abstraction helps in having a much cleaner code, which is easier to maintain and to share between different developers.
However, just like ROS adds overhead to the message transmission, SMACH adds overhead to the execution of the workflow, making it not suitable for low-level systems that require high performance.

3.4. Subtasks and modularity

Another very important aspect of using the state machine is the modularity of the small tasks of the robot. Many different tasks involve performing a set of exactly the same subtask. Navigation is an obvious example of this, since it’s a subtask necessary for almost any task because it’s very uncommon that the robot stays in the same place all the time. Other examples of common tasks are human orders listening, transcription and understanding; object perception; or object manipulation. Rather than developing a very good algorithm to perform excellent in a specific task for a competition, the most interesting aim is to develop a general algorithm that solves a specific problem in a wide range of contexts.

Following this philosophy, the team in charge of developing the AI for the Robocup@Home 2018 competition from the University of Texas (UT), named Austin Villa, decided to divide the common tasks in different sets: perception, manipulation, navigation, speech recognition. Those are all used in the different high-level task planners depending on the context.

In the 2018 edition, Austin Villa has put a lot of effort into creating ROS Action Servers for each of this subtask. An action server is a kind of ROS package with a specific interface that allows management of different events such as abortion of the task due to some kind of failure or preemption of the task due to some events external to the task (eg a new command from a human asking to stop). Figure 2 gives a general overview of this interface.

Server State Transitions

![Server State Transitions Diagram]

10
The author of this report centered its efforts mainly in perception tasks, but also in manipulation, and finally contributed to most of the implementation of the high-level planner for the “dish-washer task”.

3.5. Knowledge representation

One last aspect necessary for the AI design is its “memory”. The robot is expected to remember what has learnt through perception: map of the environment, what it has seen and what it has heard (including the orders it has received).

In order for the robot to be able to “reason” about its memory, what really needs to be stored is not raw data, but rather concepts with meaning. For instance, it’s meaningless and extremely inefficient to store the whole history of images seen by the robot. Instead, the aim is to remember facts like “the apple is on the kitchen’s table”.

In order to do so, a “knowledge representation” has been developed by a subteam of the Austin Villa team. This project aims to create a representation of the objects that contains useful information about facts known. The first approach is to store information about detected objects in the arena’s its properties and its relations with other objects. An example of that can be seen in Figure 3.
Figure 3. Knowledge Base diagram representing an example of the concepts stored. Image created by the Austin Villa team.
4. Perception algorithms

This chapter of the report focuses on some of the perception algorithms used by Austin Villa for Robocup@Home 2018, which contain the main contributions from the author of the report. The two algorithms explained here are the tabletop segmentation, which takes as input a cloud of points detected by sensors and returns the clusters of objects that could be places over a horizontal surface, and the object identification algorithm, whose input is an image and the situation of an object within the image and outputs the kind of object it is.

4.1. Motivation. Object manipulation

One of the robot’s most important features is its ability to manipulate objects. Manipulation, in the context of this report and due to the HSR’s physical abilities, should be thought of object grasping, transportation and dropping.

![Image of the HSR’s hand with its sensors]

Figure 4. Image of the HSR’s hand with its sensors

In order to perform this object manipulation, an autonomous system needs to know firstly where is the object that will need to be collected by the robotic hand. Finding the location of objects is the main objective of the tabletop segmentation. However, picking a random object is probably something useless by itself. It will be necessary to identify which objects we have in order to being able to decide, by a more task-conscious higher-level algorithm, which of those objects is the one the robot should pick.
4.2. HSR’s sensors

Previous to deciding the algorithm to apply to solve these two problems, it is necessary to know what data can be used. This can be solved looking at the robot’s specifications, which are available for developers in a private website [https://www.hsr.io/, June 14th 2018].

From the information in the developers’ guide, the sensors that are useful for the stated purposes are:

1. Head three-dimensional distance sensor (RGB-D)
2. Head stereo camera
3. Head wide-angle camera

The stereo and the wide-angle cameras are basically cameras, which provide different real-time images of what the robot sees. The wide-angle camera gives a wider range image while the stereo camera is more accurate. This last sensor is more suitable for object identification since it provides the highest definition image of what the robot sees.

On the other hand, the RGB-D sensor (standing for Red Green Blue - Distance) is designed to provide a 3D map of what is in front of the robot’s face. This is outputted as a set of 3D (6D including colors) points in the space, with a precision of about 1 cm for elements 2 m in front of the robot. For obvious reasons this sensor will be the most adequate for tabletop segmentation.

4.3. Tabletop segmentation

Given the information provided by the RGB-D sensor, a real-time-updated set of 3D points detected (the color is not needed for this algorithm so it will be ignored), it is necessary to distinguish which of those points are more likely to be part of objects that can be grasped by the robot’s hand (ie objects smaller than around 15cm).

4.3.1. Assumptions

The first assumption that will be made for this algorithm is that the target object is placed on the top of a horizontal surface (presumably a table or a shelf). This assumption or restriction is the one that gives the name to the algorithm. It is not a really restrictive assumption when thinking about domestic situations.

The other assumption is about the object’s dimensions. It needs to be taller than a minimum amount or otherwise (a) the object will not be detected and (b) the hand will not be able to grasp it. This restriction might need to be changed depending on the context (this is the case of the “dishwasher task” where silverware needs to be
picked from a table. It also needs to be shorter than a given amount, because the robot will not be able to pick a really big object.

On the horizontal axes, the object is mainly limited by the hand’s maximum opening distance, which is approximately 15 cm as it has been stated before.

4.3.2. Algorithm steps
The first step is to detect horizontal surfaces that are likely to be tables or shelves. Basically, any horizontal surface in a height between 0.2 and 1.7 m could do (the floor and the ceiling should not be counted and the height has to be such that the object is reachable by the robot’s arm).

The algorithm used to detect horizontal surfaces is called “Plane Segmentation”, and it is a variation of the “Euclidean Cluster Extraction”.

Previous to applying the plane segmentation the data is downsampled so that the data is faster to treat in real-time. The downsampling parameter was set empirically after testing the test with background processes in the robot, in which it was checked that the results were minimally affected while trying to maximize the performance. Finally the downsampling parameter was set to 1 cm.

4.3.2.1. Euclidean Cluster Extraction
The Euclidean Cluster Extraction algorithm, an implementation of which can be found in the PCL Library (a free software library developed by the Open Perception Foundation), finds clusters of points in 3D space.

This algorithm is quite simple. It starts by transforming the input data into a structure that is faster to search the points neighbors, such as a Kd-tree. Then all the points are grouped into clusters where each cluster is a group of points that are “neighbors”.

A pseudo-code version of the algorithm,

1. Create a Kd-tree representation for the input point cloud dataset $P$;
2. Set up an empty list of clusters $C$, and a queue of the points that need to be checked $Q$;
3. Then for every point $p_i \in P$, perform the following steps:
   a. Add $p_i$ to the current queue $Q$;
   b. For every point $p_i \in P$ do:
      i. Search for the set $P_i^k$ of point neighbors of $p_i$ in a sphere with radius $r < d_{th}$;
ii. For every neighbor \( p_i^k \in P_i^k \), check if the point has already been processed, and if not add it to \( Q \);

c. When the list of all points in \( Q \) has been processed, add \( Q \) to the list of clusters \( C \), and reset \( Q \) to an empty list;

4. The algorithm terminates when all points \( p_i \in P \) have been processed and are now part of the list of point clusters \( C \);

This pseudo-code has been taken from the PCL documentation at their website [http://pointclouds.org/documentation/tutorials/cluster_extraction.php, June 14th 2018].

The plane segmentation algorithm is a variation of the Euclidean Clustering Algorithm which simply adds a constraint to clusters. This constraint eliminates parts of the clusters that are further from the main cluster vertically.

Figure 5. Tabletop segmentation algorithm run on simulator. In green there are the detected surfaces’ bounding boxes and in white there are the segmented objects. In this image it can be appreciated that one shelf (the one placed more or less at the robots’ eyes’ height) is not correctly detected. This is because of the poor angle of vision that the robot has of this surface.
After the plane segmentation algorithm finishes, the surface (or surfaces) detected are stored. Then all the points that are placed above one of the surfaces are selected and the rest are discarded. A point is considered to be above a surface if it is placed between a minimum threshold distance and a maximum object height distance over any point of the area of the surface. The minimum threshold is necessary so that the surface’s points themselves are not selected, and it was empirically set to 2 cm. The maximum height is necessary because we are discarding larger objects anyway and because there might be unrelated objects. Also, since one usual situation is to apply the algorithm on a cupboard with multiple shelves, if one shelf is above another the points are cropped.

Then the Euclidean Clustering Algorithm is applied to the points selected and the clusters obtained are the candidates to be final objects. From these candidates, the ones that don’t fulfil the size requirements are discarded and the rest are considered to be target objects. The output of this whole process can be simplified by providing a bounding box for each object instead of the whole pointcloud. This bounding box can be described with a 3D central point and x, y and z sizes.

**4.4. Object identification**

Once the target objects have been obtained those need to be identified. The inputs for performing this task are the bounding boxes of the detected objects from the tabletop segmentation and the camera image obtained with the stereo camera.
A lot of research has been done on the topic of identifying an object given an image. Many different algorithms have been deployed on the topic and the most successful ones are using deep neural networks trained over big amounts of data.

One of these implementations is called YOLO (You Only Look Once) and it has some advantages that make it suitable for the current purpose:

(a) It is free to use.
(b) It is a fast algorithm that supports real-time video inputs.
(c) There exist some pre-trained neural networks based on some datasets but the neural network is prepared to easily being retrained with new data.

When being tested in HSR experiments, YOLO trained with COCO dataset showed a good performance when the objects being observed were one of the well-known objects present in the dataset. However, many objects you could expect to find in domestic environments are not in the dataset and therefore they either not detected or labelled incorrectly.

Also, the difference of image resolution between datasets and the robot’s sensors make it slightly harder for the algorithm to label objects correctly.

In the following subchapters the topics of retraining the neural network and identifying the objects given the tabletop segmentation’s output will be covered.

4.2.1. Retraining the YOLO neural network
Retraining YOLO is a must given the diversity of objects that the robot might find and that are not contemplated in any well-known database.

It is usually done using models that have been either pre-trained or already trained since training from scratch takes a lot of computation. For our goal it is not necessary to use a pre-trained network instead of a fully trained network since mainly we need to add new labels to the dataset.

In order to achieve a system able to detecting a specific set of objects that are known to be present in the robot’s environment, a re-training of the COCO dataset neural network can be done by taking some images of the objects from different angles, distances and illumination configurations. In each of these images, the position of the object needs to be manually set and then the re-training algorithm can be used.

4.2.2. Using YOLO detections with the tabletop segmentation algorithm’s output
Given the bounding box of a 3D cluster detected over a horizontal surface, the aim os to know what object this is. Using a coordinate transformation from the 3D space to the camera space, which can be calculated given the robot’s specifications, it is
possible to get a 2D representation of the object in camera space. By matching this locations with the closest detection given by YOLO, the identification of the object is obtained.

This solution, however, results problematic when there are several objects in front of each other from the robot’s point of view. This can be partially solved by only taking as the result target the object closest to the robot (in the case of two tabletop segmentation objects being very close in camera space). It si different the case where two detections are very close in camera space and it is impossible for the robot to know which one belongs to the tabletop segmentation detected object. In this case, the only solution is to tell the robot to move to the right or the left and look again.

After getting the final object, the task of the perception algorithm is done. It is then necessary to consult in the knowledge representation whether this object can be grasped or not, and if so then the manipulation task should be in charge of grasping.

Figure 7. 3D pointcloud input on the left and YOLO image detections on the right, both run on simulator.
Figure 8. Same situation as Figure 7 but instead of the 3D input point cloud there are the tabletop segmentation detections (blue for the surface and green for the objects).
5. Robocup@Home 2018 tasks high-level design and performance

Overall, the Austin Villa team made a lot of progress in the general approach of the AI and improved a lot of specific tasks, such as door opening and human following,

5.1. Dishwasher task high-level planner

The high-level planner for the dishwasher task follows the following scheme:

(If not stated otherwise, the state finishes with going to the next state.)

1. Navigate to dishwasher
2. Open dishwasher
   a. Success: go to 4
   b. Failure: go to 3
3. Ask to open the dishwasher
4. Navigate to table position
5. Move to perception pose
6. Perceive table (performs the tabletop and identifying algorithms)
   a. Success: go to 7
   b. Failure: go to 4
7. Pick object
   a. Success: go to 8
   b. Failure: go to 4
8. Move to object holding position
9. Navigate to dishwasher
10. Place object inside of the dishwasher carefully
    a. Success (counter of 5): go to 4
    b. Success (6th time): go to 11
    c. Failure: go to 4
11. Navigate to cascade pot location
12. Move to perception pose
13. Perceive table using cascade pot specifically trained neural network
    a. Success: go to 14
    b. Failure: go to 11
14. Pick object
    a. Success: go to 15
    b. Failure: go to 11
15. Move to object holding position
16. Navigate to dishwasher
17. Place object inside of the dishwasher carefully
18. Move to normal position
19. Exit arena

This task follows the RoboCup@Home 2018 Rulebook which can be found at the website [https://github.com/RoboCupAtHome/RuleBook, June 14th 2018].
6. Conclusions

Throughout the development of this project I have realized how open the world of Robotics programming has become to not so experienced programmers, such as undergraduates or even high schoolers. Given the tools and some guide, they (we) can autonomously build an AI that performs some basic (but at the same time really complex) task.

From an educational point of view, developing for a robot such as the HSR is a huge opportunity to learn a lot; however it’s a very expensive tool that cannot be used massively.

Even though its price and intense development by Toyota, the HSR lacks more higher-quality sensors and specially a lot of computation power in order to perform human-like tasks.
7. Bibliography

Papers


Joseph Redmon, Ali Farhadi: YOLOv3: An Incremental Improvement


Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, Wolfram Burgard: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees

Websites

HSR developers guide: https://www.hsr.io, June 14th 2018

ROS Wiki: http://wiki.ros.org, 14th June 2018

PointCloud Library documentation: http://pointclouds.org/documentation, June 14th 2018

Robocup@Home 2018 Rulebook: https://github.com/RoboCupAtHome/RuleBook, June 14th 2018
8. List of abbreviations

AI: Artificial Intelligence
COCO (dataset): Common Objects in COntext
(Toyota) HSR: Human Support Robot
PCL: PointCloud Library
UT: University of Texas