
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2013 Informática

Paulo Alexandre Paiva da Fonseca

MyDroid: Teaching my Robot with Android

MyDroid: Ensinar o meu robô com o Android


Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2013 Informática

Paulo Alexandre Paiva da Fonseca

MyDroid: Teaching my Robot with Android

MyDroid: Ensinar o meu robô com o Android

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor José Maria Amaral Fernandes, Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Luís Seabra Lopes, Professor associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Este trabalho é dedicado à minha família que me forneceu todas as possibilidades de aqui chegar.

o júri

presidente	Professor Doutor Joaquim João Estrela Ribeiro Silvestre Madeira Professor Auxiliar, Universidade de Aveiro
vogal – arguente principal	Professor Doutor Pedro Lopes da Silva Mariano Professora Auxiliar Convidado, Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa
vogal - orientador	Professor Doutor José Maria Amaral Fernandes Professor Auxiliar, Universidade de Aveiro

agradecimentos

Um agradecimento muito especial à minha família, por me ter proporcionado alcançar esta etapa.

Aos meus amigos que me acompanharam e ajudaram nestes últimos anos. Sem eles também não seria possível.

Ao grupo de robótica da Universidade de Aveiro, em particular ao Aneesh Chauhan, que me proporcionaram um ambiente agradável e ajuda no desenrolar da componente prática deste projeto.

Um último agradecimento aos professores que me acompanharam ao longo desta vida académica e que me transmitiram o seu conhecimento. Em particular aos professores José Maria Fernandes e Luís Seabra Lopes que abriram uma porta para que este projeto se pudesse concretizar.

palavras-chave Smartphone, interação, robô, aprendizagem, sensores.

resumo Os robôs começaram a ser usados em diversas áreas. Os principais objectivos são a optimização de trabalho e redução de custos. No entanto o modo de interacção com estes equipamentos parece não acompanhar esta evolução. Com o surgimento dos smartphones chega uma nova oportunidade. Estes pequenos dispositivos têm uma elevada capacidade de processamento e estão equipamentos com um conjunto de recursos que podem ser usados na interacção homem-máquina. Esta dissertação propõe um conjunto de soluções para interação com robôs, onde o smartphone é o meio usado para interagir (ao nivel de controlo e ensino). Pretende-se assim avaliar se o smartphone é uma alternativa viável para a interação.

keywords

Smartphone, interaction, robot, learning, sensors.

resumo

Robots started to be used in several areas. The main objectives are work optimization and cost reduction. However, an interaction method with these devices does not seem to follow this evolution. With the rise of smartphones comes a new opportunity. These small devices have a high processing capacity and are equipped with a set of resources that can be used for man-machine interaction. This thesis proposes a set of solutions for interaction with robots, where a smartphone is the instrument used to interact (both commanding and teaching). It is intended to assess whether the smartphone is a viable alternative for interaction.

INDEX

FIGURES INDEX

ACRONYMS TABLE

Term	Description
AP	Access Point
AR	Augmented Reality
HMI	Human Machine Interaction
HMMs	Hidden Markov Models
OS	Operating System
RACE	Robustness by Autonomous Competence Enhancement
ROS	Robot Operating System
SDK	Software Development Kit
SR	Speech Recognition
STT	Speech to Text
TTS	Text to Speech
WLAN	Wireless Local Area Network

1. Introduction

Technology has been evolving greatly in recent years. New electronic devices are constantly emerging in order to assist humans in their tasks. Smartphones are a great example of these devices. As they have larger processing capacities than regular cell phones, its cost-benefit ratio makes them the day-by-day user gadget of choice. The main motivation for acquiring smartphones is to access a wide range of features and services through them. Currently smartphones can be compared with small mobile computers.

Low cost electronics has also led the way to the rise of new autonomous machines intended to perform tasks normally performed by humans. These machines vary from industrial machinery to simple domestic robots that help people in their daily lives. The adoption of this type of equipment's is majorly supported by their lower operational costs and better performance, especially when compared with human workers.

The level and types of human machine interaction has evolved, however the most techniques still underperform. Most of these robotic systems often cannot be configured to do heterogeneous tasks. Moreover when they do, the configuration procedure often boils down to reprogramming using buttons or other unintuitive ways. Keep in mind that more complex forms of reprogramming are not easily performed by most ordinary users, which reduces its utility.

1.1. Motivation

Robots have already started to be used in several areas. However interacting with them still remains unchanged. In most cases, it still relies on a procedural nature where the robot receives and complies with specific and objective orders. There are not wide spread examples of robots with high interaction levels. There clearly is the need for more simple and intuitive interactions that enable robots to a larger number of users.

Smartphones present a new opportunity for more natural interaction between humans and robots. Besides being almost ubiquitous, the general population is accustomed to interact with smartphones (to make calls, take pictures, email) and their resources. Moreover, these devices provide flexible connectivity solutions (e.g. Bluetooth and Wi-Fi) and have multiple sensors that can be used to achieve higher levels of interaction (e.g.

accelerometer, cameras). Using the Software Development Kit (SDK), the implementation of new customized applications capable of accessing all the smartphone's resources is even easier. Some of these resources may be used to interact with robots.

Therefore it is natural to investigate if smartphones can support solutions to interact more efficiently with robots. Either by using standard command-based approaches or other non-conventional approaches, mainly based on implicit communication (e.g. move the hand towards a direction).

1.2. Objectives

This thesis aims to explore interfaces for interacting with robots through smartphones. Smartphones are equipped with a set of resources that allow interaction in several ways (e.g. speech and motion). We aim to explore them as accessible communication interfaces in order to improve human interaction with robots.

We will explore three specific interaction scenarios:

- Command specific instructions
- Inquire about robot's abilities
- Teach the robot (new abilities and objects)

In order to command robots to perform certain tasks, we will explore not only the conventional interaction options, through menus or command line, but also more advanced ways by using smartphone resources, such as accelerometers and gyroscopes. These methods are potentially more intuitive to the user, especially if simple and natural movements are used to interact with a robot, similar to sign language (e.g. wave the smartphone to call the robot).

Inquire about robot's abilities is necessary to implement a flexible smartphone-based application. Thus, the application can be adaptable to multiple contexts as opposed to a fixed set of predefined robot abilities and usage scenarios. These abilities will likely be commanded, in instruction format, in order to make the robot perform different tasks.

We will address two teaching/learning related interactions:

1. Teach new tasks, to learn based on its competences;
2. Teach new objects with photos taken by smartphone and tagged by a smartphone user, with the hope that the robot can learn to recognize these objects later.

In conclusion, the interaction with robots, through smartphones, can provide a favorable experience for the two sides, the user and the robot. In the user point of view, the smartphone–based interaction is easier and more intuitive, as the user is already

familiarized with smartphone features. On the other hand, the smartphone provides agile methodologies for communication with robot, minimizing the communication gap between user and robot.

On a technical point of view, this thesis has provided the following contributions:

- Implementation of a communication channel between smartphone and robot (back-end).
- Implementation of an interface application for smartphone. It allows users to command the robot (front-end). Moreover it contains the necessary tools to control and teach tasks;
- Design and implement more intuitive input methods, like motion;
- Design and implement a system that uses smartphone camera for teaching objects (vision);

We assume that the robot's software system provides a communication interface to interact with it, namely to access its services and relevant robot internal information. Our focus is not on the development of robot learning capabilities. We also assume the robot already has them.

1.3. Thesis structure

The first chapter consists of a short introduction of the thesis. It provides a general problem description, as well as major goals for this project.

The second part is an analysis of the relevant work in this research field. A critical commentary is also provided in order to realize whether the techniques adopted bring good results or not. Some of these references have been used as the basis for this thesis.

Having performed a literature review, this information will be used for defining requirements associated with our goal. The architecture of our proposal is presented. The information detailed in this chapter was essential for developing this project.

Upon the completion of our project, we have performed some tests and analyzed results in order to verify the robustness and usability of our methods. In this chapter, these results are examined, which will also serve to point to further contributions. In the final chapter, some conclusions are drawn from the project.

2. Relevant technologies and literature survey

This chapter presents a survey of the state of the art related to interaction with robots, using smartphones. It presents the background needed to contextualize, i.e. methodologies used for communication with robots and the workflow itself, but also understand the existing constrains that conditioned several decisions along this work, namely those related with interaction mechanisms. Some relevant smartphone resources, which may be used in interaction, are also analyzed.

2.1. Robot Operating System (ROS)

The Robot Operating System (ROS[1]) is an operating system for robots that is widely used in robotics, with growing acceptance [1]. It provides libraries and tools to help developers create new robot applications. ROS frees the programmer from all low-level issues, namely hardware, libraries, message-passing and package management [2].

ROS is based on a graph architecture, where processes occur in nodes that can have multiple functions like receive and post messages, multiplex sensors, control hardware, preserve state, perform planning activities among others.

In ROS there are five main concepts:

- **Nodes**: Nodes are processes that perform computation. In a robot, multiple nodes can be running. For example, one node controls a laser range-finder and another node performs localization.
- **Master:** The ROS Master provides name registration and lookup to the rest of the nodes in a graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
- **Messages:** The data exchanged between nodes is called messages. A message is a well-defined data structure. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. It is possible to build more complex data types based on primitive types.

[1] http://wiki.ros.org/

- **Topics**: A topic is a place where information is exchanged. Messages are routed via a transport system with publish/subscribe semantics.
- **Services**: Client/server system in ROS.

2.1.1. ROS Publish/Subscriber

The publisher/subscriber[2] pattern is one of the simplest methods to exchange data between two, or more, nodes in ROS. It is necessary the existence of a common *topic*, or channel. Topics are a place where information is read and placed. Each topic is strongly typed by the ROS message type, used to publish, and nodes can only receive messages with a matching type. In topics, communications are unidirectional.

To exchange data, the publisher publishes a message in a given topic, and subscribers listen to new data updates (messages published) on this topic. In this way, multiple publishers and subscribers are allowed in a same topic. Subscribers will not establish message transport unless the types are consistent with the topic. Nodes that subscribe a topic will request connections from nodes that publish to that topic.

The master is used only to aid the entities establishing connection, allowing the registration and search topics by publisher and subscriber, respectively. The master node does not verify the type consistency of messages.

2.1.2. ROS Services

The publish/subscribe model is a very flexible communication paradigm, but it is not very suitable for request/reply interactions, commonly used in a distributed system. In ROS, request/reply patterns are known as *services*[3]. In this model, a server node provides services, and client nodes can perform requests, similar to client/server architecture.

ROS services server is defined by a string name, and a client uses this name to call the service. A client can make a persistent connection to a service, which enables higher performance at cost of less robustness to service provider changes. The client will have to implement a callback, to receive asynchronously replies. Services are defined by a file, where the input arguments and response type are specified.

---

[2] http://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers

[3] http://wiki.ros.org/roscpp/Overview/Services

2.1.3. ROS and Smartphones

ROS provides libraries for implementing robotic applications in mobile devices that allow accessing the robots execution environment, resources and internal information. Such solution exists already for smartphone devices using the Android OS.

ROS in Android consists of a set of libraries, here called *android_core*[4], to help write ROS applications for Android smartphones. It is possible to create new interaction methods favorable to both sides: the user that is familiarized with smartphones; and the robot, that perceives the smartphone language because both have the same execution environment (ROS environment). It is still possible to apply all the concepts mentioned above on a smartphone. Due to its agile connectivity tools (such Wi-Fi), the smartphone can be easily integrated into a ROS graph, where it can interact with other nodes at several levels (publisher and / or subscriber or client and / or server).

2.2. The RACE Project

The Robustness by Autonomous Competence (RACE) project is formed by a consortium of several universities, including the University of Aveiro. RACE develops methodologies for robots to learn through task execution, expanding their knowledge and adapting to new environments [3]. There are other factors and learning domains beyond task execution/learning.

Object recognition is an important factor for task execution. In this sense, RACE implements mechanisms for recognition and learning objects by the robot. This being one of the secondary objectives of this project.

RACE rests on a well-defined architecture (Figure 1) consisting of several modules, where each module has different functions. This architecture is implemented in the RACE robot in order to coordinate their resources (head, arms) and allow interaction with him. The PR2 robot is used (Figure 2).

[4] http://www.ros.org/wiki/android_core

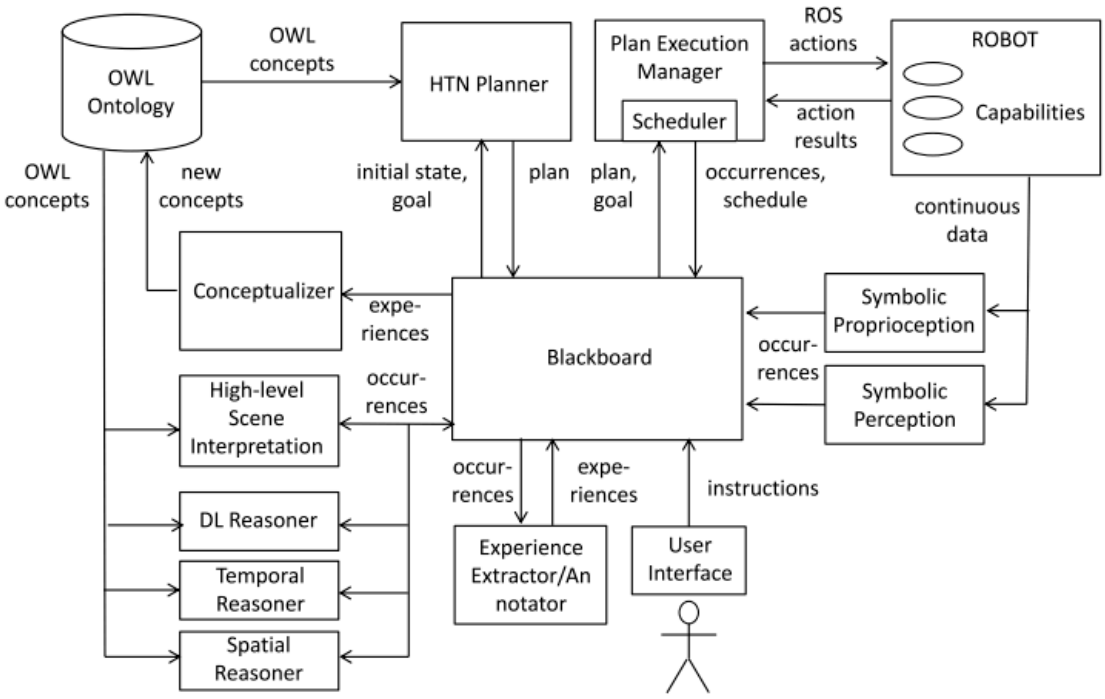


FIGURE 1: RACE ARCHITECTURE [3]

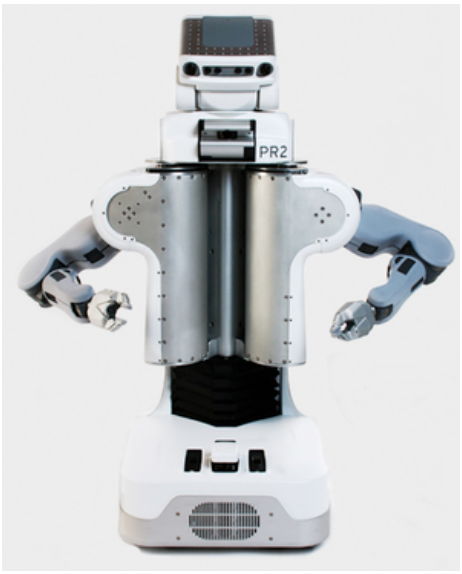


FIGURE 2: PR2 ROBOT

The central block of the RACE architecture is the Blackboard. This module provides a set of services, supported by ROS services, to access the internal robot contents, as well as sending commands/instructions. This is the main Blackboard characteristic for interaction. The Blackboard implements other functions, such tracking the evolution of both the internal state of the robot and the events observed in the environment, and handles information for other modules.

In the RACE architecture, all interactions with the robot are carried out via the *User Interface* module [4]. So far, this interface is implemented in command-line style, being inflexible, and does not provide interaction methods easy for most people.

RACE also proposes an internal data structure and well-defined services that are described in more detail in Appendix 8.1.

To demonstrate RACE, a restaurant scenario (Figure 3), was chosen by the project, where a robot acts as restaurant waiter [3].

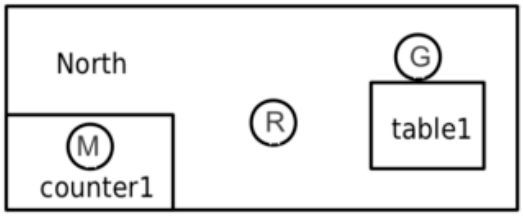


FIGURE 3: SAMPLE RACE SCENARIO PLAN FOR COUNTER1, TABLE1, ROBOT TRIXI (R), MUG1 (M) AND GUEST1 (G) [3]

2.3. Smartphone resources

Smartphones have become the devices of excellence for daily use by the users. The sell of smartphones grew since its introduction in the market. Currently, these devices can be compared to ultraportable computers [5].

Smartphones are equipped with a set of solutions that allow interaction with the user: microphone (allows sound recording [6]); inertial sensors (support gestures and activity recognition [7]); video/camera (allow acquisition of images/video [8]) and touchscreens that display visual information and also allow touch interaction.

This section is intended to address some of the smartphone resources that can be used to interact with robots. Some works that use these resources will be analyzed to get an idea of the different uses that they give.

2.3.1. Smartphone touchscreen

Interaction using touchscreen is the natural interaction way with smartphones. Touchscreens are an increasingly common feature in personal devices, especially smartphones and tablets, where there is an aggregation of several hardware components

(keyboard, mouse, keypad etc.) into a single interface. This aggregation arises primarily due to size and the need to adapt the user interface for interaction with fingers / hand [9].

In smartphone-based development, it is necessary to follow a set of guidelines to make an application minimally usable [10]. The SDKs provide a set of visual elements [5]liable to be used in an application (elements such as textboxes, buttons and menus).

The touchscreen is a resource that typically needs vision for interaction. However it can even be used by people without vision, or with deficient vision. With the use of some resources is possible to develop applications for blind people. Some studies have verify the viability of using the touchscreen for writing Braille [11]. This system is capable of being implemented in a smartphone, since it provides very positive results [11]. The touchscreen can be used for learning braille [12]. For this purpose, the interface buttons are used. The user only needs to know their location the first time using the application. Figure 4 shows an application example.

FIGURE 4: BRAILLETOUCH ON A SMARTPHONE [13]

Smartphone allows an alternative input method to the keyboard. Gestures, as used in smartphone touchscreen context, are fingertip movements on a touchscreen [14]. With gestures, it is possible to draw in a touchscreen, by swiping our fingertip on the touchscreen as if it were a whiteboard. Figure 5 shows an example, where the user is using gestures to find a contact in the phonebook. Current mobile operating systems already provide gesture recognition (e.g. drawing letters to introduce text, or screen change using a swipe) and allow adding new ones.

[5] http://developer.android.com/guide/topics/ui/index.html

FIGURE 5: ANDROID GESTURES[6]

Gestures support random access of a smartphone's content (applications, contacts) and functionality (search). The user no longer needs to manually search and navigate [14]. In application development gestures can be configured by the developer for different purposes. Android shows the gestures in yellow for recognized gestures and a lighter yellow for not recognized.

2.3.2. Smartphone inertial sensors

In smartphones there are typically two inertial sensors that can be used to detect movements and orientation[7], the accelerometer and gyroscope respectively. The *accelerometer* is especially used to detect motions and analyzes the device acceleration, in magnitude and direction. The *gyroscope* is used to measure changes in the smartphone orientation against an inertial reference.

The acceleration is the time rate of change of velocity with respect to magnitude and direction. There are changes in acceleration when there are variations in velocity of an object. The acceleration of an object can be positive or negative according to the movement direction. In smartphones, this acceleration is obtained by the accelerometer when the smartphone is moved [7].

To detect motion direction by smartphone there is a system composed of 3-axes. When a device is held in its default orientation (Figure 6), the X axis is horizontal and points to the right, the Y axis is vertical and points up, and the Z axis points toward the outside of the screen. The axes position remains unchanged whatever the smartphone position.

---

[6] http://www.google.com/mobile/gesture-search/

[7] http://developer.android.com/guide/topics/sensors/sensors_overview.html

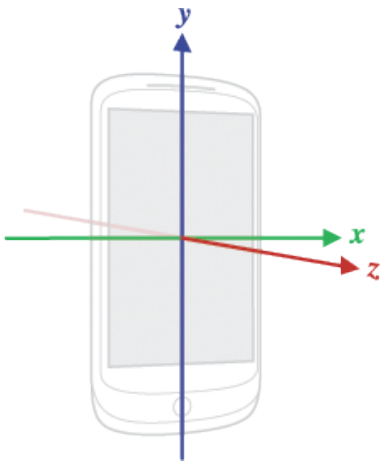


FIGURE 6: COORDINATE SYSTEM USED BY THE SENSOR API

The data from the accelerometers can be presented in the form of 3D vector (x, y, z) that represents the current acceleration of the controller in all three dimensions. Using mathematical formulas it is possible to calculate several movement parameters, and infer the smartphone's displacement [15].

Before smartphones others equipments with accelerometers have been produced, for motion recognition (some examples in Figure 7). A good example is the Wii controller that uses accelerometers and gyroscopes to support motion recognition to play video games. Currently, it is also being explored for other purposes. One example is in Human-Machine Interaction (HMI) [16]. In this work, the sensor data from Wii controller is sent, via Bluetooth, to a PC where it is analyzed. The main goal was to recognize gestures with a small number of training samples [16].

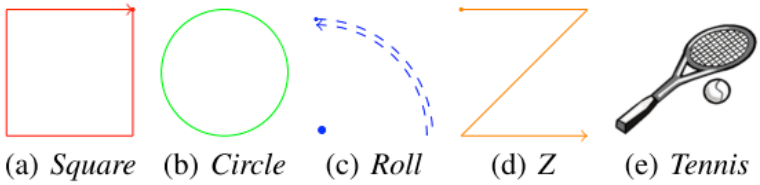


FIGURE 7: REFERENCE GESTURES [16]

With the appearance of the first mobile devices with motion sensors, some tests with these devices began to be made [17]. These devices, with low computation, send data to a computer where they are analyzed and processed. The authors compared two recognition patterns systems: support vector machines and Hidden Markov Models (HMMs). Both have high success rates, above 90 percent [17].

With next-generation smartphones, all processing can be done on the devices. No need to send the data to a computer. With the use of smartphone mobile characteristics, motion sensors began to be tested in different contexts. Some tests have been developed to show that it is possible to extract sensitive information from the user. Activities like walking, running, jumping and holding the phone can be distinguished using only motion sensors [7]. It is also possible to extract more sensitive user information, such as emotions [18].

2.3.3. Smartphone microphone

Mobile phones are equipped with microphones to perform voice communications. Today, these resources have been used for more noble purposes (like voice recording, sound recognition).

Speech Recognition (SR) algorithms allow the conversion of Speech to Text (STT) and Text to Speech (TTS) [19]. Is also possible to detect speech in several languages, with some success [20].

The latest smartphones incorporate services that allow interaction with them as if it were a personal assistant, like iPhone *Siri*[8]. It is possible to interact via voice and get the response through voice too. This interaction can arise in several contexts, such as question, or in an order, or simply a dialogue. These devices have an artificial intelligence component allowing the interaction to be more realistic.

The SR systems are becoming available for mobile developers[9]. The Google provides an STT interface that can be used for mobile applications. This interface has been used in some projects to convert speech to text [6], or text to speech. There are other SR platforms available for smartphones [21]. However, the Google SR is already integrated with Android SDK facilitating access to their services.

Speech recognition systems have been widely used in smartphones. These systems can be used in several different contexts: interaction with TVs (like change the channel, select the channel by gender)[22], an interface for sending messages through voice (converting speech to text)[23].

[8] http://www.apple.com/ios/siri/

[9] http://developer.android.com/reference/android/speech/SpeechRecognizer.html

This interaction method may have advantages over other interaction methods, such as touchscreen, especially if it is necessary to select multiple items to run a command that gets resolved with just a voice command, i.e. voice commands works as shortcuts [24].

2.3.4. Smartphone camera

Smartphones are equipped with cameras[10]. Initially these cameras were used to take pictures. With the telecommunications evolution, cameras began to be used for video conference. Modern mobile computing developers can use the camera for taking pictures, recording video, or use them in more complex applications, such as Augmented Reality (AR).

The camera has been exploited in other ways such as authentication using biometrics [25]. The idea is to perform implicit authentication, using for this purpose the ear of the user, during a call. Thus, when the user raises the smartphone to the ear to perform a call, it is verified they authenticity. Other authentication methods were explored also based in biometrics. Authentication using a finger is another authentication strategy [26].

The use of the camera for augmented reality is a very interesting technique. Typically, an AR application shows on the smartphone screen the content captured by the camera. Then, there is image recognition, in real time, and is drawn content over these recognized images. This content is also visible on the screen. One application example is the TranslatAR [27]. This application translates text that is captured by camera. In real-time the text to translate is overlaid by an image that contains the translated text.

The camera can also be used for other purposes. Color vision deficiency is a current problem that affects many people. The smartphone camera can be used to help these people [28]. This feature can be used to diagnose the type of color deficiency (red / green or blue / yellow) and help people analyze some colors that normally would not be recognizable. It is also possible to show to the people without visual deficiency how deficient people see the world. All this processing is done based on image processing to change the scene colors.

Due to the smartphone high processing capability and high resolution of both camera and touchscreen, smartphones have all that is necessary for image processing, namely methods for acquisition of 3D real objects [29] or object recognition [30].

The camera, coupled to a set of services, may be used in learning [31]. This technique allows children to rapidly gain access to a large repository of multimedia information. For

[10] http://developer.android.com/guide/topics/media/camera.html

this purpose, photos of specific codes attached to places or objects are taken. The photo is then sent to a service in order to obtain access to content specific to that location / object.

2.4. Interacting with robots using smartphones

In section 2.3, some smartphone resources that can be used in interaction are analyzed. Some of them may provide alternative interaction methods, with good results. The human–robot interaction, using smartphone, can be supported in different ways, using several smartphone resources. This section is intended to address some of the most used.

2.4.1. Interaction via touchscreen

The use of a smartphone touchscreen is one of the simplest and easy ways to interact. Here, the interaction can be done in two ways: interaction by text commands or visual elements, such as buttons [32]. This interaction method does not induce error. A user event is automatically converted into the language understood by the robot.

Touchscreen interactions technique is typically used in remote control. The smartphone works as a command device to control the robot. There are other interaction ways using the smartphone keypad to move the robot [33].

The touchscreen can be used for other purposes. Due to its display features, it is possible to show the robot vision, i.e., what the robot is seeing [34]. Thus, it is possible to control a robot, based on its vision, by swiping a finger in a direction that the robot should move. The main purpose is to guide the robot to an object to provide instructions involving this object. A scene can have multiple objects. The multi touch feature is used to select the desired object. The smartphone touchscreen brings very good results, compared to the use of other tools, such as a laser or Wii remote, to point to an objects [34].

2.4.2. Interaction via speech

Voice is another way to interact, more practical and interactive. It makes the environment more realistic, looking like interaction between two humans. This is the natural way for communication.

Interactions with robots already use SR mechanisms. To use this feature, the smartphone needs Internet connection. The use of Google's voice recognition engine can bring good results, for both men and women, older and younger [6].

In addition, we can address other interaction approaches. The propagation of sound may be another interaction method. The smartphone can act directly on the robot based on sounds that captures [32].

2.4.3. Interaction using inertial sensors

When verbal language is not possible, gestural language can be a viable alternative. Sign language is practiced using hands and fingers to perform movements and / or shapes.

As we saw in section 2.3.1, the inertial sensors can be used to detect movements made by the smartphone. Using this technique, it is possible to interact with robots using hand gestures. With the smartphone, we can only interact at hands level. It is not possible to make gestures more detailed, as with fingers, due to the device size.

These sensors also allow detecting the smartphone tilt. With this interaction, we can have alternative methods to move a robot based on the rotation and tilt of the device [35]. Here, the smartphone acts as remote control.

3. Smartphone-based interaction with a PR2

This chapter describes several interaction scenarios, mapping them with a set of smartphone-based interactions with a PR2 robot, using the RACE software architecture on the robot's side. It is intended to show how the smartphone resources were used for interaction, based on functionalities and services that the robot offers.

3.1. Architecture

In the global architecture (Figure 8), there are two principal entities: the smartphone and the PR2 robot. This architecture is based on robot characteristics. Here, the smartphone supports a user interface to access the robot information, and the robot provides services to support commands given by the user, via the interface.

The smartphone interacts with the PR2 using ROS services. The smartphone–robot connections are possible if both are in the same WLAN, since communication is done via Wi-Fi. With a ROS module in the smartphone, the smartphone is able to access the RACE Blackboard services, enabling to migrate the former command-line interface to an interface implemented on a mobile device.

This architecture supports multiple smartphone instances, i.e., it is possible the simultaneous existence of various smartphones connected to the robot. However, this can bring some constraints in task execution. The robot can only perform one task at time. While executing a task, it discards further instructions from any source.

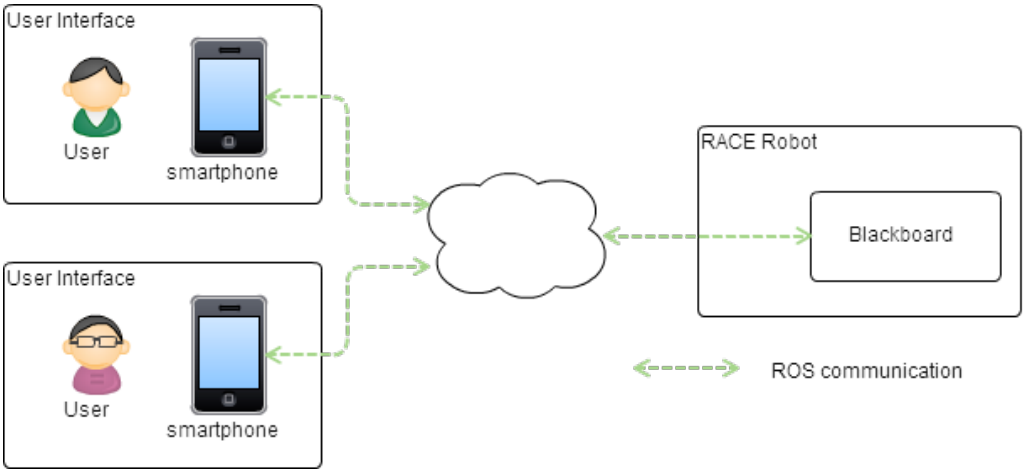


FIGURE 8: GENERAL ARCHITECTURE

3.2. The smartphone as main interaction unit

The smartphone is the central interaction unit between a human (e.g. operator, guest/client) and the robot (employee), in a restaurant context. As central interaction unit, several smartphone abilities are explored to build interaction solutions, in a set of selected scenarios, involving the PR2.

In this approach, the smartphone is explored to support the following interaction types:

- Interaction via touchscreen;
- Interaction using inertial sensors.

3.2.1. Interaction via touchscreen

The interaction using touchscreen can be in two different ways: menus (text commands) and symbol drawing (or gestures, in Android context).

Menu selection and text insertion

The explicit interaction with the robot is made by menu option selection (select tasks, and its attributes) and text insertion. Since the interaction with the robot must be made through selection of options, an interface that implements menus and text boxes is a good choice.

These text boxes, besides the text insertion, have extra functionalities in order to help the user when s/he selects the task, or enters text. The user does not know, nor is required to

know, which tasks the robot can execute. To circumvent this problem, some strategies are used, such as autocomplete, where the user remembers part of an instruction and the interface completes it, and option menus, where all options are presented in an alphabetical ordered list.

These dialogs must be dynamic, because each task is associated with a variable set of arguments. Therefore, the user must first insert the task and the application automatically displays as many text boxes as the number of arguments. In each text box, an instance (an object known by the robot) must be inserted. As for the tasks, the same insertion mechanisms can be used for instances (autocomplete and list of all options). These arguments are in the same order in which they are interpreted by the robot.

Symbol drawing

Symbol drawing on the touchscreen is another communication modality, to be used as an alternative to inserting text. Text insertion can become tiresome, especially for tasks that are constantly used. This alternative communication mechanism may facilitate task insertion. The symbols can be anything.

Before its use, it is necessary to make an association symbol–task, so that it can be recognized, and be converted to a task. Thus, when the user draws a symbol, the respective text boxes are automatically filled, and the user only needs to confirm. So, we avoid writing, which is often slow, and replace it with a quick insertion method.

3.2.2. Interaction using inertial sensors

Inertial sensors allow motion capture, especially some movements usually performed by the hand. These gestures represent, implicitly, commands that can be sent using the previous interaction method (explicit commands).

In the current implementation, there are two pre-programmed motion patterns:

- Shake sideways to catch the robot's attention;
- Shake back-and-forth to call the robot.

These gestures are typically used between people. To catch the robot's attention, it is necessary to shake the smartphone sideways, similarly to saying "Hello", with hand. In the smartphone coordinate system, this is a horizontal movement, along xx axis. This movement is called *hiCommand*. To call the robot, the movement is similar to calling a person, using

hand. In smartphone coordinate system, this is a movement according to the z-axis. This movement is called *comeCommand*.

The accelerometer values are used to measure the position, and the acceleration, of the device. The XYZ coordinates represent the position and orientation of the device when the acceleration occurred. So, it is measured in two consecutive points and the absolute value of their difference is calculated. This result is greater when movement intensity is higher. A prior check is performed to reduce noise. Note that small smartphone position variations need to be discarded.

The *hiCommand* is detected when there is a horizontal movement, according to the x-axis. This movement pattern is detected after the user makes three similar movements. The initial motion direction is not important, because the absolute value is used. The *comeCommand* is similar to *hiCommand*, just changing the axis. In this case, the *z* values are used. It is necessary to perform this movement twice.

The *comeCommand* can only be recognized if the *hiCommand* is recognized before. The *hiCommand* is not associated with a specific task to be executed by the robot. This command only serves to make the robot aware of the user. The *comeCommand* is associated with *drive_robot_Task*, which results in the robot moving towards the user.

The *drive_robot_Task* requires a location, from the locations that the robot knows. This attribute must be entered manually, by the user, before the movement. The use of user localization algorithms is not the focus of this thesis.

3.3. Use cases of interaction

The PR2, due to its similarities with humans (mobile robot endowed of upper limbs), can perform most tasks as humans, including act as restaurant employee to serve guests.

The following use cases for a robot like PR2 are considered (Figure 9), which were abstracted as high-level instructions.

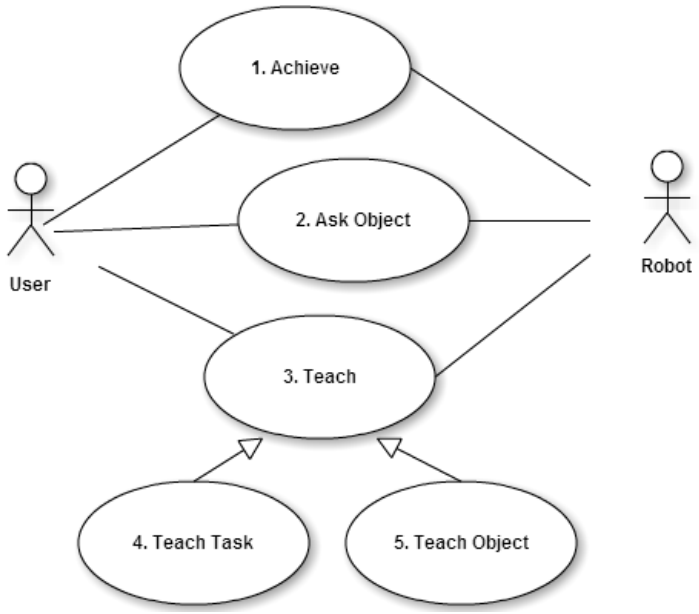


FIGURE 9: USE CASES DIAGRAM

The *Achieve* instructions allow commands submission, to the robot. These commands consist of tasks to be planned and executed. Using the menu interface, the user selects the task that the robot should execute, and the respective parameters. At each time, only one task can be selected, from all existing tasks. This use case has two main objectives: send commands to the robot to perform a task; support the step-by-step teaching of a complex task.

The *Ask Object* instruction allows asking the name of an object. For this purpose, the user uses the smartphone camera to capture an object, and then sends it to the robot. In response, it is expected that the robot replies with the category name of the object. To do this, the robot must implement algorithms for objects recognition. The goal of this use case is to check if an object has been well learned. If not, the object must be taught again.

Teach is an abstraction of two types of teaching instructions. The robot can be taught in several ways: to perform new tasks, or learn new objects. The starting point is always the user, which selects the option that s/he wants to teach.

The *Teach Task* instruction is a subtype of *Teach*. The purpose of this instruction is teaching new tasks. To teach a new task, the user sends several *Achieve* instructions, to attain the desired goal, and then sends the Teach Task, specifying the task name and respective parameters. The result of *Teach Task* can be checked if it was learned properly also using Achieve and observing the result.

The *Teach Object* instruction is another subtype of *Teach*. The goal of this use case is teach new categories of objects. It is necessary to use the smartphone camera for such purpose. A picture of an object is sent, coupled with the category name of the object. An

object learning service can then incorporate this new training instance to improve its object category knowledge and finally to improve object recognition.

The research around the PR2 and RACE is focused on learning. We will only use the PR2 abilities to support the interaction with the PR2 (e.g. send instructions to the PR2 and see if the PR2 acts accordingly).

3.3.1. Achieve

The *Achieve* instruction consists of sending a task to be executed. In RACE, each instruction has a well-defined structure. The instruction is defined by its type, which in this case is *InstructorAchieve*, and its name, e.g. *instructorAchieve34*, which ends with a unique numeric identifier for each occurrence [4]. This name is used by the Blackboard to distinguish experiences, or occurrences, in time. For each achieve instruction, the user interface will create two fluents (instances) in the Blackboard, namely an instance of *InstructorAchieve* and an instance of the selected task type. In the properties list, it is defined that this instance of *InstructorAchieve* instruction has a specific task:

[*hasTask, Task,* <*task_type*> + <*unique_identifier*>]

Then, it is necessary to identify the task content. This content is defined by a *fluent*, whose type is derived from the respective task type (e.g. *drive_robot_Task*), followed by a unique identifier. Each task can have multiple arguments. It is necessary to create a properties list, where each property specifies a specific argument. For example, the *drive_robot_Task* has one argument, the area, which is the location where the robot should move. Then, we would have a property similar to the following:

[*hasFirstArgument, Area, preManipulationAreaSouthTable1*]

Where the first element defines the argument position, the second element defines its type and the third element defines the instance used as value for the argument.

To be able to control the robot, the smartphone needs to know the tasks that can be performed. This is done by accessing the robot services to load this information. In the RACE project, such querying service is supported by the Blackboard. This information is then stored on the smartphone to avoid consecutive accesses, which may compromise the mobile

device battery. To update this information, it access the Blackboard services periodically, or when the user needs. This update is necessary to load new tasks that result from teaching.

3.3.2. Teach Task

Teach Task is intended to teach new tasks. The new task name is chosen by the instructor. However, it is only possible to teach based on contents already acquired. For example, the robot can move itself to a given position, and also knows how to pick up a coffee mug. However, the robot only knows how to perform these tasks independently. Thus, it can be instructed to perform a new task based on the combination of these two. Note that argument types and sequence are important in order to be correctly interpreted by Blackboard and other modules.

As in *Achieve*, a *Teach Task* instruction is also described by a fluent in the Blackboard. Such fluent will be an instance of *TeachActivityCategory,* with a name formed by "*teachActivityCategory*", followed by a unique identifier that identifies the specific occurrence [4]. A properties list, which describes this instruction, is created:

- [*hasConstrainedState, ConstrainedState, isNamed_ui*+<unique *identifier*>]
- [*hasRelationSubject, CoumpoundTask, CoumpoundTask_ui+<unique_identifier*>]
- [*hasRelationObjectName, xsd:string,* <*name_of_task_type*>]

This means that there is a relation (the constrained state) according to which a certain task (the relation subject) is named by the given name (the relation object name). For instance, when teaching how to serve a coffee, the relation object name could be "serve_a_coffee". Then, it is necessary to identify the task content. This content is defined by another fluent, of type *CoumpoundTask*, with name *CoumpoundTask_ui* + <*unique_identifier*>. This fluent is described by a properties list that establishes the task arguments, as specified by the instructor. For a task like "serve_a_coffee", the properties could be:

- [*hasFirstArgument, Table, table1*]
- [*hasSecondArgument,Guest, guest4*]

This way, by sending a sequence of suitable *Achieve* instructions, by which the robot serves coffee to *guest4* in *table1*, and a *Teach Task* instruction, the robot can learn that the performed activity is an instance of "serve_a_coffee" to *guest4* in *table1*.

3.3.3. Teach Object

The *Teach Object* instruction allows to teach the category of a real-world object. For this purpose, the smartphone camera is used to take a picture, followed by a selection to get the content that matters. Then, when the object is selected, it is associated with the category name of the object and sent to the robot. To make recognition possible, the robot is supposed have a service that receives this image and name and updates the category model. The category models then support object recognition algorithms.

While this work was carried out, the RACE architecture didn't include services for object recognition. However, a publisher was created that sends a message to a topic, with the object image and its name. For this purpose, a new ROS message type was created, with the following nomenclature:

Image image
string image_name

This message is published in */android_image* topic.

3.3.4. Ask Object

The *Ask Object* instruction uses the same approach as in *Teach Object*. The smartphone camera is used to capture an image of the target object, and then this image is sent, in an *Image* ROS object, to */ask_object* service. Note that only the image is sent.

If an object recognition algorithm is implemented, it is expected that the robot will reply with the object name, if available. When this information is received, the application asks the user if the name is correct. If not, *Teach Object* can be used to teach the object again.

4. Results

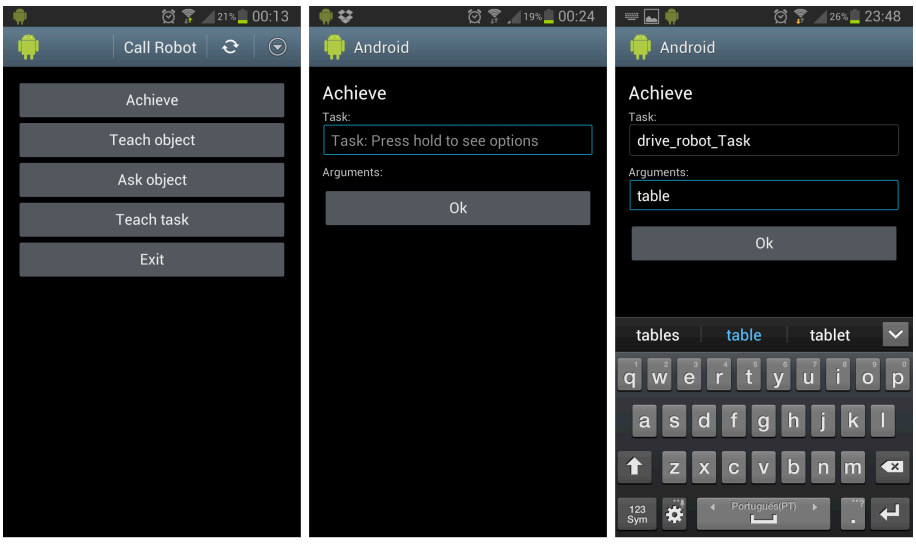
This section is intended to show and discuss the work carried out. It aims to simply and visually describe the work developed. This work consists of two major components: an interface for interaction, and a ROS client to communicate with the RACE Blackboard. The main objective was to develop interaction methods, using the smartphone resources, to communicate with robots in several ways.

4.1. Interaction using touchscreen

The interaction, using the smartphone's touchscreen is used in all use cases: *Achieve*; *Teach Object*; *Ask Object;* and *Teach Task*.

The most natural and easy way to interact with robots is via verbal communication. The current interface for communicating with the robot in RACE is a textual command-line interface. Other interaction methods must be converted into a message format perceptible by RACE.

Sending instructions is where almost all interaction rests. In application context, *Achieve* represents the transmission of commands. In *Achieve,* a task is associated with a variable number of arguments. As a result, it is necessary to have a dynamic interface. Figure 10 shows, visually, the implementation of this interface.



a) b) c)

FIGURE 10: ACHIEVE

Figure 10-a) shows the main interface, where all functionalities implemented are available. Figures 10-b) and 10-c) represent *Achieve*. Initially, it is necessary to insert the task (10-b) to be able to insert the arguments (10-c), since they depend on the task, in number and type. In each text box, only a particular data type can be inserted.

In order for the user to know the tasks that the robot can perform, it is necessary to access the robot Blackboard services to download this information. This information can be presented to the user in two different ways: autocomplete style (Figure 11-a) and a list of all options (Figure 11-b), to assist in their selection.

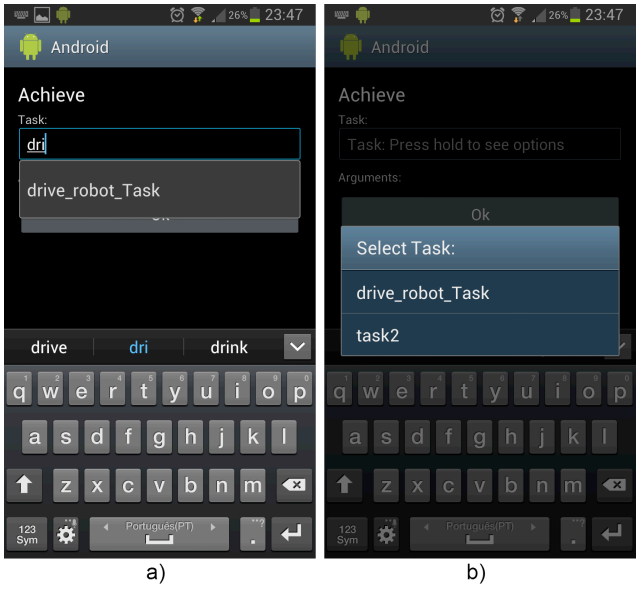
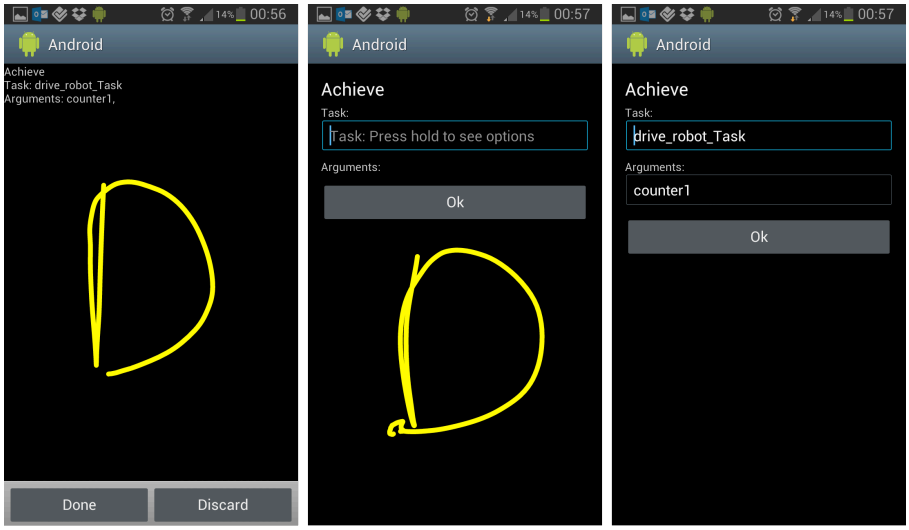


FIGURE 11: ACHIEVE - HELP

The text insertion is simple, but slow. For frequent tasks, the insertion of the same information can become tiresome. Therefore, it was created an alternative insertion method. This alternative method is based on drawing symbols (Figure 12).



a) b) c)

FIGURE 12: ACHIEVE – SYMBOL DRAWING

Each task is associated with one symbol (Figure 12-a). In *Achieve,* when we draw a symbol, anywhere on the screen (Figure 12-b), the fields are automatically filled (Figure 12-c), facilitating the text insertion. The user only has to confirm.

To teach a new task, interaction via screen is also used. The text insertion is used again (Figure 13). Here, there are no alternative insertion forms, since each case is different from others.

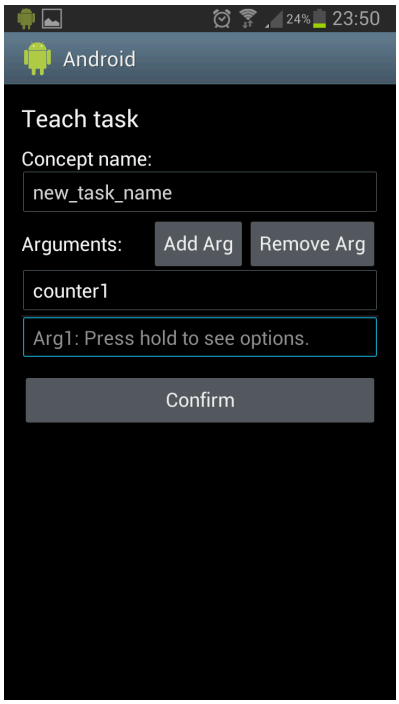


FIGURE 13: TEACH TASK

Note that the robot learns new tasks based on *Achieve* and especially *Teach Task* instructions [3].

In *Teach Task*, it is very difficult to add new alternative interaction. In the *Achieve,* this is possible. However it is necessary that they be preconfigured.

4.2. Interaction using sensors

The principle of using the keyboard, or symbol drawing, is the same: sending explicit instructions via text commands. With the use of inertial sensors, the interactions may become simpler and more attractive.

As we saw in section 2.3.1, the use of motion sensors has proven effective in interaction. For this mechanism to work properly, an association motion-task is necessary, in order to be possible to convert a motion into an appropriate RACE message.

A movement common to any scenario, including restaurant, is calling the waiter. The idea is to shake our hand to get the employee attention and then shake our hand again to call him. These examples have been exported to the application. Given various contexts in which these movements can be inserted, these two types of motions have been selected for interaction[11].

Figure 13 shows the impact of smartphone movements in a real robot. Initially a motion to call the robot is performed (Figure 14 - a) and then, the robot moves to a location chosen by the user (Figure 14 - b).

a) b)

FIGURE 14: CALL ROBOT (REAL ENVIRONMENT)

The instruction to move the robot to a given position is designated *drive_Robot_Task*. This task has a type of argument that corresponds to the destination. Therefore, before one can

[11] http://www.youtube.com/watch?v=BVGM995ZndI

perform the movement, it is necessary to indicate the target position (Figure 15). After performing the smartphone movement to call the robot, a message is automatically sent.

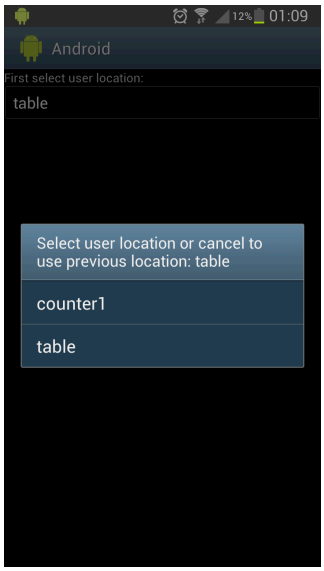
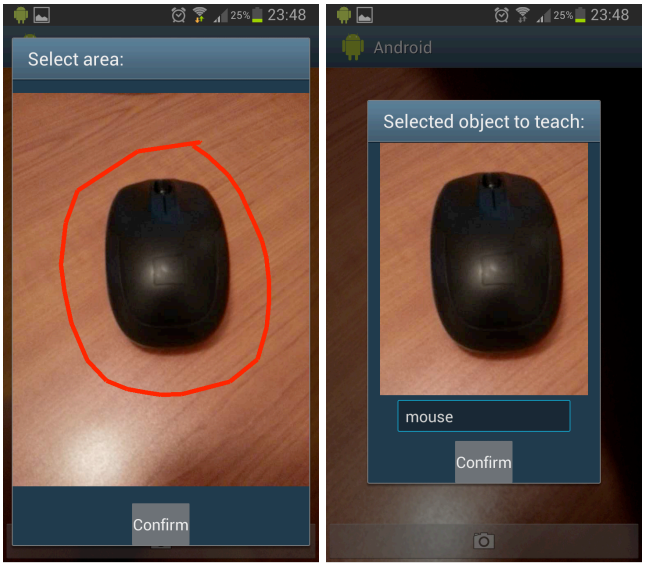


FIGURE 15: CALL ROBOT (LOCATION SELECTION)

4.3. *Interaction using camera*

In any context, it is necessary that the robot has knowledge of the world. It is necessary that robots can identify physical objects to easily adapt to different environments and execute tasks correctly (pick a mug, serve a coffee) [3].

In the interaction context, using smartphone, learning objects can be solved using the smartphone camera. With an image of an object, associated with its category name, it is possible to teach new objects, using the *Teach Object* instruction (Figure 16).



a) b)

FIGURE 16: TEACH OBJECT

The smartphone camera is used to capture a scene. Then, a selection is made to get the specific object (Figure 16-a), followed by a description entered by user (Figure 16-b).

Ask Object activity is similar to *Teach Object*. A picture is taken and the object is selected (Figure 16-a). With this, it is expected that the robot responds with the object name. If the name is wrong, the object is taught again.

5. Conclusions and future work

In this thesis, we present some interactions methods, using smartphone resources. Two interaction methods are used: through menus, text insertion and symbol drawing, using the touchscreen; and through inertial sensors, to detect motion patterns corresponding to gestures.

Menus are the easiest method to implement, since SDK provides contents for this purpose. However, it becomes monotonous. It is constantly necessary to select or write the commands. When we need to send multiple commands in a row, this mechanism can be slow. Drawing symbols is intended to reduce some of this monotony, facilitating the interaction.

The movements are an alternative to touchscreen-based interaction, but its implementation is much more difficult. There are always associated errors, and it is necessary to filter certain movements. However, the interaction is more intuitive and interactive (waving the hand to call robot is more interactive than typing the command, using a command line, or selecting in a menu). With this method, it is difficult to perform all possible tasks, since the commands are preprogrammed. To cover all tasks that can be performed, it would be necessary to use non intuitive movements. Not being able to cover all tasks, it is better to focus gestures common to most contexts, since different environments have different tasks.

ROS is quite flexible and greatly facilitates all communications and messaging. This middleware provides an abstraction level appropriate for implementation of Android applications. Throughout the development, and study of the state of the art, it was noted that ROS is not widely applied in smartphones. This might mean that smartphones and robotics have not been conjugated very often. The trend is communication between human-robot directly, i. e. face to face. However, the smartphone can be a viable tool when the communication cannot be directly established.

The interaction with RACE becomes somewhat limited when exported to a mobile device. The initial interface used to send instructions is a command line style. A mobile interface that implements menus is well suited. The same is not true for movements, because for each instruction is it necessary a previous association movement - task. Even for common movements, such as in movement to call the robot, there is not an interaction complete

without the need to choose options, such as calling the robot when it is necessary to indicate a location. This can decrease the interactivity level.

It is possible to run the application on all Android smartphones and tablets. However, this is not recommended. Tablets, due to the larger screen size, facilitate text insertion by the user. However, it makes the motion execution difficult. In small smartphones, the opposite happens. So it is advisable to use the application on a smartphone with screen size between four and six inches. The application was tested on Samsung Galaxy S3 with a 4.8 inch screen.

5.1. Future Work

Currently we just presented a proof of concept of using motions to control robots. More sophisticated solutions (e.g. pattern recognition techniques) for more complex motions could be explored, namely asking for the bill, or signaling when we are done with the meal. Allow the user to create their gestures, in order to perform custom gestures for specific tasks.

It would be interesting to add more interaction methods, like speech, to be able to make a more comprehensive study of interaction methods. In the state of art, we analyzed a system that is integrated in smartphones and provides services for speech recognition.

It would be interesting to take the restaurant scenario and develop the application accordingly. For example: include a restaurant plant in order to the user chose an internal location (table, counter), and the robot move to that location; include a restaurant menu (facilitates the order).

6. References

[1] S. Cousins, "Exponential Growth of ROS," *Robot. Autom. Mag. IEEE*, vol. 18, no. 1, pp. 19–20, 2011.

[2] S. Cousins, "Welcome to ros topics," *Robot. Autom. Mag. IEEE*, vol. 17, no. 1, pp. 13–14, 2010.

[3] S. Rockel, B. Neumann, and J. Zhang, "An Ontology-based Multi-level Robot Architecture for Learning from Experiences," in *AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI II*, 2013, no. Figure 1.

[4] A. Chauhan, L. S. Lopes, and A. M. Tomé, "Towards Supervised Acquisition of Robot Activity Experiences: an Ontology-based Approach," in *Artificial Intelligence – Local Proceedings: EPIA2013 – XVI Portuguese Conference on Artificial Intelligence*, 2013, pp. 228–239.

[5] M. Innovation and E. Workshop, "Terms of re-use Also by VisionMobile," *Developer Economics 2013*, pp. 1–61, 2013.

[6] D. Banisakher, T. Alexenko, M. Biondo, and M. Skubic, "Android Speech Interface to a Home Robot." 2012.

[7] S. Das, L. Green, B. Perez, M. Murphy, and A. Perring, "Detecting user activities using the accelerometer on Android smartphones," in *Team for Research in Ubiquitous Secure Technology REU Research Program*, 2010.

[8] S.-H. Shin, J.-Y. Yeo, S.-H. Ji, and G.-M. Jeong, "An analysis of vibration sensors for smartphone applications using camera," in *ICTC 2011*, 2011, pp. 772–773.

[9] T. Nakagawa and H. Uwano, "Usability differential in positions of software keyboard on smartphone," in *The 1st IEEE Global Conference on Consumer Electronics 2012*, 2012, pp. 304–308.

[10] N. Mi, L. a. Cavuoto, K. Benson, T. Smith-Jackson, and M. a. Nussbaum, "A heuristic checklist for an accessible smartphone interface design," *Univers. Access Inf. Soc.*, Oct. 2013.

[11] M. Romero, B. Frey, C. Southern, and G. Abowd, "BrailleTouch: designing a mobile eyes-free soft keyboard," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, 2011, pp. 707–709.

[12] L. R. Milne, C. L. Bennett, and R. E. Ladner, "VB G host : a Braille - Based Educational Smartphone Game for Children," in *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, 2013, p. 75.

[13] C. Southern, J. Clawson, B. Frey, G. Abowd, and M. Romero, "An evaluation of BrailleTouch," in *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services - MobileHCI'12*, 2012, p. 317.

[14] Y. Li, "Gesture Search: Random Access to Smartphone Content," *IEEE Pervasive Comput.*, vol. 11, no. 1, pp. 10–13, 2012.

[15] K. Seifert and O. Camacho, "Implementing positioning algorithms using accelerometers," *Free. Semicond.*, pp. 1–13, 2007.

[16] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a Wii controller," in *Proceedings of the 2nd international conference on Tangible and embedded interaction - TEI'08*, 2008, p. 11.

[17] Z. Prekopcsák, "Accelerometer based real-time gesture recognition," in *Proceedings of the 12th International Student Conference on Electrical Engineering*, 2008, pp. 1–5.

[18] H. Kim and Y. S. Choi, "Exploring emotional preference for smartphone applications," in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, 2012, pp. 245–249.

[19] A. a. M. Abushariah, T. S. Gunawan, O. O. Khalifa, and M. a. M. Abushariah, "English digits speech recognition system based on Hidden Markov Models," in *International Conference on Computer and Communication Engineering (ICCCE'10)*, 2010, no. May, pp. 1–5.

[20] H. Lin, J. Huang, F. Beaufays, B. Strope, and Y. Sung, "Recognition of multilingual speech in mobile applications," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4881–4884.

[21] A. Platform, M. P. F. Orlando, C. A. E. Andrea, and F. I. D. Marcela, "Tools evaluation for speech recognition based on domain ontologies over the," in *2012 IEEE Colombian Communications Conference (COLCOM)*, 2012, pp. 1–6.

[22] G. Németh and A. Viktoriusz, “Speech-enhanced interaction with TV,” in *Cognitive Infocommunications (CogInfoCom), 2011 2nd International Conference on*, 2011, pp. 1–5.

[23] S. Primorac and M. Russo, “Android application for sending SMS messages with speech recognition interface,” in *MIPRO, 2012 Proceedings of the 35th International Convention*, 2012, no. Dvm, pp. 1763–1767.

[24] S. Schaffer and M. Minge, “Error-prone Voice and Graphical User Interfaces in a Mobile Application 1 Introduction 2 Study,” in *Speech Communication; 10. ITG Symposium; Proceedings of*, 2012, pp. 1–4.

[25] P. N. Ali Fahmi, E. Kodirov, D.-J. Choi, G.-S. Lee, A. Mohd Fikri Azli, and S. Sayeed, “Implicit authentication based on ear shape biometrics using smartphone camera during a call,” in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012, pp. 2272–2276.

[26] C. Stein, C. Nickel, and C. Busch, “Fingerphoto recognition with smartphone cameras,” in *Biometrics Special Interest Group (BIOSIG)*, 2012, pp. 1–12.

[27] V. Fragoso, S. Gauglitz, S. Zamora, J. Kleban, and M. Turk, “TranslatAR: A mobile augmented reality translator,” in *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, 2011, pp. 497–502.

[28] S. Schmitt, S. Stein, F. Hampe, and D. Paulus, “Mobile services supporting color vision deficiency,” in *2012 13th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, 2012, pp. 1413–1420.

[29] J. H. Won, M. H. Lee, and I. K. Park, “Active 3D shape acquisition using smartphones,” in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 29–34.

[30] K. Jung, “Object recognition on mobile devices,” in *2012 IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, 2012, pp. 258–262.

[31] K. Mitchell and N. J. P. Race, “uLearn: Facilitating Ubiquitous Learning through Camera Equipped Mobile Phones,” in *IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE’05)*, 2005, pp. 274–281.

[32] R. Aroca, L. Gonçalves, and P. Oliveira, “Towards smarter robots with smartphones,” in *Robocontrol 2012*, 2012, pp. 1–6.

[33] H. Nasereddin and A. Abdelkarim, “Smartphone Control Robots Through Bluetooth,” *Int. J. Res. Rev. Appl. Sci.*, vol. 4, no. 4, pp. 399–404, 2010.

[34] P. Rouanet, P.-Y. Oudeyer, F. Danieau, and D. Filliat, “The Impact of Human–Robot Interfaces on the Learning of Visual Objects,” *IEEE Trans. Robot.*, vol. 29, no. 2, pp. 525–541, Apr. 2013.

[35] A. M. Walker and D. P. Miller, “Tele-operated robot control using attitude aware smartphones,” in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction - HRI’12*, 2012, p. 269.

7. Appendix

7.1. RACE data model

RACE has a well-defined data structure. The services for accessing the robot's world model are provided by the Blackboard. These services can be accessed through ROS command *rosservices list*. The message formats for each service can be accessed via *rosservice args /service_name*. This analysis is an important point at the beginning of development.

The RACE data model, which is used to represent the knowledge of the world, is represented in OWL2 DL, extended by SWRL rules to express constraints [3]**.** The domain concepts describe occurrences, which may happen as robot or guest activities.

Here is an example of an activity where the robot should move an object from one location to another (*MoveObjectFromTo* task):

Class: MoveObjectFromTo
EquivalentTo: RobotActivity
AND (hasObject EXACTLY 1 PhysicalEntity)
AND (hasFromOn EXACTLY 1 On)
AND (hasToOn EXACTLY 1 On)
AND (hasEmptyHand ATLEAST 1 EmptyHand)
AND (hasGetObjectFrom EXACTLY 1 GetObjectFrom)
AND (hasMoveObjectTo EXACTLY 1 MoveObjectTo)

These concepts represent activities that can be performed by the robot. As we see in the previous example, each task is associated with specific formatting and a particular set of arguments. These arguments have a specific order and type, and cannot be changed, in order to Blackboard can understand the task correctly.

The arguments differ from task to task, in number and type. Consider a simpler example. Suppose we have a task to be performed, *drive_robot_Task.* This task should move the robot to a given location. It makes sense that this task has one argument, which is the position to which the robot should move. This parameter is associated with the type Area, and can have

multiple instances as *preManipulationArea*, *Counter*, *Table*, etc. Each argument type have a set of instances.

To get the robot information, it is necessary access to a set of services. The information is distributed across multiple services. As mentioned earlier, there are tasks, and each task is associated with a set of arguments. Initially, an access to a load all available tasks is performed.

This service has the following message format:

string owl_class
---
string[] subclasses
BlackboardResponse result

The tasks available are stored in subclasses array. For each task, we need to know their properties, which have the following message format:

string owl_class

PropertyTypeDescription[] property_type_descriptions
BlackboardResponse result

The request parameter, *owl_class,* represents the task name. In response we have a *property_type_descriptions*, where all arguments information is stored, for a particular task.

For each argument type, we must know their instances. These instances are available through *blackboard/get_fluents_by_query* service. This service returns a list of *Fluents*, where each fluent has an attribute that represents an instance of a specific argument type. In the previous example, Table is an instance of type Area, which is an argument type of *drive_robot_Task*. It is necessary to access this service as many as the number of arguments types.

After accessing these services, the smartphone stores, in one organized structure, all the tasks that the robot can perform. To maintain the information organized in the application, two map structures are used. The first structure associates each argument type to a list of instances for this type. The second structure associated each task to the previous structure.

The service to submit information is the same for all these types, *blackboard/add_fluents*, and is defined by:

Fluent[] fluents

BlackboardResponse result