Algorithms and Graphic Interface Design to Control and Teach a Humanoid Robot Through Human Imitation

Marc Rosanes Siscart

Advisor: Dr. Guillem Alenyà
Supervision: Dr. Montserrat Ginés

Master Thesis

Institut de Robòtica i Informàtica Industrial
Universitat Politècnica de Catalunya
April, 2011

Abstract

Given that human-like robots are finding their place in different areas of our world, research is being carried out in order to improve human-robot interaction. Humanoid robots not only require a human appearance but also require human-like movements. Along these lines we present this project which tries to address the question of human to robot arm mapping with the objective to control in real time and to teach a robot by imitation.

For the technical implementation we have worked with a SR3000 ToF camera to sense the human movements which allows us to perform a markerless arm tracking based on depth information. The robot used is a Robotis Bioloid; still, the software has been designed to be adapted with few modifications to other robots having similar arm structures.

Aknowledgments

I would like to thank my advisor Guillem Alenyà for the supervision of this project and for giving me the opportunity to work at the Institut of Robotics (IRI) from the CSIC. I would also like to thank Montserrat Ginés to accept being the liaison with the university at ETSETB, and for her help in the bureaucratic tasks.

I would like to thank Pol Monsó, Sergi Foix and Sergi Hernandez for their help during the first steps of my project; Gerard Mauri for his help to record the videos of Bioloid; Eduard Nus for helping me with LaTex; Albane Delos for her advices; and Moha Marrawi and Edgar Simó for the talks and the time shared at IRI.

I would also like to thank the Backup group at T-Systems, with whom I have been working during five months that have coincided with some period of this project. It gave me some economic independence and I have spend a great time with the group.

Thanks to my flatmates, Naxo and Wilmar, for sharing the flat and the moods. Shared, life is more.

I especially thank all my family and particularly my parents, Leonor Siscart and Josep Ramon Rosanes, and my sister, Elisa Rosanes, for their love, economical support and understanding throughout all this time. Finally, I would like to thank Pau for the time shared during a large part of the project period.

Contents

List of Tables

List of Figures

1 INTRODUCTION

The present report is framed in the field of robotics and more generally, in the field of Human-Robot Interaction. Robotics is a multidisciplinary science related to other fields which mainly are: mechanics, electronics and software. This project will mostly focus on the software development part.

Humanoid robotics is one of the main research axis that are being carried out nowadays at the international level and currently, different research groups are exploring the possibility to control humanoid robots imitating the human movement. Humanoid robots able to communicate with humans using motions not only need to look like humans but they have also to behave like humans to allow a good communication without misunderstandings. Moreover, these robots must avoid movements that are not natural in humans if they have to act like humans. Arm gestures are some of the most important motions allowing communication between human beings. This non-verbal communication consisting on the body language receive the name of kinesics.

Following these lines, the goal of the present project has been to implement a system allowing the imitation of the human arms movements by using a humanoid robot. The system will work in real time as well as in deferred time, in such a way that it will be possible to store data about the movements and the robot will be able to "learn" different movements by imitation.

Thus, this project is part of a broader field known as Human-Robot Interaction (HRI). HRI has contributions from many other disciplines as human-computer interaction, artificial intelligence, robotics, natural language understanding and social sciences, amongst others [1]. Here, we can find the logical link between the Master of Science in Information and Communication Technologies and this project which has the goal to allow to program the robot arms movements by means of imitation and therefore will ease the information transfer between the human and the robot.

The report is organized following the next structure. First we begin by listing the main objectives of the project. After that we present the motivations describing which will be the main contributions of the project, how the project falls within the IRI research[1] and why it has been interesting from a personal point of view. Then, we introduce the state of the art on chapter 4, by describing the technological advances in the fields which are related with the project. On chapter 5 we describe the robot, the hardware and the software used for the implementation. Chapter 6 can be considered as the core chapter as it is here where we describe the implementation of the solution retained and we chose and analyze, through experiments, the main parameters used in our computer program. On chapter 7 we present the results and the experiments that have been done in order to test our computer program and to qualify the imitation. Finally, we list the conclusions at section 8.1 by describing which are the objectives that have been attained and to what degree. A chapter with possible ideas that can be implemented as a future work is provided at the end of this technical report.

[1]IRI: Institut de Robòtica i Informàtica Industrial. IRI is a Joint Research Center of the Technical University of Catalonia (UPC) and the Spanish Council for Scientific Research (CSIC). The present project has been carried out at IRI.

2 OBJECTIVES AND PLANNING

The aim of the project is to implement a software allowing the imitation of a human upper torso by a humanoid robot. This imitation will allow to control the robot arms via teleoperation and this control will be doing through imitation of the human actions. The software must allow to internalize the actions that have been imitated one time and it will allow to reproduce the same actions later on using the robot. Following what we said above, we present the main targets we have set with the project:

- The robot using this software must be able to imitate the human arms movement in real time as well as in deferred time. Thus, it will not only be possible to control the robot in real time, but the software will also allow the robot to learn new actions by imitation.
- The system designed has to be little obtrusive or totally unobtrusive, the ideal being to develop a markerless human tracking. For this reason, the input sensor used to sense the human movement will be a vision system.
- The system architecture must allow to extend, in the future, the software to other robots with similar arm structure. Thus, the software has to be modular enough in order to allow the integration of small modules of software, allowing to control different robot arms.

In order to accomplish the objectives previously described we present a short planning indicating the direction that will be followed to implement the project.

1. The first option is to use an existing human tracking system in order to focus our work in the mapping between the human and the robot joints. The mapping has to provide a mean to transform the human movements into similar robot movements by adapting the different number of joints between a human and the robot. We will study the viability to integrate a mapping functionality to one specific tracking software using it as departure point. This solution will be retained if the tracking given by this software fits our needs, providing reliable information in order to incorporate a mapping algorithm in it.
2. The second option is to develop a program from the beginning to the end. This program must include a human tracking system using the information from the input sensor and a mapping algorithm making possible to perform the imitation using the processed information coming from the tracking. In this case, a graphical user interface (GUI) has to be implemented. The GUI will create the link between the human, the input sensor information and the robot and must allow the user to check possible problems in the real time imitation. This second solution will be deployed in case of not retaining the first.

3 MOTIVATION

The motivations are presented from the more general to the more specific. In this way we begin by presenting some reflexions about the research in general, then we introduce the specific aspects that make interesting to develop this project and how it falls within the framework of IRI, and finally the personal motivations are presented.

3.1 General Motivations

Robotics is a discipline with a great future ahead, it is still a young science and it is little developed. Currently, Spain is not in a good economic situation due to the global crisis which is affecting us from 2008 to the present moment. Spain has always being one of the European countries investing less in the R&D sector. It allocated only a 1.38% of its GDP on 2009 when the EU average inversion was up to 1.9% [2]. Moreover, the government has announced a drop of a 20% in R&D inversion, in relation with 2009 [3]. It seems that it is too late to expect a quick recovery of economy and the society will have to address the consequences until the end of the crisis.

A healthy economy needs a healthy inversion in research, as it is one of the few ways to make a difference with other countries and to develop specific areas of the knowledge that later on can be exploited in the industry, helping to increase the exportations and nourishing the national market. Robotics along with biotechnologies, aerospace or renewable energies research are some of the sectors which nowadays can make the difference. Countries like Germany, which have some of the highest European inversions in R&D (2.6 of its GDP) is already emerging from the crisis and one of the main reasons is its R&D expansion.

In these times, new innovative applications providing unobtrusive tracking are emerging. This is for example the case of Kinect, the new XBOX 360 controller [4], which allows to play and interact with the video games without the need of contact and by means of gestures and movements. This controller uses a technology able to track 3D images from the scene with a camera, and the video games uses this information to interact with the user. At the beginning of this project, the Kinect technology was not still available and for this reason this project has used another type of camera.

The present project will use a Time of Flight (TOF) camera which also allows to obtain depth images. By processing the information of these kind of cameras it is possible to deploy an unobtrusive and markerless tracking and to use it to the control a humanoid robot. As usual, technology has to be simple to the user, and the complexity must stay on the developer side. Vision sensors allows a really simple interaction with computers and other digital devices and depth cameras are providing new means to develop new unobtrusive ways of tracking a person.

Providing robots with a software allowing their control by means of imitation offers a new way to define their movements and actions without the need of writing any code. This fact simplifies a lot the interaction with the robot and can help to open the domestic humanoid robot industry as it will be really simple to program robots by only showing them the actions that has to be performed. The robot programming will be less technical and more human. This project has also the objective to nourish the research in learning by imitation techniques, by exploring a specific way of mapping the human movements to the robot and drawing conclusions

about its reliability and effectiveness.

The capability to operate robot arms by imitation in real time can find and can develop existing applications in the robot teleoperation field. Using teleoperation, robots can be used in places that have difficult or impossible human access. This is the case of the submarine world which could help the archeology and the oceanography research; the study of the underground caves which could develop the speleology research in fields as different as geology or biology; or the study of the moon where we could apply the teleoperation with a delay of 2.52s which is the time that takes the electromagnetic waves to go and return from the moon. Apparently it is a field with a great future.

3.2 The project in the mark of IRI

The present project fits well into the real needs of the IRI research and the robotics community in general. Currently much research is being carried out with the objective to teach new actions to robots by means of demonstration and reinforcement[2]. The final objective of this research line would be to improve the HRI and the autonomy of the robots by allowing them not only to learn new actions by simple reproduction, but by inferring new methods to improve the actions and gain precision, taking as departure point some input movements that the robot acquire by demonstration.

Within the framework of IRI different projects allowing robot arms to learn new actions are being developed. Some examples are the project APREN which uses both, learning by demonstration and by reinforcement with different manipulators available at IRI, like the Stäubli robot arms; the National project PAU which goal is to provide a theoretical foundation of the relation between perception and action; and the European project GARNICS which aim is sensing the plant growth and building perceptual representations for learning the links to actions of a robot gardener.

The input movements can be given to the robot by different means. At this moment the input movements are taught to the robot by means of direct manipulation of the robot arms. Another strategy is to teach the robot by means of teleoperation using a delta robot with an haptic sensor and creating a mapping between the movements of the haptic sensor and the robot arms. Once some initial movements are taught to the robot, further algorithms of reinforcement learning allow the robot to improve the actions using some reward function related to the goal that must be attained. This project would explore the technique and the validity to teach new input movements to the robot by means of human imitation sensing the demonstrator with a vision system.

3.3 Personal Motivations

One of my main motivations doing this project was to gain some experience in a research institution. During my studies, I had the opportunity to make an enterprise internship, and I also had the opportunity to make a project in the university. I am really attired by the research world and for this reason I liked to try this experience. At this moment I think that doing research it is more difficult to find a place and a work stability after the PHD. On the other hand currently it begins to be difficult to find a job on enterprises and the new hired people tend to be students on internship contract. For all these reasons I have still some doubts about

[2] An example of learning by demonstration and reinforcement can be seen on: http://www.youtube.com/watch?v=qtqubguikMk

my professional future between the research and the enterprise world and I think that it has been interesting to diversify my trajectory during my studies.

When I was looking for a PFM project at IRI, I have been particularly attracted by this project because of its multidisciplinarity. At my point of view, current research can evolve a lot by simply building bridges between different disciplines and it is what is happening in a lot of current projects. Robotics is already itself an example of a multidisciplinary field of study which joints electronics, software design and mechanics. This project has the objective to map the human body movements to a humanoid robot, and this implies to know which are the structure and the movements achievable by a human body and how to link them to a robot. Thus, we can see that a link between robotics and biology appears when we face the subject of imitation. Moreover, imitation is a field closely related with learning, a capacity that has been exclusive of the animal kingdom, begins to be extrapolated to robotics. This is a subject that attract my attention, along with evolutionary computation and collective intelligence.

The previously listed interests bring me to another objective that I was looking for, with this project. It was to improve my knowledge in C/C++ programming languages. I already had some basis in these computer languages, but my knowledge was limited to the academical learning. Being C and C++ two of the most used programming languages nowadays, a good knowledge of them can open doors and it can allow me to learn more about these subjects of research. Moreover their possibilities are enormous and for the moment compilers are free, so and infinity of projects with really different aims can emerge with some creativity.

4 STATE OF THE ART

4.1 Introduction

In this section we present the state of the art on the subject of human movements imitation by humanoid robots. We can consider this subject as being part of the Human-Robot Interaction (HRI). HRI is a discipline which has the goal of achieve a friendlier interaction between robots and humans by means of providing robots with human capabilities. Some examples of those capabilities are facial and gesture recognition, natural language understanding, the ability to take decisions and to build dynamical models in function of the inputs, etc [1].

Human movements imitation by humanoid robots pursues many goals. On the one hand, friendly robots have to act as humans, thus it is important to design robots able to perform human movements and to avoid the movements that do not look human. For example, probably we would find strange a robot with human aspect able to bend the elbow joint towards both sides and we would react with empathy, disgust or another emotion from a wide range. As robots get closer to humans in appearance and way of acting, we begin to feel towards them the same emotions that we feel toward other humans [5]. Moreover, imitation and other forms of social learning are becoming a powerful means for robots to acquire new tasks and skills.

On the other hand we are looking for a more efficient way to program robot movements, as programming the movements using a programming language is really unintuitive. In fact, human movements are complex movements which are mostly goal oriented, especially in the case of children [6] which learns by means of imitation. Thus it is quite difficult to program these compound movements by setting the servos independently. One solution is to get information of the trajectories followed by the hands and feet to deduce the articulation angles by using inverse kinematics. With this technique we can deduce the servomotors angles with the information of the end effectors position and orientation that we want to achieve [7]. It also exists software allowing to program the movement of robots by directly manipulating the robot as being a puppet and saving the positions of the robot. After saving some positions, the robot is able to reproduce them, following the saved positions. One example of this kind of software is the Motion Editor for Bioloid robots from Robotis [8]. Even having all these techniques, the task of programming the robot movements is not straight forward. It becomes clear that being able to teach human movements to humanoid robots by simply observing a demonstration and allowing the robot to imitate it would simplify this task, providing human-like movements to the robot [9].

The real time imitation gives us the possibility to make our robot interact with the environment in the way desired by the human demonstrator. That is to say, acting in real time allows us to close visually the loop avoiding movements that could damage the robot such as self-collisions and guiding the robot to the desired positions. Moreover it allows to check the quality of the imitation, as we can compare the human subject with the robot imitation in real time.

The state of the art will first of all present the more general aspects of imitation and how imitation is strongly related to larger fields as is the case of learning. In fact, imitation is one of the most important animal capacities allowing learning [10] and robotics is taking advantage of that by creating robots able to learn by imitation [11]. We will present some humanoid robots used nowadays in the market as well as in the research world implicated on imitation tasks and we will give examples of its applications. Different techniques are used to extract information about the positions of his body (tracking using marks, markerless tracking, visual tracking techniques, exoskeleton tracking, inertial sensors tracking, haptic devices...); we will

present some of the tracking techniques and we will expose the techniques used to map the human positions to the robot positions which will allow to perform a human imitation.

4.2 Imitation and Learning

Many scientists think that there exist an important relation between biologic and robotic imitation systems. Breazeal and Scassella express on their own words [9]:

> It is our belief that research on these issues in artificial systems will both benefit from, and inform, research on imitation in biological systems.

Social learning is a discipline that has born in the animal kingdom and nowadays its being extrapolated to the artificial domain, with robotics and artificial intelligence. In fact, the biology research in social learning tends to be descriptive of the behaviors observed, while the robotic research community is going towards generative social learning models [9]. Thus, social learning in animals is a great source of inspiration for robots.

As exposed on [9], two fundamental problems appear when we want to address imitation. First, how does the robot know what to imitate from the whole human action? And second, how does the robot maps that perception onto its own action repertoire to replicate it? The first of these questions will be addressed on section 4.3 and the second question will be addressed on section 4.4. The precedent two questions are intimately related with learning using imitation, because the software used by the robot has to be able to give indications about what is important in the scene and what has to be tracked. After this perception, the robot must convert that perception into a sequence of its own motor responses to achieve a similar result. Nehaniv and Dautenhahn have termed this the correspondence problem [12].

Imitation and learning can be addressed at different levels. For example, animatronic devices, such those used in amusement parks, are able to replay with high fidelity movements previously recorded by manually putting the machine into a sequence of postures. Those machines are non-interactive and cannot adapt to new situations, so in that case there is imitation but not learning. Another kind of learning which is closer to our experience is what we call *learning by demonstration* which consist into a robot able to perceive human movements and to perform a task that has been shown previously by the human. Some explorations do not only focus on perceiving the movement of the human demonstrator, but they also focus on observing the effects of those movements on objects in the environment. Giving information about the goal to the robot allows it to improve its task by means of a trial and error strategy. Other projects tries to link in a deeper way biology and robots, by creating software abstractions of the brain regions implicated in the motor and imitation tasks. This is the case of Aude Billard and Maja J. Mataric project [11] which is based in learning human arm movements by imitation using a connectionist approach between biology and robotics. In their project, they create abstractions of the human brain and nervous system zones in order to perform the imitation of the human.

4.3 Tracking

To determine what is important and what is superfluous of an action is a challenging problem for perceptual systems, specially for vision systems. When a robot has to imitate an action it needs to focus on relevant information, for example, what is changing in the scene, or what are the objects with which the human interacts, etc. Thus, the perception module needs to direct the attention towards the important aspects of the scene. This problem is more pronounced

on visual systems but other motion capture technologies reducing the uncertainty about what to track exist. Some examples are the externally worn exoskeletons that measures joint angles, another solution consist into placing magnetic or colored markers on certain joints and tracking them. Other solutions of perceiving human movement through vision are based on techniques which try to combine sensory data with predictive models providing information on how a human is moving in a visual scene.

In their project Aude Billard and Maja J. Mataric [11] use two different methods in order to perform the tracking. The first one calculates the position of nine points situated at specific parts of the upper torso and the tracking its done off-line at a frame rate of 15Hz. The second method uses the FastTrack marker mechanism for recording the positions of four markers situated on the upper arm, near the elbow, the wrist and the hand. With the precedent tracking systems they obtain the Cartesian coordinates of the specific points. Those tracking systems do not provide information of the humeral rotation. After that, they apply a low pass filter in order to eliminate the fast small movements which are considered noise for their experiment. Finally they parameterize and segment the movements in terms of speed and direction.

Less complex approaches using only a hand tracking and fixing the shoulders position has been proposed. On [13] they use a skin color segmentation approach which allows them to find the hands and face. They select a hand and background sample and they use it as training data, and the hand position is estimated at every time using a Kalman filter. They give much importance to the View-Point Transformation (VPT) which says has received great attention in psychology but not in the field of visual imitation. The VPT consist into the capacity to transform an image from the allo-image, that is to say, the image that we obtain when observing a demonstrator, to the ego-image which is the image that would be obtained if the same gestures were actually performed by the person (or robot) which must to perform the imitation itself.

4.4 Mapping

In order to perform the mapping between the human movements and the robot movements two main different directions have been taken by researchers. One of them is to use motion primitives that are reproduced when the imitation software detects human movements which are interpreted as a movement corresponding to one of the motion primitives. For example we could use a primitive for a left step and another for a right step and we could reproduce these movements with a software able to pick out whether we are performing a step with the left leg or with the right leg. Another example of primitive would be to set a robot movement for waving. The primitive movements have to be programmed the first time, using inverse kinematics or any software allowing to program them. But after saving these movements, we can reproduce them as many times as we want, and making a vision software able to discern between many human actions we could make an imitation software for the robot that would use the previously stored motion primitives. One of the problems of this technique is that the reproduced actions can be quite different from the demonstrator actions. In order to have a good imitation we would need a huge database of motion primitives, and a good software which could discern between all these actions. Another problem of this technique would be the transition between one reproduced primitive and the next one. In fact, one single primitive could be really well performed, but the transition between two of them has to be processed, in order to obtain a smooth transition. The point for this technique is that it is really useful for specific movements that has to be tuned precisely and have to be performed on the same way most of time. For example, a primitive for walking straight would be a good solution to imitate a walking, as the equilibrium of the robot in the walk is delicate, in this way we could tune manually the walking movement of the robot. But if we wanted to imitate the way of walking for different actors with its specific characteristics,

we would need to create as many different primitives as actors are, so the primitive database would grow.

The other technique to perform the mapping is to use inverse kinematics based on some specific human points. As the human movements are many times goal oriented [6], the most typical way to map the human arms to the robot arms is to track the hands in order to find the servos angles for the robot arms and to track the feet in order to find the servos angles for the robot legs. This is sometimes not enough as the elbow position is also important in the case of the arms, and the knee position is important in the case of the legs. Anyway, when we want to perform a mapping, it is really important to take into account the morphology of the robot. In fact depending on the degrees of freedom of the robot it is not always possible to set the position of two points (e.g. elbow and hand). For example in the case of Bioloid where we only have three degrees of freedom (DOF) on the arm, if we set the position of the hand, most of times we have few options for the position of the elbow, while for a human, it is almost always possible to find an infinity of elbow positions for every hand position. Thus, for robots with few DOF the hand position is almost setting the elbow position. For other robots with more DOF the mapping can be more faithful to reality. Another example which manifest the importance of morphology is the walking. In this case the dynamics are very important and it is not only the morphology but also the distribution of weight and the inertia which plays a decisive role. It is for that reason that the imitation of a human walking by a robot cannot always be a faithful imitation of the actor if we want our robot to do not lose its equilibrium. In fact if different people have different ways of walking is also because of their body morphology. So a humanoid robot really similar to a human, even in their weight distribution, will have more possibilities to imitate a human, than a humanoid robot being still quite dissimilar to a human. Other limitations that have to be taken into account are the servo velocities and the position ranges. Every robot needs to be studied in detail in order to know which are the limitations which will influence in the imitation. Thus, the imitation has always to be a trade-off between the human movements to imitate and many parameters imposed by the robot own structure which can limit the imitation quality.

This last approach has been used in [13]. In their work, Manuel Cabido and José Santos-Victor use inverse kinematics in order to perform a real time imitation of a human, using a simulated robot. They create a module named SMM, standing for *Sensory Motor Map*, which is charged to perform the mapping using the demonstrator shoulders and hands coordinates. They also need to compare the real forearm and arm lengths with the lengths obtained in the images used to perform the hand tracking. Two kinds of mappings are studied in their work, for the first they take into account the elbow position (full-arm SMM), and for the second they only use the hand and the shoulder position (Free-Elbow SMM). When using the Free-Elbow SMM the only restriction imposed to the elbow angle is to try to obtain an angle as far as possible from the angle limits. They have also developed a method which makes possible to the system to learn by itself the SMM. Instead of giving to the system the mapping equations, some movements are performed and the simulated robot is trying to estimate the best angles to follow the demonstration.

4.5 Robots using Imitation and Applications

Many systems are used in order to explore robot imitation. Some are upper torsos, some are complete robots, others are expressive faces equipped with vision systems. Robotic arms are popular for exploring learning how to perform physical tasks by demonstration [9].

Simulations are also quite used to explore learning results but because of the inherent nature of the simulations, they often fail in the real world and thus, they have to be tested.

Shinishiro Nakaoka and its research group have used imitation in order to preserve traditional human dances that are currently disappearing [14]. This cultural asset can be recorded with a video camera, but it is more interesting to obtain 3D information in order to be able to reproduce the dance later on. With this purpose they have developed a software able to imitate human dances through robot primitives movements. Dances are usually composed of complex movements, making it difficult to preserve robot balance if imitation is performed directly using inverse kinematics. It is for that reason that initial motion legs primitives preserving the balance are used, defining a database of possible legs movements. This group has tested its software with the HRP-1S humanoid robot. The optically capture system used in their work is composed of eight cameras and the human dancer uses 30 markers strategically placed at their joints in order to track the dance movements. Once the tracking is performed they uses different techniques to imitate the arm and the leg movements. For the arms they mostly follow inverse kinematics based on the joint movements, and for the legs they mostly keep the leg primitives initially recorded allowing to preserve the balance.

A new vision sensor has been created on November 2010 and its software became open source. This sensor called Kinect, is able to extract the 3D information from the environment as it does the SR3000 camera and it has revolutionized the robotics teleoperation world in only few months. The price of this controller oscillates around 200 EUR; at least 10 times less expensive than the rest of ToF cameras in the market. This fact has opened the door to the freelance programmers that are now able to get all the necessary hardware for an affordable price. Some demonstrations of the results of these teleoperation and imitation softwares are already available on Youtube. Different robots have been used for the demonstrations, one of them being the NAO robot. The basic software of Kinect can be found as open source [15]. This open source code include the drivers of audio, video and depth sensors and also includes a real-time motion capture system able to capture a human by tracking his elbows, shoulders, hips, knees and head (Figure 1). This is the system that has been exploited by the community working on robotics in order to perform the mappings with different kind of robots. As it is a really new technology, there are not still scientific publications using Kinect, but some workshops have already been planned e.g. during the ICCV 2011 (International Conference on Computer Vision).

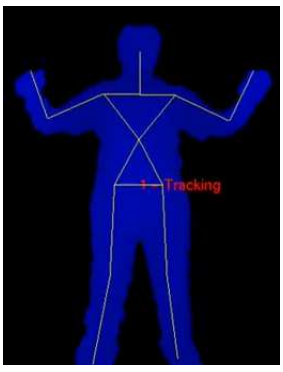

Figure 1: Kinect real-time motion capture system.

One of the developers that has used the Kinect in order to perform a mapping is Taylor Veltrop. He is an experienced humanoid robot builder which has used ROS (robot operating system) and the Kinect system in order to perform human imitations with his robot. He has

also used the robot NAO for this purpose and he posted some videos [3].

Another software pushing the imitation work to its limits its V-Sido [16]. This software is another real time control system for humanoids. A demonstration of his work can be seen on Youtube [4]. This software is not only able to map the arms but it also maps the legs movements by keeping at the same time the robot balance.

[3] http://www.youtube.com/watch?v=GdSfLyZl4N0

[4] http://www.youtube.com/watch?v=w8BmgtMKFbY&feature=related

5 PREVIOUS STUDY: DEVICES AND SOFTWARE USED

5.1 Introduction

In this section we present the software, hardware and all the devices that we have used to carry out the project. On Figure 2 we can observe the devices arrangement used for our experiments that we will describe below. The first device used as input sensor is a *Time-of-Flight camera* (TOF camera) [17]. These kind of cameras are able to provide 3D information of the scene and one of these cameras has been used in order to extract relevant features from the human body helping us in our objective to perform the mapping between the human and the robot.

A program able to exploit the 3D information coming from the TOF camera is *'VooDoo'* [18]. This software has been created at the Karlsruher Institut für Technologie and its able to perform a human tracking, returning a dynamic computer model of the human. We have studied the possible application of this program to our needs at section 6. The VooDoo information consist on different rotation matrices describing the relative position of the different body limbs. These limbs rotation matrices are organized on different XML files. We have used the XML parser from the Kranf website [19] in order to extract and be able to use them.

We have used C++ as programming language to develop our application. C++ is a compiled Object Oriented Programming language (OOP). This language allows to program at a level low enough to get a good speed of execution; allowing at the same time to visualize and conceptualize the code because the methods and attributes are organized in classes which represents objects.

The robot used has been *Bioloid* from Robotis. This robot is a 39cm humanoid robot, it is mostly designed by personal and educational usage. The microcontroller used by this robot is an *Atmel Atmega128*, but in our case we needed to send the angles directly from the computer to the robot without passing throuhg the Bioloid microcontroller. In their place we have used the *USB to Dynamixel adapter* which allows to directly communicate with the robot servomotors, from the PC.

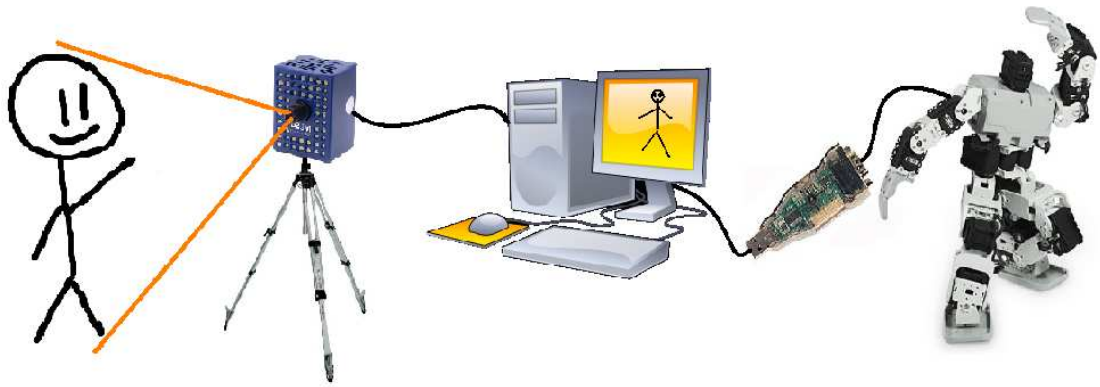

Figure 2: Assembly diagram representing the devices used and its arrangement

5.2 TOF Camera: Swiss Ranger 3000

The Time-of-Flight camera that we have used for this project is called Swiss Ranger 3000 (SR-3000, showed on Figure 3) from MESA Imaging [20]. MESA Imaging is a Swiss enterprise fully dedicated to TOF cameras design.

Figure 3: Swiss Ranger 3000 (SR-3000) Camera.

The ToF cameras from MESA imaging are able to obtain 3D images and videos by means of calculating the phase difference between an infrared light modulated by a RF wave which is irradiated from the camera to the scene, and the reflected wave [17]. On Figure 4 we present the process of sensing a point of a spherical object using the phase difference. Each pixel of the obtained image represents a phase difference and with this information, the time that takes the wave to go from the camera to the object and reflected back from the object to the camera can be deduced. With this Time-of-Flight information, the camera software calculates the distance traveled by the wave during this time and dividing this distance by two, the distance from the camera to the object is obtained. At the end the information consists on a matrix of distances from the camera to the scene. The software provided with the camera is able to do the transformation between the radial distances to the Cartesian distances and doing this the 'x', 'y' and 'z' coordinates from each point of the scene to the camera are obtained. Thus, the 3D images can be reconstructed.

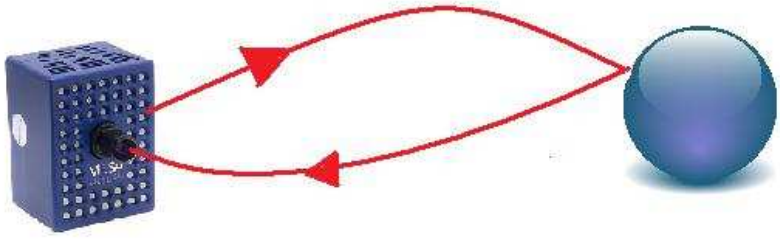

Figure 4: ToF phase difference diagram.

The usage of TOF Cameras as SR3000 has many advantages upon other kind of technologies as stereoscopic cameras or laser scanner systems. The basic scheme of these three different technologies is presented on Figure 5.

The three strengths of a ToF camera can be listed as follows [17]:

1. Registered dense depth and intensity images
2. Complete image acquisition at a high frame rate
3. Small, low weight and compact design
4. No mobile parts needed
5. Auto-illumination

Compared to the laser scanning technology where the scanning is done point by point, TOF cameras provides real time video frame rate measurements with no dependency on scanning cycles, as every image frame is composed by a matrix of distances.

Binocular stereoscopic vision systems have to implement complex calculation algorithms before to deliver 3D data. On the other hand, TOF cameras directly delivers depth values for each pixel. This results in faster, less computation intensive systems. Moreover, measurements are not dependent on the presence of contrast in the surface of measured objects because the scene illumination is provided by the camera and there is no need of color information to obtain the 3D information with TOF technology. Another advantage is that, contrarily to stereoscopic systems where it is necessary to increase the camera base length (distance between the two cameras forming the stereoscopic vision system) in order to measure larger objects, with a TOF system there is no need to perform this kind of calibration as the system is composed of a single camera.

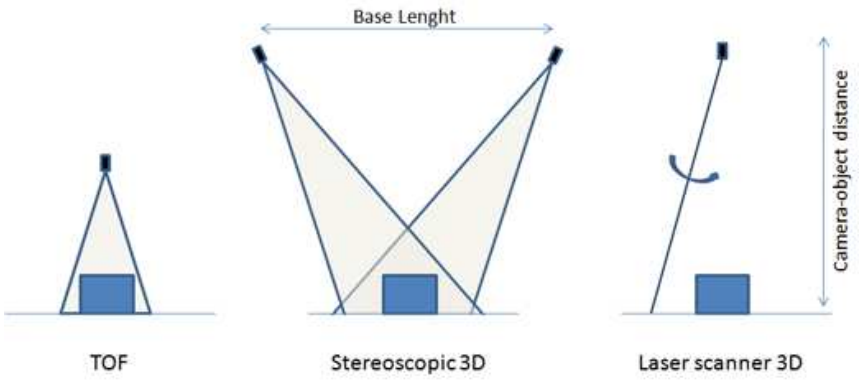

Figure 5: 3D vision technologies diagram.

On Table 1 the main features of the SR-3000 camera are displayed. SR-3000 has a housing size of 50 x 67 x 42.3 mm$\hat{3}$, thus it is little cumbersome and very portable. It has an average consumption of 12W and must be powered with 12V.

The images or frames obtained with this camera contains 176 pixels of length by 144 pixels of width, thus a total of 25344 pixels for image. For each of these pixels, the camera returns the three Cartesian coordinates of the shot point plus the intensity of the reflected wave returning to the camera for the same point. SR-3000 has a typical frame rate of 25 frames per second, allowing to obtain images of the scene in real time, giving a motion picture as a result.

In order to obtain the images, an infrared light of 850nm from an internal source of the camera is irradiated to the scene. This light is modulated at 20MHz, but this modulation can be set at other values to get images at different distances. Modulating at 20MHz, images up to 7.5 meters of depth without ambiguity in the distance measurements can be obtained. From table 1 we see that the theoretical distance resolution is about 1% of the distance measured. We will study the accuracy of this data in upcoming experiments.

SR-3000 Specifiactions:	
Pixel Array Size	176 x 144
Field of View	47.5 x 39.6 degrees
Illumination Power	1 Watt (average)
Wavelength	850nm
Housing Size	50 x 67 x 42.3 mm$\hat{3}$
Power Supply	12V
Power Consumption	12W
Output Data	'x','y','z' coordinates and 'i' (intensity)
Modulation Frequency	20MHz, default
Non-ambiguous range	7.5 meters
Distance Resolution	1% of range
Frame Rate	25 fps, typical

Table 1: SR-3000 Camera specifications.

The last parameter that is presented, but not the less important, corresponds to the integration time. The SR-3000 integration time varies from 0.2ms to 50ms. This time can be set to obtain more or less intensity from the scene. For objects that are sensed near to the camera lower integration times must be set in order to not saturate the image, otherwise there exist a threshold for which the information is lost and the images appear as being burned [1]. On the other hand, if we want to obtain images from farther objects, the integration time must be increased to get more reflected light from the scene. In fact, the wave intensity decreases in a manner inversely proportional to the square of the distance and the way to increase the reflected light is to increase the integration time.

As said previously, using a modulation frequency of 20MHz we can only measure depth up to 7.5m. That can be easily calculated using the relation:

$$c = \frac{e}{T} = e \cdot f \Rightarrow e = \frac{c}{f} = \frac{300000Km/s}{20MHz} = 15m$$

Where 'c' is the speed of light, 'T' is the wave period, 'f' its frequency and 'e' the distance traveled during a period. Thus, the distance traveled by the wave during a period is equal to 15 meters. We can only measure a distance of half the period without ambiguity, otherwise the same result in phase difference could be found for different distances. Because of this fact, a maximum distance of 7.5 meters of depth can be measured without ambiguity, using a modulation frequency of 20MHz.

In this case, the non ambiguity range is 7.5 meters most of time. However, some materials with specific reflection features, as metals and glasses can introduce extra phase differences on the wave, creating noise and distorting the 3D information and the depth images obtained with the camera [21].

Depth measurements with ToF cameras face the appearance of both systematic and non-systematic errors [17].

The systematic errors corresponds to the following errors:

• Depth distortion error depends on the measured depth distance and this error follows a sinusoidal shape. It is sometimes referred to as wiggling.

• Integration-time-related error: Integration-time-related error. Integration time can be selected by the user. It has been observed that for the same scene different integration times cause different depth values in the entire scene.

• Built-in pixel-related errors are related to the position of the pixel in the sensor array and are due to different material properties in CMOS-gates.

• Amplitude-related errors occur due to low or overexposed reflected amplitudes.

• Temperature-related errors happen because internal camera temperature affects depth processing, explaining why some cameras include an internal fan. The general strategy to palliate temperature depth errors is to switch on the camera and let it take a stable working temperature before using it.

Calibration to reduce the depth distortion, integration-time-related errors and amplitude-related errors have been addressed on section 6.5.1.

Four non-systematic errors can also be identified in depth measurements with ToF cameras:

• Signal-to-noise ratio distortion appears in scenes not uniformly illuminated. Low illuminated areas are more susceptible to noise than high illuminated ones.

• Multiple light reception errors appear due to the interference of multiple light reflections captured at each sensor's pixel.

• Light scattering effect arises due to multiple light reflexions between the camera lens and its sensor.

• Motion blurring, present when traditional cameras are used in dynamic environments, appears also with ToF cameras. This is due to the physical motion of the objects or the camera during the integration time used for sampling.

Some general challenges affecting ToF cameras have also to be mentioned. Short integration times contribute to obtain a strong noise ratio while high integration times can result in pixel saturation. Other concerns include ambient light noise, motion artifacts and high-reflectivity surfaces in the scene. Ambient light may contain unwanted light of the same wavelength as that of the ToF light source which may cause false measurements in the sensor.

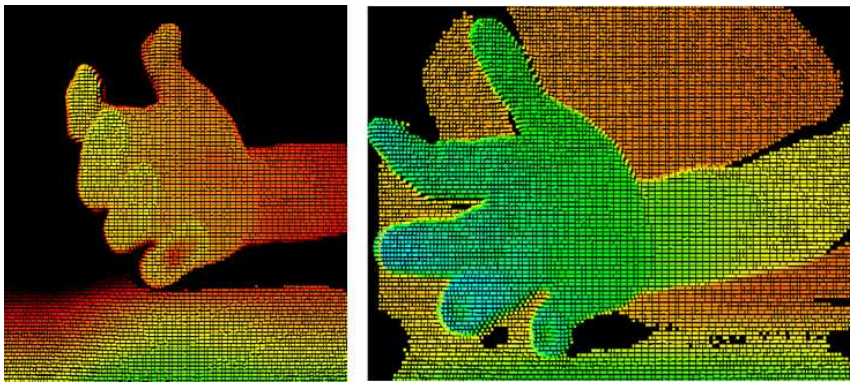

Figure 6: Images obtained with a Time of Flight camera.

Figure 6 show an examples of images obtained with the SR3000 camera. This images uses a gradient of color going from the blue for the closest objects, to the red for the farthest objects to the camera. On the figure we can observe a hand and an arm in perspective with this color gradient. At left we can see the image of the arm that is farther from the camera, and because of that it appears in reddish colors. At right, the hand is closer to the camera, and we can appreciate the fingertips and fingers in blue, because they are the closest elements; the hand and forearm in green, and the arm in yellow and orange. The background appears in orange and reddish colors. These images have been taken with the program SR3DView, which is the software that goes with the SR3000 camera.

Other ToF cameras are present in the market. This is for example the case of the SR4000 camera, which is the evolution of the SR3000, and has been also created by MESA Imaging; or the CamCube by PMD Technologies which is also used for research at IRI.

5.3 VooDoo Tracking Software

VooDoo is a tracking software created in the Karlsruher Institut für Technologie, designed to implement a full human body tracking by reproducing the human movements with a computer human model made of cylinders of different sizes, each cylinder representing a different human limb. The input information used to perform this tracking came directly from the SR-3000 camera. The VooDoo developers have always been working with the objective to integrate different sensors to their software (TOF, stereoscopic camera...), but at the time of this project, the input with TOF SR-3000 was the only fully operational. Different short sequences of tracking using the VooDoo software are available at the website on the footnote [5]. In these sequences we can appreciate the actors performing some movements and the computer body model reproducing them.

The VooDoo human model is composed by 10 different limbs, each of them represented with a cylinder: the head, the torso, 2 arms, 2 forearms and 2 limbs for each leg. The output information is organized on XML files. Each of them contains the rotation matrices needed to define the limbs orientation and position in relation to other limbs. A set of frames contains all the information of a movement sequence. On subsection 5.4 we explain how we have used an XML parser to exploit the VooDoo output information.

Figure 7 represents three different frames of one specific VooDoo sequence, where we can

[5] http://wwwiaim.ira.uka.de/users/loesch/har/

observe the imitation done by the computer VooDoo model of an actor which pulls back his right hand.

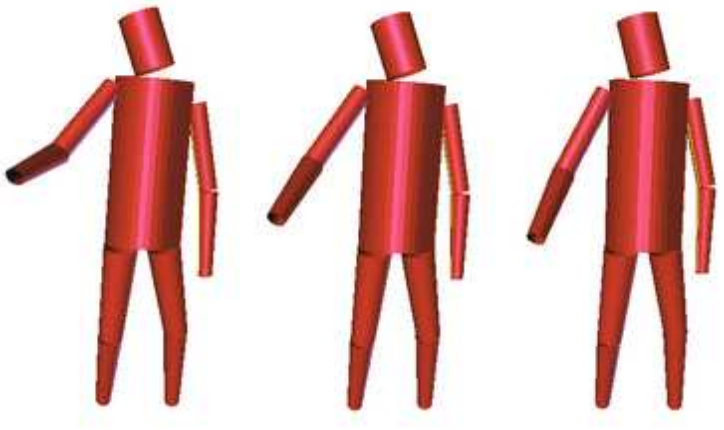

Figure 7: 3 different frames from a VooDoo computer model sequence.

In order to develop our imitation software, our first idea has been to study the possibilities of the VooDoo tracking software. We wanted to map the movements obtained with the VooDoo model, to servomotors angles ready to be reproduced with a robot of three degrees of freedom (DOF) on each arm. In order to do that we needed a perfect quality of tracking, because the movements of the robot would depend on the processed VooDoo information. Figure 8 shows this organizational diagram.

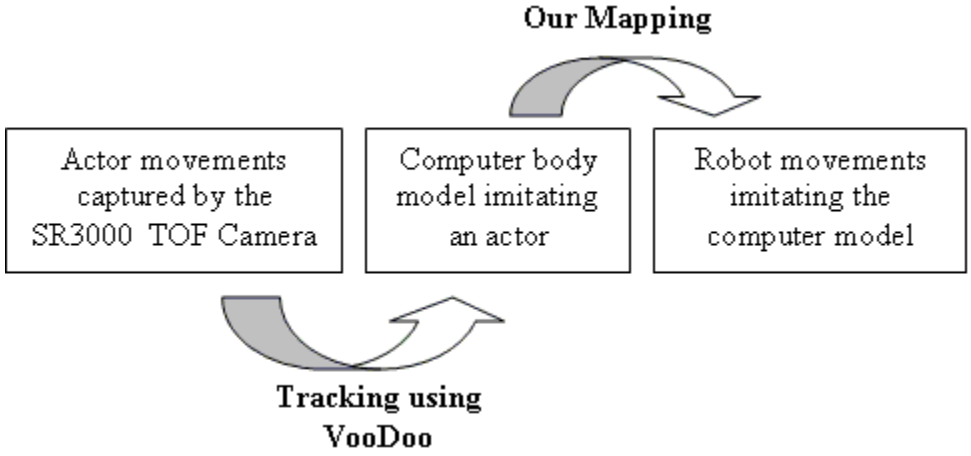

Figure 8: First organizational diagram.

After testing the VooDoo software in order to check the tracking quality, we have seen that it was little adapted to our needs. On one hand we observed that the tracking was lost many times for some limbs under our laboratory conditions. On the other hand because VooDoo performed a full body tracking. Working with an arm tracking rather with a full body tracking would allow us to safe computer resources and this is very important when working with real time applications.

Thus, we have decided to implement our own tracking system based on the depth information given by the SR-3000 camera, being specifically designed to be adapted to our needs. The tracking information obtained with the developed software will be exploited afterwards in order to map the human arm movements to the robot. Thus, we will develop a new software integrating at the same time a tracking and a mapping, with the ultimate goal of performing the human imitation using a humanoid robot.

Moreover, we will integrate the capability to reproduce VooDoo sequences in deferred to our software. This feature will allow the robot to learn movements through deferred imitation and working in deferred time will allow us to check the tracking quality of the VooDoo sequence before to generate a new imitation of the VooDoo sequence using the humanoid robot. The organizational diagram retained is presented on Figure 9 and two of the main parts of our project: the real time imitation using our own tracking and mapping software, and the deferred time imitation using a pre-recorded VooDoo sequences as departure point. We will detail these new organization diagram on chapter 6.

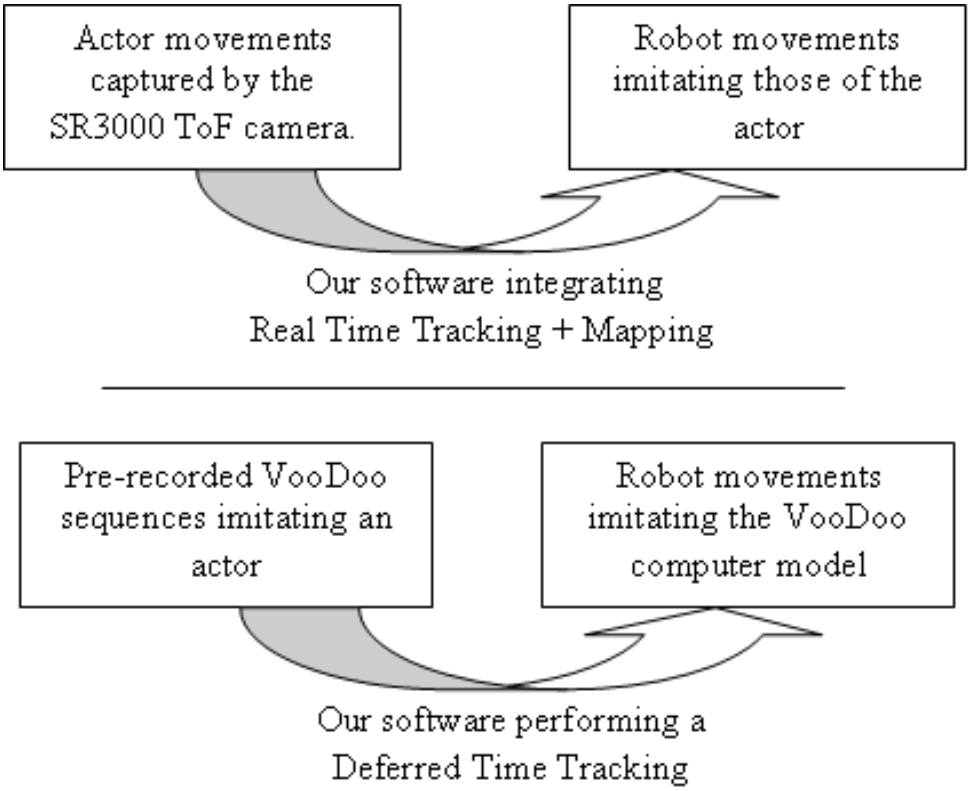

Figure 9: Retained organizational diagram.

5.4 XML parser

In order to work with the output information from the VooDoo tracking software we have used an efficient XML parser allowing to obtain the elements of the XML VooDoo output files concentrating the tracking information in form of rotation matrices. The library used has been created by Dr. Frank Vanden Berghen [19] and it has been designed to be compact, simple and portable. Its total size is 104KB and it is composed of only two C++ files: xmlParser.h and xmlParser.cpp.

This XML parser loads an XML file in memory and it generates a tree structure representing this file. Thereafter, it is easy to explore the tree and get the desired data from the XML.

Next, we show an example of how it works this XML parser, by obtaining some specific information from one of the VooDoo generated XML files that we will name: exampleXML.xml. This XML file, represented of Figure 10, has all the information needed to define the VooDoo human computer model position at a given time.

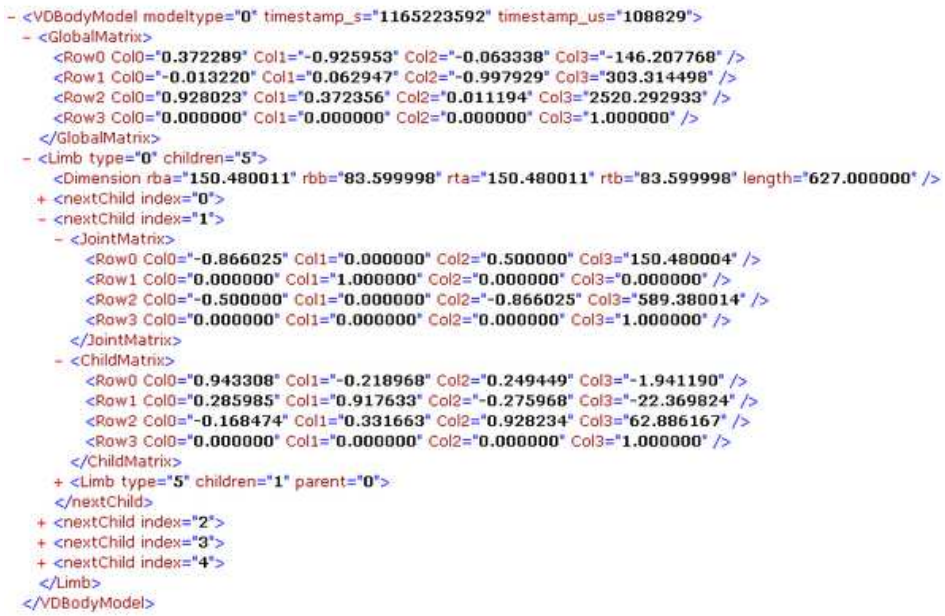

Figure 10: VooDoo XML output file representing one specific body position.

The first thing to do is to load in memory the complete XML file. When doing so, the XML charged in memory is organized in a tree structure that will allow us to extract its data later on. To load the file we use the following function:

```
XMLNode bodyNode= XMLNode::openFileHelper("exampleXML.xml","VDBodyModel");
```

With the precedent command we charge the main node of the XML file `exampleXML.xml`. We observe on Figure 10 that this main node is called `VDBodyModel`. We put all this information in an XMLNode object that we have called `bodyNode`.

Now we are able to extract the information by simply using two functions: `getChildNode()` and `getAttribute()`. The function `getChildNode()` allow us to move throughout the XML nodes tree, allowing to keep any desired node, and the function `getAttribute()` allow to extract the values and information stored on the kept node.

Next we show how we can extract the timestamp indicating the time reference in seconds at which has been generated the XML, which corresponds to the time reference at which the computer VooDoo model, imitating the human actor, was in a given pose:

```
long time= bodyNode.getAttribute("timestamp_s");
```

In this case, executing the precedent function, will return a long named `time` equal to: 11665223592, which is the value extracted from the attribute `timestamp_s` from the XML.

Finally, we will show how we can extract one of the shoulder rotation matrices corresponding to the relative position between the arm and the torso of the VooDoo body model. The rotation matrix corresponds to the sub-matrix of 3x3 components. In this case are looking for the `ChildMatrix` of the arm, which represents the arm position in relation to a reference arm position. We can see on the XML figure that this matrix is situated inside the child node called `Limb`. This child node, in turn, is composed by many limbs called `nextChild`. We are interested in extracting the information from the `nextChild` number 1. With all that we can already load in memory the `ChildMatrix` node, that we were looking for. We will place it in a XMLNode object called `leftShoulder` as follows:

```
XMLNode leftShoulder=bodyNode.getChildNode("Limb").getChildNode("nextChild",1).getChildNode("ChildMatrix")
```

The last thing that we have to do is to get the values of the matrix. We do so with two consecutive `for` loops and using again the function `getChildNode()` to get the matrix rows one by one. After that we use the function `getAttributValue()` to get the first three independent values of each row:

```
for (int i=0; i<2; i++)
{
    for (int j=0; j<2; j++)
    {
        shoulder[i][j]=leftShoulder.getChildNode(i).getAttributeValue(j);
    }
}
```

We have extracted the sub-matrix from the XML arm `ChildMatrix` and placed its nine values on a c++ object of doubles named `shoulder` representing the 3D rotation matrix:

$$shoulder = \begin{pmatrix} 0.943308 & -0.218968 & 0.249449 \\ 0.285985 & 0.917633 & -0.275968 \\ -0.168474 & 0.331663 & 0.928234 \end{pmatrix}$$

Once here, we can already process this information as desired since we have load the matrix values on our program.

With this simple process we are able to extract all the necessary data from the generated VooDoo XML files in order to process this information and obtain, with a mapping algorithm, the servomotors angles that we have to pass to the Bioloid robot to perform the human imitation.

5.5 BIOLOID Robot

The robot Bioloid from Robotis Company has been the robot used. The Bioloid kit allows to form different robot configurations, but in our case, as we want to work with human imitation, we have used a humanoid configuration. This configuration weights 1.7Kg and its height is about 39cm. The humanoid uses three AX-12 servomotors in each arm and six AX-12 servomotors in each leg. On Figure 11 the servomotors distribution with the main Bioloid limbs are displayed.

Its head is composed by the module AX-S1 which includes an infra-red sensor, an internal microphone , a temperature sensor and allows IR remote transmission and reception data. Its processing unit is a microcontroller is an Atmega128, but in this project we have not neither

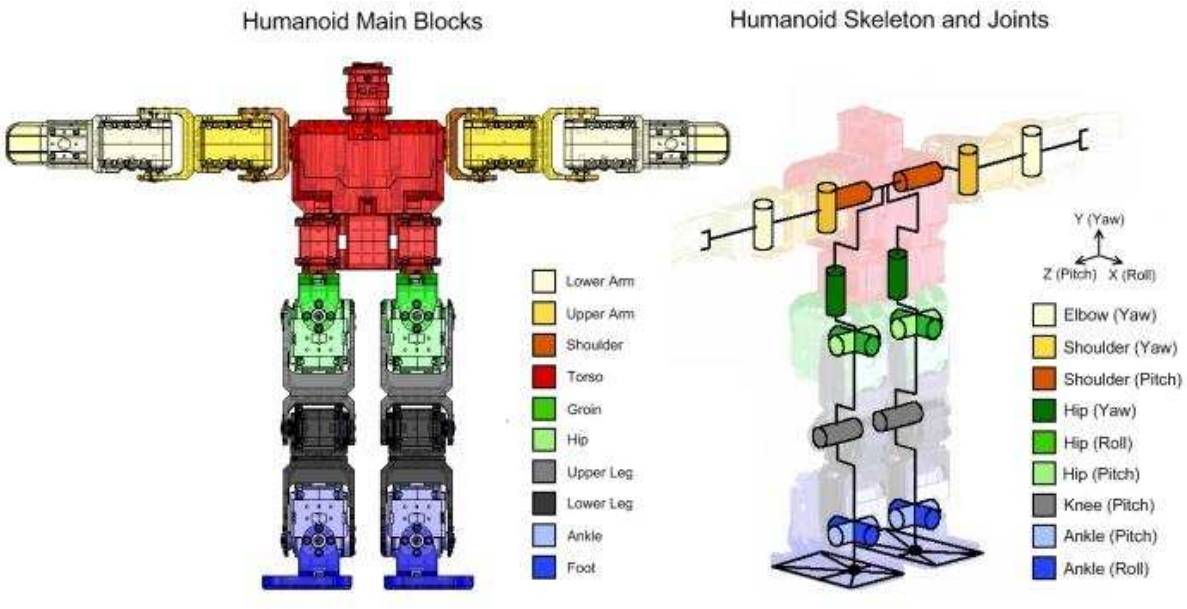

Figure 11: Bioloid robot with its main limbs.

the robot microcontroller, nor the AX-S1 sensors, as we have used the SR3000 Camera as main input sensor.

The servomotors involved in this project has been the three from each arm. They have been controlled directly from the PC after information processing on the same computer, this allowed us a faster processing capability as we have worked with a PC with a 2GB RAM memory.

The AX-12 servomotors allow programmable serial communication going from 2400 bps up to 1Mbps and they provide position, speed, load, voltage and temperature feedback. In these project we communicate with the AX-12 by reading and setting its position and speed from the PC. Each servo has a unique electronic ID (Table 2) used to identify the servo to which are directed the indications such as the position setting, etc. The AX-12 allow a rotation movement of 300 degrees in 1024 increments and a speed control also in 1024 increments. Moreover they shutdown automatically when the voltage, load or temperature are too high.

AX-12 ID	**AX-12 Position**
1	First AX-12 in the right arm shoulder
3	Second AX-12 in the right arm shoulder
5	Right arm elbow
2	First AX-12 in the left arm shoulder
4	Second AX-12 in the left arm shoulder
6	Left arm elbow

Table 2: AX-12 IDs and its relative position in the Bioloid arms

Next, we present the Bioloid arm configuration which uses three AX-12 servomotors for each arm to achieve different hand positions. Two of the AX-12 form the robot shoulder joint and the other forms the robot elbow joint. The diagram presented on Figure 12 shows the three different degrees of freedom (DOF) from the left arm. The first DOF of the chain is given by

the AX-12 with ID 1 for the right arm and ID 2 for the left arm. It allows one rotation of the arm around the axis passing through both shoulders (shoulder yaw: in red). The second DOF is possible thanks to the AX-12 with ID 3 in the right arm, and the servomotor with ID 4 in the left arm. They are the second AX-12 of the arm chain and allow the arm pitch (in green). Finally, the AX-12 with IDs 5 and 6 creates the elbow joint for each arm (in blue).

Figure 12: Robot arm configuration diagram.

On Figures 13 to 16 we represent module by module the left arm of the robot with its different joints and the AX-12 servomotors allowing the creation of these joints.

Figure 13 represents the AX-12 with ID 2. This AX-12 is the first of the left arm chain, it is fixed to the Bioloid chest and will allow the first shoulder yaw.

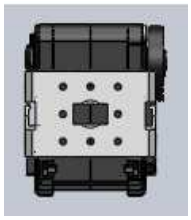

Figure 13: AX-12 with ID:2 (fixed in the Bioloid chest)

On Figure 14 we represent the AX-12 with ID2 with the shoulder frame. Here we can observe this first rotation created between the chest and the shoulder.

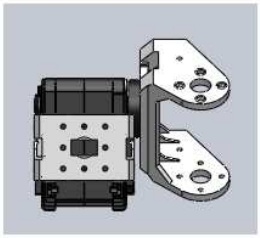

Figure 14: Left Shoulder. Joint with the chest (shoulder yaw angle).

On Figure 15 we represent the AX-12 with ID 2, the shoulder frame and the AX-12 with ID 4. This last AX-12 is the second from the left arm chain and it allows the arm pitch movement. This second rotation is created between the shoulder frame and the arm.

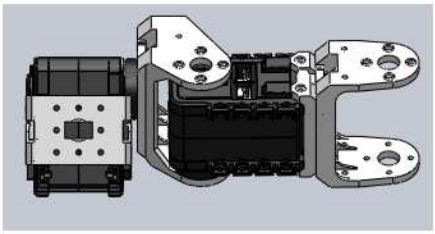

Figure 15: Left Shoulder+Arm. AX-12s with IDs: 2 and 4 (yaw+pitch angles).

On Figure 16 we represent the Bioloid left arm in its totality. The left elbow joint is created thanks to the AX-12 with ID 6, and allows one DOF between the arm and the forearm.

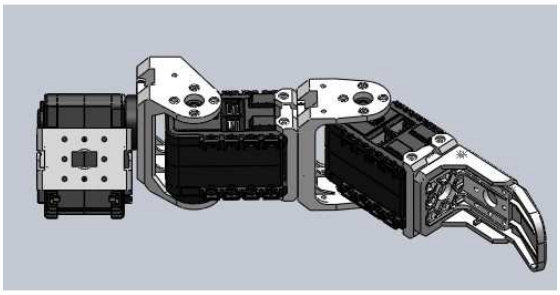

Figure 16: Shoulder+ Arm + Forearm. Total Left Arm: AX-12 with IDs: 2, 4 and 6.

This arm structure is very flexible and it can generally achieve any position within the working space. Three main constraints are present. The first is that the Bioloid hand has not any joints, so specific hand orientations cannot be set. The second one is that AX-12 servomotors are limited to a 300^{o} rotation and there exists collisions with the Bioloid frame itself, which reduces the range of possible positions. The third constraint is that the human arm has at least two more possible degrees of freedom; they are the roll arm movement and the roll forearm movement. This two additional movements together with the three previously presented degrees of freedom would allow a wider range of movements; for example, it would be possible to adopt different elbow positions while keeping the same hand position.

The Bioloid Kit, integrates the 'Motion Editor' software. This program allows to set different Bioloid positions and to create different motion sequences. After setting a precise Bioloid position, the Motion Editor software allows to check the position values of each AX-12 servomotor (red zone on 17); we have used this feature to determine the servomotor movement ranges expressed in the Bioloid nomenclature.

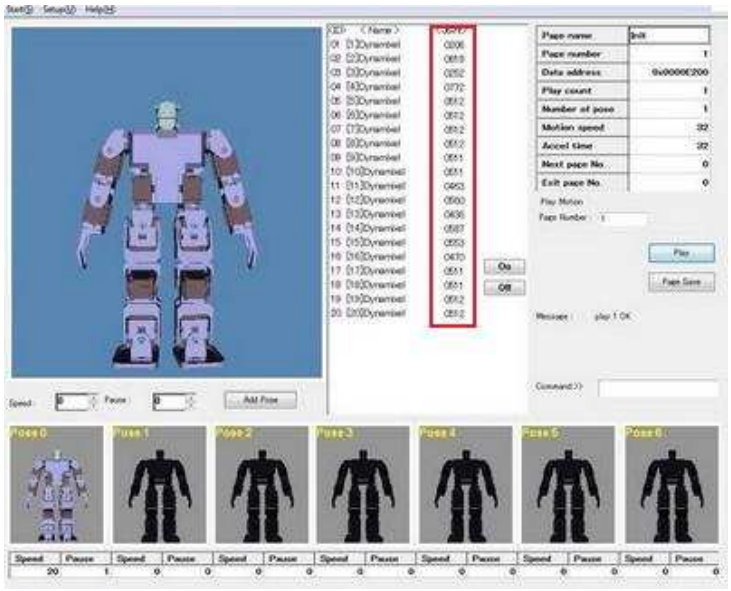

Figure 17: Bioloid Motion Editor Software.

Figure 18 represents the maximum movement range for the ID2 left arm servomotor (shoulder yaw movement). Its symmetrical for the right arm would be the servomotor with ID1.

On the Bioloid nomenclature, the shoulder yaw range limits for the left arm are:
Position 1000 for the angle of -60º.
Position 100 for the angle of 210º.

The shoulder yaw range limits for the right arm are:
Position 0 for the angle of -60º.
Position 900 for the angle of 210º.

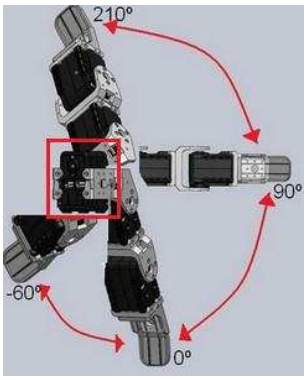

Figure 18: Movement range for the first left shoulder servomotor (ID2).

Figure 19 represents the maximum possible movement range for the ID4 left arm servomotor (arm pitch movement). Its symmetrical for the right arm would be the servomotor with ID3. On the Bioloid nomenclature

On the Bioloid nomenclature, the arm pitch range limits for the left arm are:
Position 815 for the angle of 0º.

Position 210 for the angle of 180º.

The arm pitch range limits for the right arm are:
Position 210 for the angle of 0º.
Position 815 for the angle of 180º.

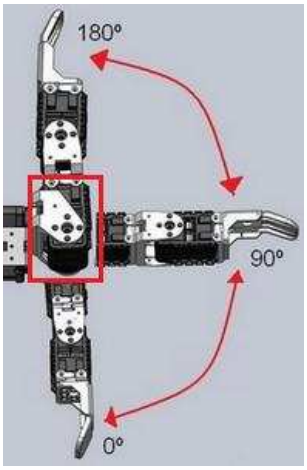

Figure 19: Movement range for the second left shoulder servomotor (ID4).

Figure 20 represents the maximum possible movement range for the ID6 left arm servomotor (elbow bending). Its symmetrical for the right arm would be the servomotor with ID5.

On the Bioloid nomenclature, the elbow range limits for the left arm are:
Position 200 for the angle of -90º.
Position 800 for the angle of 90º.

The elbow range limits for the right arm are:
Position 800 for the angle of -90º.
Position 200 for the angle of 90º.

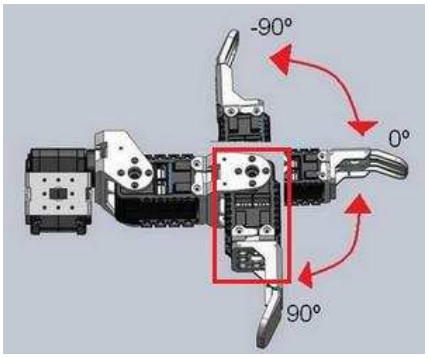

Figure 20: Movement range for the left elbow servomotor (ID6).

The Bioloid arm configuration is also used by many industrial robots designed for a wide range of applications including welding, painting, etc. One example of those industrial robots that present in the IRI Laboratories is the RX160 from Stäubli, represented on Figure 21. This robot has three extra degrees of freedom allowing different hand orientations, but the arm DOF

configuration is the same.

Figure 21: Industrial robot RX160 series from Staubli.

5.6 USB To Dynamixel

The USB To Dynamixel Adapter from Robotis allows to communicate with the Bioloid servomotors without involving the robot microcontroller during this process. This device is ideal in our case because our objective is not to create an algorithm to improve the autonomy of the robot, but instead we want to provide the robot with the capability to imitate a human by processing the data in a computer. The personal computer offers a higher computation power and this is of vital importance if the goal is to implement a real time imitation. On Figure 22 the USB to Dynamixel adapter is displayed and on Figure 23 we show how it allows to connect the Dynamixel AX-12 servomotors to a PC through a USB port. To do so, one of the AX-12 servomotors has to be connected in series and one of them has to be connected to the USB to Dynamixel adapter.

Figure 22: USB to Dynamixel adapter.

In order communicate with the Dynamixel servomotors, two libraries programmed with C++ are provided with the USB to Dynamixel adapter. These libraries are the *ftd2xx.lib* and the *dynamixel.lib*. The main functions of these libraries allows to activate or deactivate the USB to Dynamixel data transfer, and to write and read data from the servomotors. This data correspond to different parameters that can be set for each servomotor: angles, velocities, current intensity, etc. Next we present the four main functions used on the project.

The following functions are used to initialize and terminate the data transfer from the PC

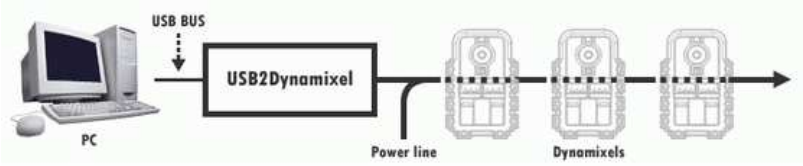

Figure 23: USB to Dynamixel connection diagram.

to the Bioloid servomotors:

```
dxl_initialize();
dxl_terminate();
```

The main functions are the following two, that are used to write and read data from the servomotors:

```
dxl_write_word(servomotor_ID, register, data_to_be_written);
dxl_read_word(servomotor_ID, register);
```

The first parameter used to operate these functions is the ID from the servomotor with which we want to work, allowing to communicate to a specific servomotor. The second parameter is a number which corresponds to the register and indicates the kind of data that we want to read or write. Along the project we have only used three different registers: the 30, the 32 and the 36. The register 30 is used to write a goal angle to a servomotor, when we write in this register, the servomotor moves until reaching the angle indicated by the third parameter: `data_to_be_written`. The 32 is used to set the servomotor speed before to move to a specific angular position. The two precedent registers are used in the `dxl_write_word` function. The register 36 is used to read the current angular position where the servomotor currently is, and it is used in the `dxl_read_word` function. The previously described functions are all we need to communicate and give the precise orders to the robot servomotors.

6 IMPLEMENTATION AND SOLUTION PROPOSED: IMITASOFT

6.1 Introduction: imitaSoft Organization

On this section we present the solution retained and the software implemented to apply this solution, we have called this software **imitaSoft**. ImitaSoft is a Graphical User Interface (GUI) created to merge all the aspects necessary to perform a human imitation using the Bioloid robot with input information coming from different inputs. Figure 24 shows the main screen of imitaSoft.

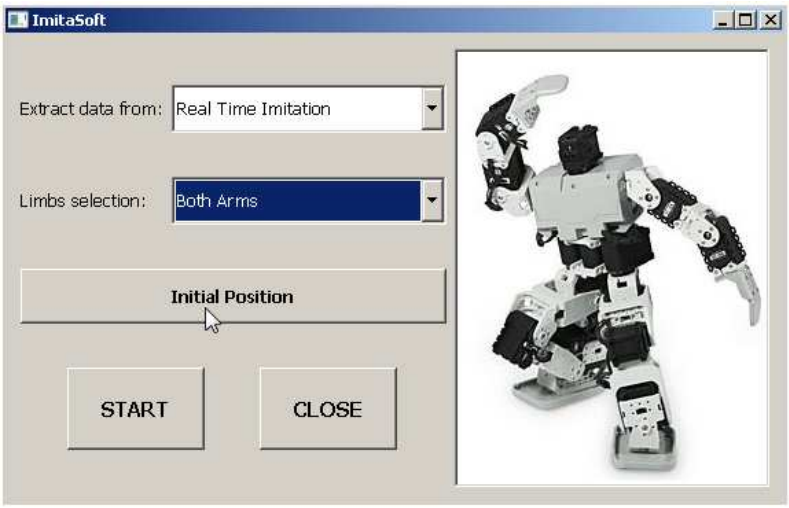

Figure 24: Main screen of the created software: imitaSoft.

ImitaSoft allows Bioloid to perform an unobtrusive imitation of the human arms movement in real time using the input information given by the SR3000 camera. This unobtrusive real time imitation becomes possible thanks to a hand tracking and a mapping between the human and the robot movements that we have created with this objective.

ImitaSoft also allows Bioloid to perform an imitation of the human arms in deferred time using the XML information from VooDoo sequences. Working in deferred allows us to chose the tracking VooDoo sequences of better quality in order to perform the mapping from there.

Moreover the software created is able to store the servos angles and velocities that has been computed during the mapping. This information can be stored both, when using a XML VooDoo sequence as input information or when using the SR-3000 with our own tracking in order to perform the imitation. Storing this angles and velocities is seen as 'learning' the movements, that is to say, once the imitation has been performed once, imitaSoft allow Bioloid to reproduce the same movements every time you want.

In fact, learning in the full sense of the word, would imply not only to be able to reproduce exactly the same movements, but also to infer new movements based on some initial movements taught by demonstration; the initial movements taught by demonstration are the base for the algorithm that will allow the robot to learn by means of trial and error and to correct its actions little by little. The reinforcement learning looks for maximize some notion of cumulative reward until the desired action is achieved. This kind of learning for robots has been explored by many research groups. One example is the technology developed in the Italian Institut of Technology. This group has created a software able to teach by reinforcement and demonstration a WAM

robot arm to flip pancakes using a pan [22]. This group has posted some videos [6].

Our software imitaSoft allows only to perform the first step of a full robotics learning software, which is the step of learning the input movements by demonstration and reproduce them without modifications. In our case the demonstration would be done without direct contact of the robot but it would be done by imitation of the human movements. From now on we will refer to the reproduction of new actions acquired as "learning".

Thus, imitaSoft allows to "learn" new movements by imitation, giving a very efficient way to program new movements. At the end we present some experiments developed in order to chose the most suitable parameters for imitaSoft and to present the limitations of the software and how we have to work with it.

In this way the final diagram schematizing these three different options of imitaSoft is presented on Figure 25.

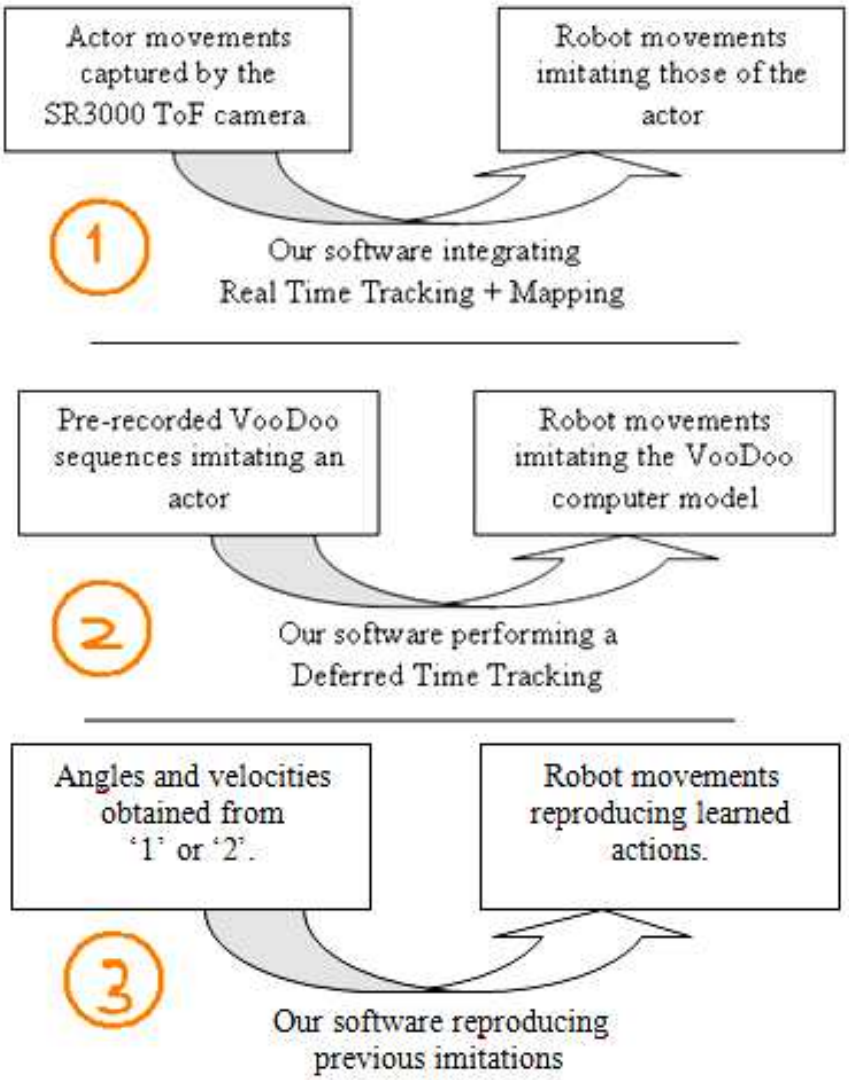

Figure 25: General diagram displaying the imitaSoft capabilities.

[6] Video of a WAM robot learning how to flip a pancake: http://www.youtube.com/watch?v=W_gxLKSsSIE

Next, we present the distribution of the imitaSoft GUI, its main zones and its menus with the options that can be selected. The imitaSoft GUI has been divided in three main zones: one zone with two drop down menus, another zone with the action buttons and a third zone used as the images display zone.

On the first zone we find two drop down menus. The first one is represented on Figure 26. This menu is the main one and it allows us to chose between the three different ways of working of the program:

- *Real Time Imitation*: This option allows Bioloid to perform a real time imitation using the input information from SR3000 and executing our own developed tracking and mapping.

- *XML VooDoo Sequence*: This option allows Bioloid to perform a deferred time imitation using some previous stored XML VooDoo sequences and executing our own developed tracking and mapping.

- *Angles and Velocities Files*: This option allows to reproduce Bioloid movements that have been 'learned' by executing one single time one of the first two options.

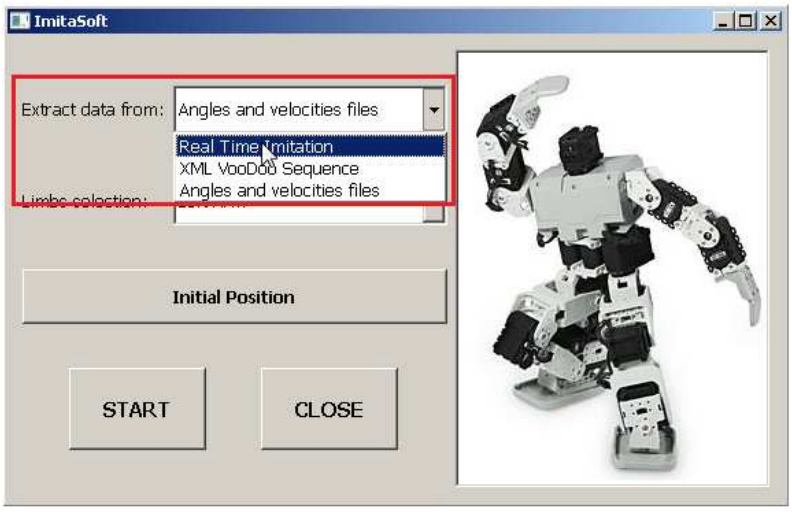

Figure 26: Main menu allowing to chose between deferred time, real time imitation and learned movements.

The second drop down menu represented on Figure 27 allow us to chose between a right arm mapping, a left arm mapping or a both arm mapping. This options are available for each one of the input choice in the first menu.

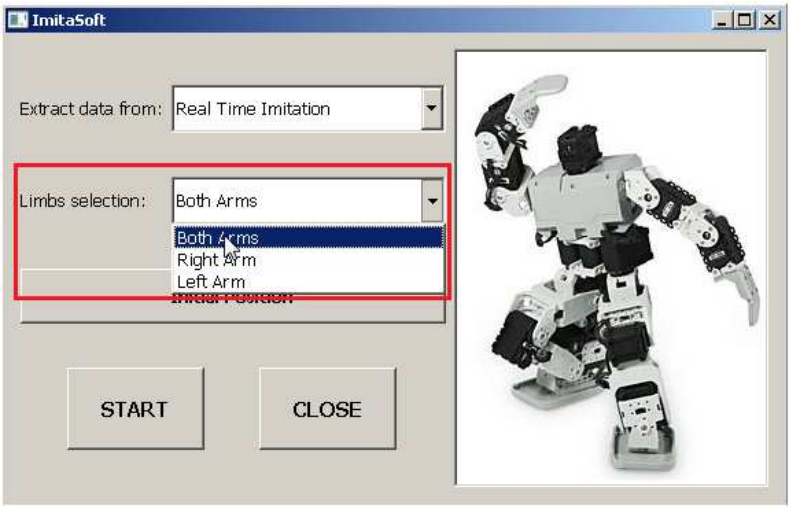

Figure 27: Drop down menu allowing to chose the arm for which we want to perform the imitation.

The action button zone presented on Figure 28 is often composed by three different buttons. However choosing one option or another in the main drop down menu, modifies the action buttons and configuration of this zone that is adapted to the main option selected after pressing the START button. Thus, the START button is used to begin the actions that we have selected in the two drop down menus. The CLOSE button is used to shut down the imitaSoft the application.

A third button is present in this zone. The Initial Position button sets a specific stable standing position for the Bioloid robot. This button is used to fix the legs servos before to begin with the arm imitation. The position adopted once the this button is pressed is shown on Figure 29.

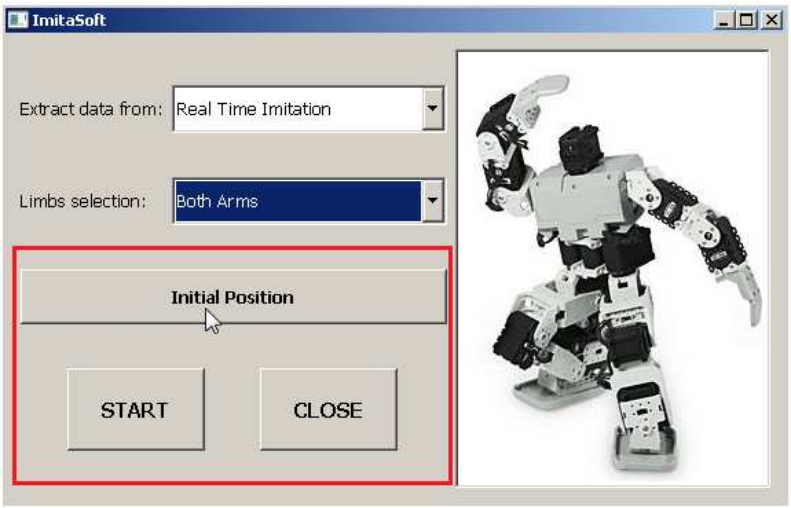

Figure 28: Action button zone.

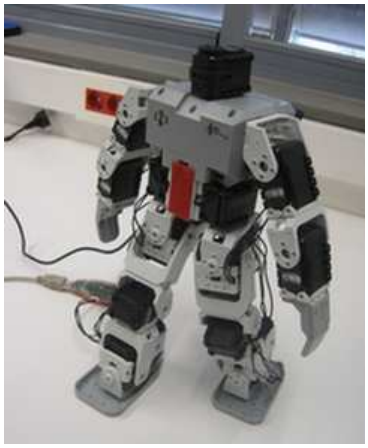

Figure 29: Initial Stable standing Bioloid Position.

The area represented on Figure 30 corresponds to the image display zone. This area is mostly used when working with the Real Time Imitation because is in this area where we visualize the tracking. For the other two options of the main menu, as we are working with other input information sources this area is not used and shows the image of Bioloid that we observe on Figure 30.

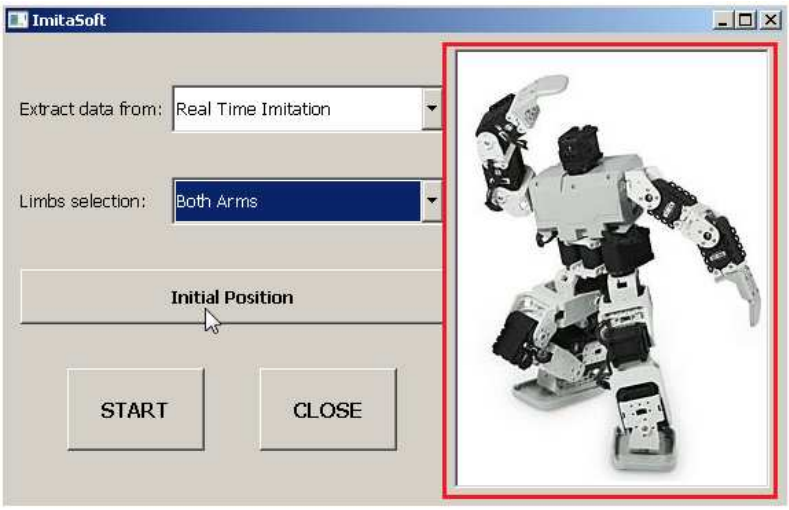

Figure 30: Images display zone.

6.2 Real Time Imitation

6.2.1 The Tracking

The two main parts of the Real Time Imitation module from imitaSoft are a tracking used to extract information about the arm positions and a mapping allowing to map the human arm positions to robot arm positions, imitating the human. On Figure 31 the diagram of this module is presented. In this subsection we present the tracking part from this diagram.

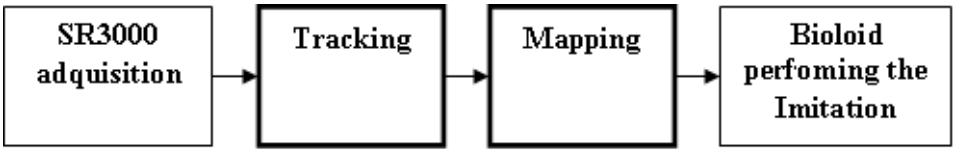

Figure 31: Diagram of the Real Time Imitation module from imitaSoft.

Once we have selected the Real Time Imitation option on the main menu and we have pressed the START button, the SR3000 image acquisition begins displaying its images on the image display zone and the button zone configuration is modified. If the SR3000 is not connected or not detected by the PC an error message appears (Figure 32).

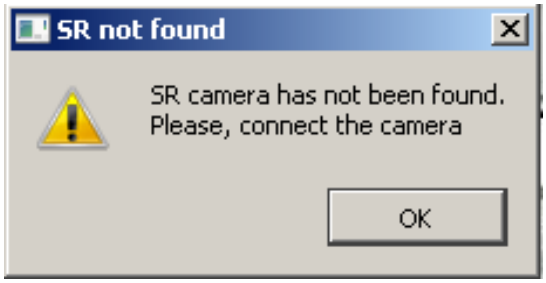

Figure 32: Warning message appearing if the SR3000 camera is not detected by the PC.

If the SR3000 is connected the video acquired is displayed and two new buttons appears: a button called *START SR TO BIOLOID* allowing to begin the human imitation performed by Bioloid using the information from the SR3 camera and a *STOP* button allowing to stop this imitation without shutting down the program. On the image display zone two different videos are displayed: one of them corresponds to the intensity image given by the SR3000 camera (at left), the other represents the 3D depth information which corresponds to the SR3000 z coordinate (at right). This depth image is displayed with a gray scale gradient with clearer colors for objects closer to the camera and darker colors for farther objects. Figure 33 shows the new button zone distribution and the intensity and depth tracking frames for a given instant of time.

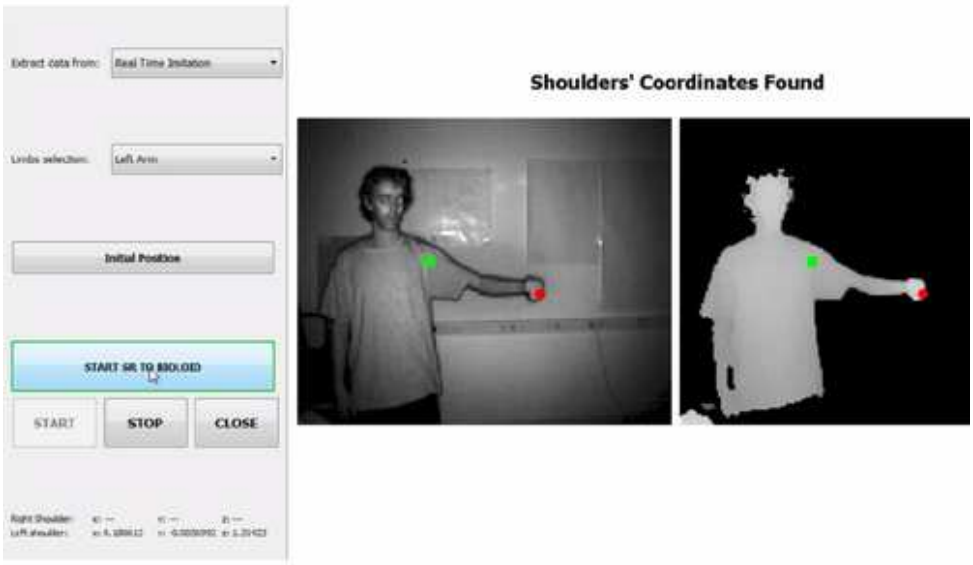

Figure 33: Image of the Tracking used for the Real Time Imitation.

Once the *START SR TO BIOLOID* is pressed the tracking begins and the information obtained from it, feeds the mapping in real time. The arm imitation performed is goal oriented. This means that we will be interested on the hand positions more than on the arm configuration, it is for this reason that we have implemented a hand tracking. As Bioloid only has 3 DOF for each arm, knowing the hands position with regard to the rest of the body is almost enough to determine the arms configuration without ambiguities. We will see that in more detail on section 6.2.2.

As we have seen on section 5.2 SR3000 camera images gives 3D information about its images or frames. In fact, all the points given by the SR3000 can be projected on a plane (the computer screen) and thus the information given by this sensor is considered to have only 2.5 dimensions because we have not information about the whole image but only the visible part to the camera. However, each one of its 176x144 pixels indicates a Cartesian coordinate with its three components x, y and z. Thanks to this fact we have implemented a totally unobtrusive tracking based on the SR3000 information. This tracking can be performed for the right arm, for the left arm or for both arms.

What we do in order to determine the position of the hands by respect to the rest of the body is to look for the shoulder-hand vectors. These vectors will be the input information for the mapping described on section 6.2.2.

Knowing that the used SR3000 camera used as input sensor to perform the tracking returns 3D information with the Cartesian coordinates of each pixel, we have delimited a given 3D area inside which the actor will be placed for its tracking. The objective of doing this is the segmentation of the actor, isolating him inside the SR3000 video frames in order to eliminate noise and objects that have not to be tracked. This 3D zone has a cubic shape and has been delimited by identifying maximum values for x, y and z coordinates. Everything being outside of this cubic zone will not be displayed on the images and the information coming from this external pixels will not be taken into account. Figure 34 is a diagram representing the delimited tracking area; the coordinates axes for the SR3000 images are also displayed. This cubic space is bounded by the following distances: in depth (z coordinate) between 0.8 and 2.8m, in height by -1m and 1m (y coordinate), and in width (x coordinate) it depends on the mapping which

we are working: single arm or both arms.

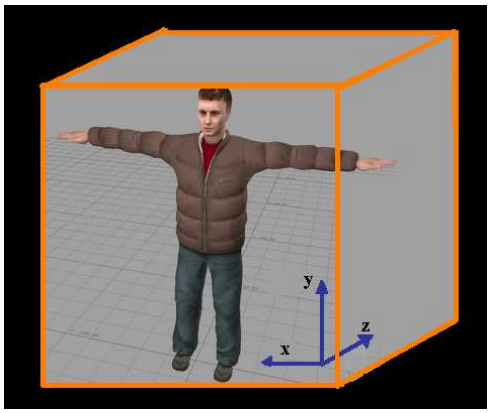

Figure 34: Cubic space within with the tracking is carried out and image coordinates axes.

Shoulders Coordinates Acquisition:

After pressing *START SR TO BIOLOID*, the program gives a short initial time to the actor to position himself in front of the SR3000 camera. During these few seconds the actor has to be placed at the center of the SR3000 field of view. The first thing that imitaSoft will do is to extract the shoulders coordinates. Two green square marks are present on the imitaSoft image display, they allow the actor to know in which location he has to be positioned in order to be tracked. The actor has to position its shoulders in line with the green marks as it is shown on Figure 35. In this figure we also observe the actor segmentation on the SR3000 depth image, the rest of the objects present on the intensity image is not present anymore on the depth image thanks to this segmentation.

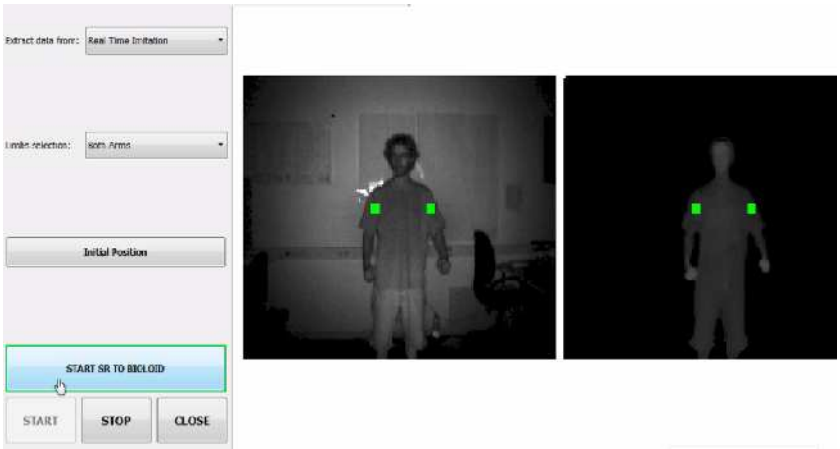

Figure 35: Actor position on the SR3000 camera field of view. The shoulder position must correspond with the green marks position on the image.

Once the initial time has elapsed, if the actor has been well positioned with its shoulders in line with the green square marks the shoulder coordinates are extracted. For doing so, imitaSoft takes one simple SR3000 frame and makes the average from 9 pixels situated in line with the green square. We observe the diagram of this shoulder coordinates extraction on Figure 36.

This average is done for the x, y and z coordinates. The process is performed for the right and for the left shoulders independently.

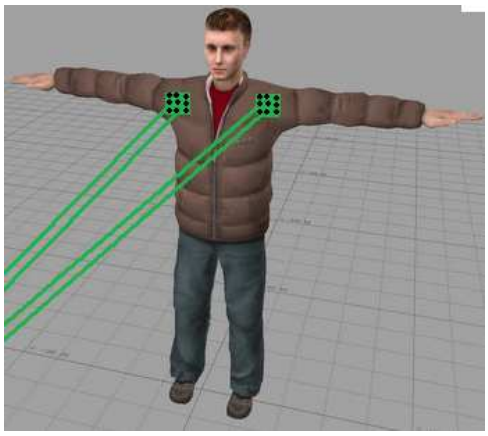

Figure 36: Shoulder coordinates extraction.

If the actor shoulders are not placed in line with the green marks and all the pixels in line with these marks are outside the tracking cubic area, the shoulder coordinates cannot be determined and a message indicating this fact is displayed on the imitaSoft image display zone. In this case the hand tracking does not begin because the vector shoulder-hand cannot be determined if we don't have the shoulder coordinates. On Figure 37 the message appearing when the left shoulder coordinates could not be determined is shown; on this figure we can see that the actor left shoulder was not in line with the green mark. In this case the process must be restarted by clicking another time on the button *START SR TO BIOLOID*.

Figure 37: Error on determining shoulder coordinates.

If the actor shoulders are in line with the green square marks, the shoulder coordinates are found and the hand tracking begins. Once the shoulder coordinates are extracted, they are the same for all the tracking duration, thus the actor must stand at the same place during the hand tracking, moving its arms freely allowing Bioloid to imitate this arm movements.

Unobtrusive Hand Tracking:

Most of the human arm movements that we perform usually are carried out in front of the body coronal plane (also known as frontal plane). On Figure 38 the main body planes are displayed. This is due to the human arm mobility which is higher in front of this plane rather than backwards and also due to the visual feedback that we have when performing activities with the hands in front of this plane.

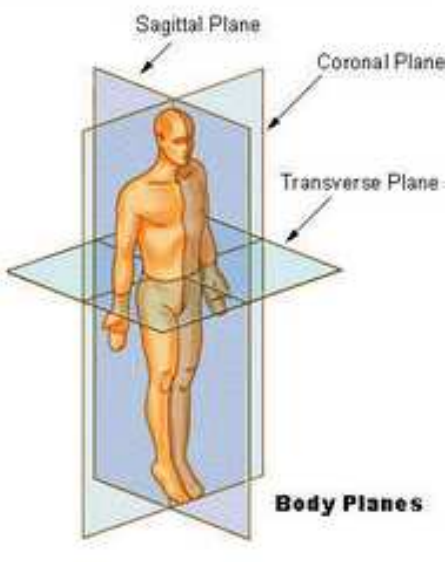

Figure 38: Main human body planes.

Since we are able to obtain 3D information with the SR3000 camera and knowing that hands are mostly carrying out actions in front of the coronal plane we have based the hands tracking on proximity information. The main idea is to track the body limbs closer to the camera. This simplifies the tracking as we are able to focus on the hands which will be the closer limbs to the camera. Moreover this way of working will allow us to obtain a totally unobtrusive tracking since we do not need any special body marks to perform the tracking. ImitaSoft implements a local hand tracking in order to hinder the loss of the tracking and to diminish the number of calculations. In real time it is always important to reduce the computations since they are performed in loop and the elapsed time between two rounds of loop must be the shortest possible to have near real time result. Thus, the number of computations is limited by the elapsed time between two rounds of loop.

After this short introduction to the principle ruling the imitaSoft hand tracker, we will detail it by presenting the developed hand tracker used to track both hands at the same time. The single hand tracker uses the same principle with some simplifications.

The hand tracking developed can be divided into two main stages. The first one is used to initialize the hand tracking and find the global hands location on the SR3000 video. Once the initial hands location has been found, the second stage performs a local hand tracking allowing to follow the hands be looking for them near to the location where they were in instant immediately preceding.

During the first stage, the hands must be situated in front of the body, that is to say in front of the coronal plane, and one at each side of the body: left hand at left, right hand at right. The initial actor position placing its hands in front of the body is displayed on Figure 39.

Figure 39: Initial actor position with their hands in front of the body to initialize the tracking.

What it is done in order to determine the hands coordinates is to find its location inside the SR3000 images thanks to the tracking and then the SR3000 information gives directly the three coordinates for this given pixel which corresponds to a given location. The hands are tracked inside the SR3000 video frames thanks to two red small circles which follow them.

To determine these two initial locations where are placed the hands for the first round of loop, the image is divided into two parts. This initial search is carried out independently for the right hand and for the left hand. This division is shown on Figure 40 where we have drawn an orange square and a blue square representing both halves.

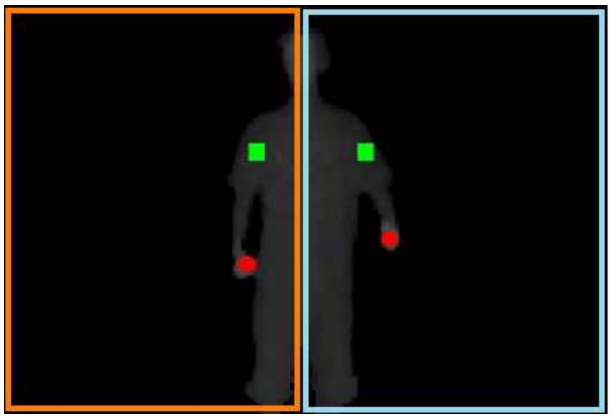

Figure 40: Initial tracking with the image divided in two main parts. Two red circles track the hands.

The tracking technique is explained for one simple hand; the other hand tracking follows the same principle. In order to find the right hand location it should be enough to find the pixel inside the orange square with the closest distance to the camera by comparing the **z** coordinate from all the pixels of the image and taking the pixel for which this coordinate is the smallest. The **z** being the coordinate which represents the depth from the camera to a given point of the image.

However, after performing this kind of simple tracking on the laboratory, it was not performing well because of two main reasons. In fact the SR3000 has some dead pixels which

indicates always the same distances and does not work properly anymore. Second, because there are some reflector elements such as glasses, which introduces a lot of noise in the image. But even without these elements some pixels are randomly noisy. We have appreciated this fact after implementation of this simple global tracking, because even having the hands in front of everything else, the red tracking points were moving chaotically, following sometimes the hands and getting completely lost other times. To solve this problem two ideas have been implemented: one has been to implement a local tracking after the initial global tracking and the other has been to look for groups of pixels being close to the camera instead of looking for single pixels close to the camera.

Since hands are objects that are extended to more than one pixel on the image, imitaSoft tracking will look for groups of pixels being close to the camera. Doing that, we eliminate the noise of random single pixels representing short distances to the camera which could distort the tracking. In order to find the right hand what we do is to search a group of 25 pixels being the pixels which represents the closest points to the camera inside the orange zone represented on Figure 40. Once we have these 25 pixels minimizing the **z** coordinate, we calculate the distances in pixels between a given pixel and everyone else by using the euclidean distance. The next equation shows an example of how to calculate the distance in pixels for the two given pixels shown on Figure 41.

$$distance = \sqrt{a^2 + b^2} = \sqrt{2^2 + 2^2} = \sqrt{8}$$

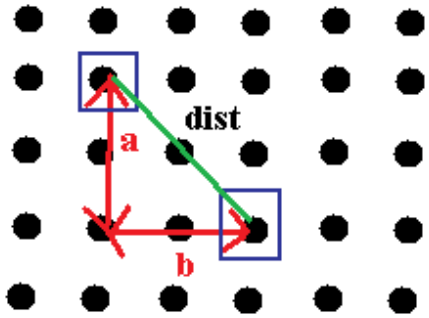

Figure 41: Distance between pixels in a pixel scale.

After that, we count the number of pixels that are closer than a certain distance in pixels. We have used a distance of 6 pixels; this means that we will count all the pixels around a given pixel having a distance to this pixel smaller than 6 pixel unities. We do the same process for every one of this pixels. Figure 42 shows this process, we can observe the zone delimited by a given distance for two different pixels: a pixel represented in green which has an higher density of pixels around it (3 more pixels), and a pixel represented in black which has no more pixels inside the area around it. The last thing done is to keep the pixel which has the larger number of pixels around it. That will mean that this pixel is inside a dense group of pixels and the probability to find a hand there is bigger than the probability to find a hand there where the density of closer pixels to the camera is small. The 3D coordinates of that pixel will be taken as the hand coordinates and the initial global right hand tracking finishes here. The initial global left hand tracking uses the same principle used for the right hand but instead of looking inside the orange half zone the left hand is looked for inside the blue zone represented on Figure 40.

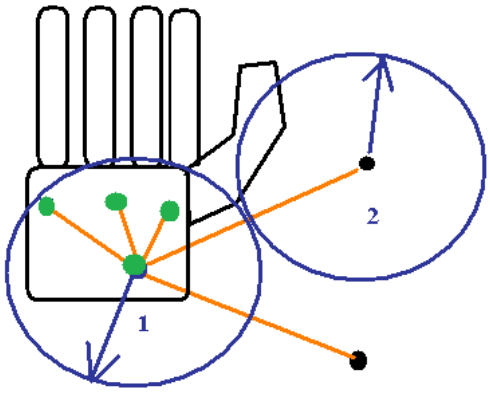

Figure 42: Search of groups of pixels close to the camera to find the hand.

Now that the initial hand positions has been found inside the global scene, we already dispose of a reference hands position inside the image which is its current position. This will allow to perform a local tracking from now on. This local tracking uses the same principle used by the initial global tracking, looking for dense groups of close pixels which having more probabilities to find the hands inside the image. The difference is that with the local tracking we are not searching the hands inside the whole image, instead of that we will look for the hands position only in a limited area around the precedent hands position on the image. With that we are limiting the number of calculations required to find the 25 closer pixels and at the same time we are reducing the probabilities to lose the tracking because we are always looking around the precedent positions where the hands were found. Imitasoft uses local areas of 30 by 30 pixels, centered at the precedent tracked hand position (Figure 43). This windows must be big enough to don't lose the tracking avoiding the hand to go out of the area between a round of loop and the next one. But they must be as small as possible to reduce the number of computations and to focus the tracking avoiding to lose the tracking because of possible noise, crossing hands, etc.

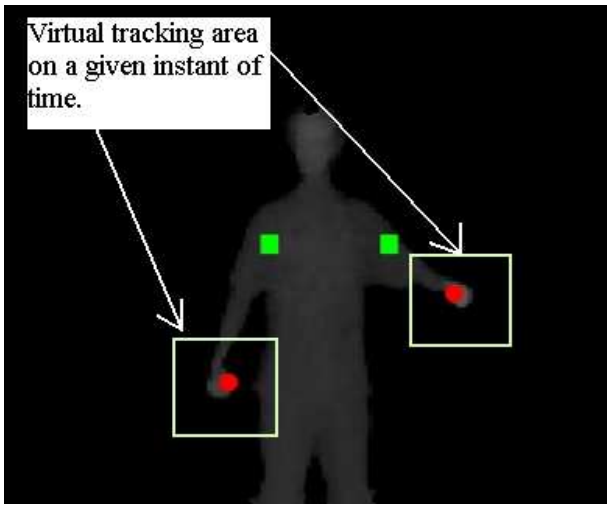

Figure 43: Local tracking represented in this image by two squares around the hands (Not represented during the tracking; only used in the algorithm.)

This local tracking only can work if it is performed on real time, that is to say, if the

elapsed time between two rounds of loop is to long the hand will have time to move farther and it will have time to go out of the search area occupied by the precedent hand position making impossible to find the current position in the area. The time between tracking frames must be as short as possible in order to set small local areas around the precedent position. Shorter is the elapsed time between frames, smaller will be the searching local areas and more difficult will be to lose the tracking. However, the elapsed time between frames must be long enough to have time to make all the tracking and mapping calculations in order to operate the robot, thus a compromise has been found.

The minimum elapsed time between frames that we can use its bounded by the SR3000 integration time, which is the time that needs SR3000 to do a frame acquisition. The integration time must be adapted in function of the distance of the object that must be acquired with the camera. In our case we have used an intermediate integration time of 18ms which allows to find quite good results on the cubic space used inside which the actor is performing the actions. In section 6.5.1 an experiment is carried out to explain why this value has been chosen.

ImitaSoft uses a time of 70ms between two tracking frames, with that we obtain around 15 frames per second which allow to almost preserve the continuous movement sensation during the hand tracking and gives some time to perform the algorithm tracking and mapping computations.

With this local tracking we obtain, one instant of time after another, the positions of the pixels occupied by the hands on the image frames. Having the pixels position on the image we position the red points tracking the hands exactly there and we extract the 3D coordinates of them, given by the SR3000 camera. Figure 44 is the hand tracking algorithm diagram resuming the text above.

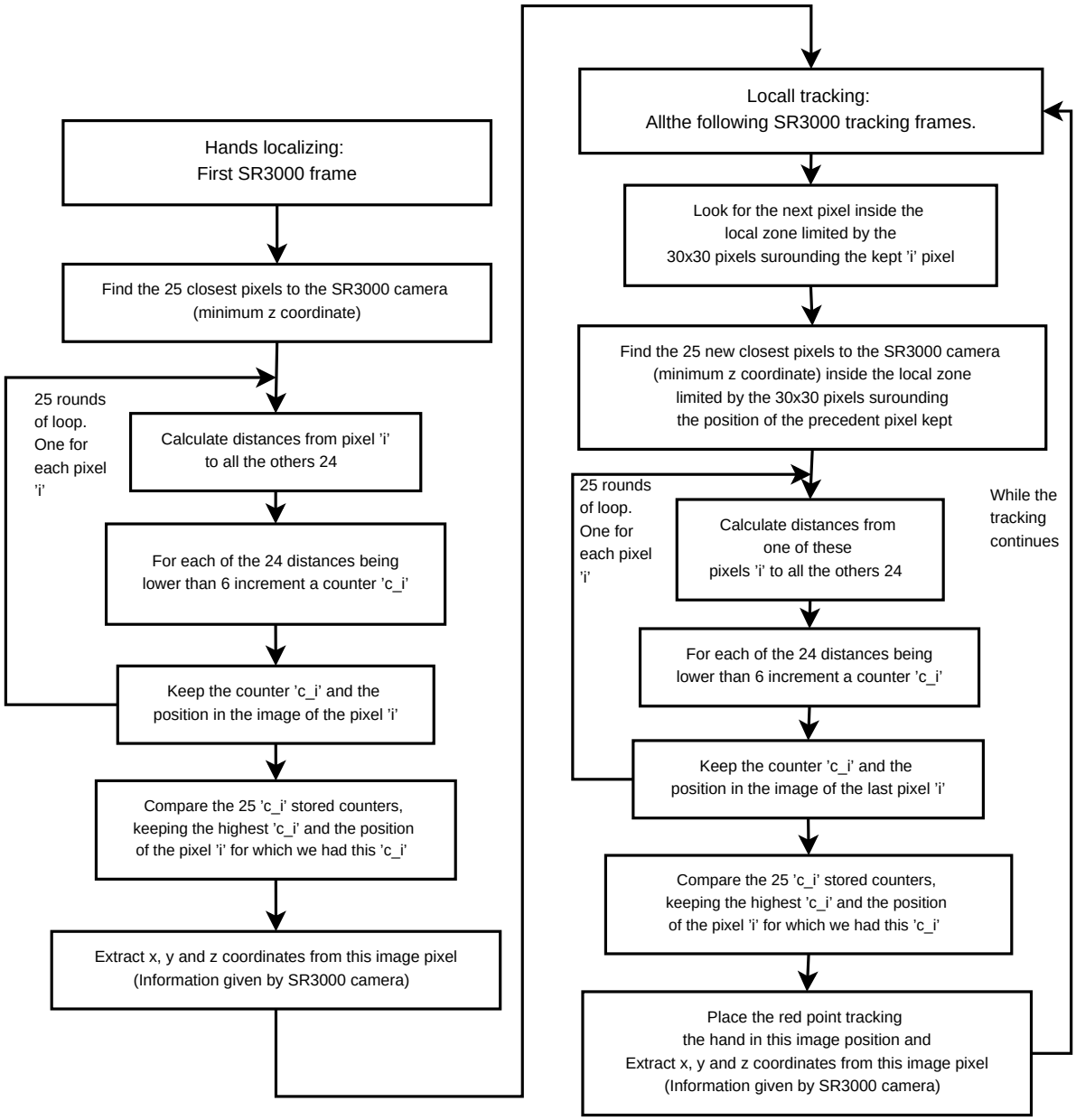

Figure 44: Diagram of the hand tracking algorithm.

With the found shoulders 3D coordinates found at the beginning and the 3D coordinates from the hands found thanks to the tracking, we can already extract the vectors 'shoulderHand' for both arms and for each time step t_i, which components are:

$$shoulderHandRight_x(t_i) = Hand_x(t_i) - Shoulder_x(t_i)$$

$$shoulderHandRight_y(t_i) = Hand_y(t_i) - Shoulder_y(t_i)$$

$$shoulderHandRight_z(t_i) = Hand_z(t_i) - Shoulder_z(t_i)$$

$$shoulderHandLeft_x(t_i) = Hand_x(t_i) - Shoulder_x(t_i)$$

$$shoulderHandLeft_y(t_i) = Hand_y(t_i) - Shoulder_y(t_i)$$
$$shoulderHandLeft_z(t_i) = Hand_z(t_i) - Shoulder_z(t_i)$$

These precedent vectors will be used to map the human positions to similar robot positions allowing to perform the human imitation. The mapping algorithm is explained on section 6.2.2.

On the text above, we have explained how imitaSoft works to perform the tracking of both hands. The tracking of an individual hand (right or left) has also been developed and it uses the same principle. The only difference is that for the initial global tracking the image is not divided into two parts and this initial tracking is done on the whole image. In this case, only one shoulder is found and only one hand will be tracked.

6.2.2 The Mapping

This subsection will present the mapping algorithm developed, which uses the information in form *shoulderHand vectors* that we obtain thanks to the tracking. At each round of loop, the information coming from these vectors is processed in order to extract new angles and velocities for the Bioloid servos. Thus, imitaSoft will map in real time the human movements to similar robot movements.

The first thing that we have decided has been the procedure retained in order to adapt the movements from human to robot. As we have seen in section 5.5, Bioloid only has three servos per arm and this reduces the number of possible positions that can be taken by Bioloid in regard to the positions that can be taken by a person.

Next, we present the main arm human movements allowed by the shoulder and elbow joints. The shoulder is composed by three bones that are the clavicle, the scapula and the humerus. The articulations between these three bones make up the shoulder joints. The main shoulder movements are:

- The scapular retraction and protraction: These movements allows to move the shoulder posteriorly and anteriorly, by squeezing the shoulder blades together or by squeezing the pectoral muscles together, respectively (Figure 45).
- The scapular elevation and depression: The scapula is raised or lowered from elevation respectively (Figure 45).

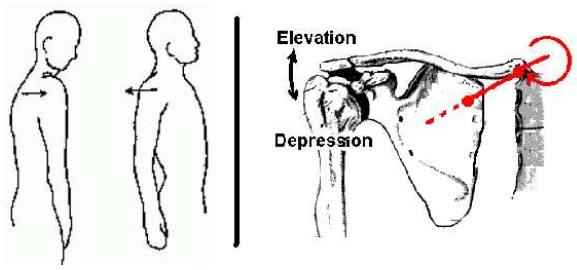

Figure 45: Scapular movements.

- Arm flexion and extension: The arms first held at the torso sides rotate around the axis parallel to the clavicle passing through both shoulders, going out of the coronal plane (Figure 46). This movement occurs in a plane parallel to the sagittal plane [23].

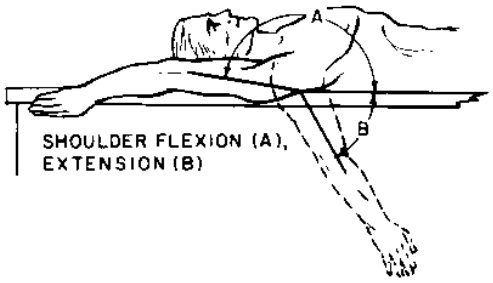

Figure 46: Arm flexion and extension movements.

- Arm abduction and arm adduction: Arm abduction is the movement which draws the arms away from the sagittal plane of the body. The adduction is the opposite movement, when the arms approach the sagittal plane (Figure 47). Usually, these movements are described parallels to the coronal plane (vertical abdcution) or parallels to the coronal plane (horizontal abduction).

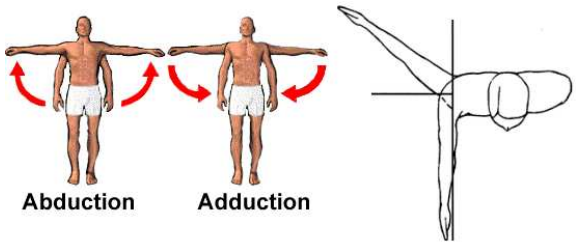

Figure 47: Vertical and horizontal arm abduction and adduction movements.

- Internal rotation of the arm around its longitudinal axis (Figure 48).

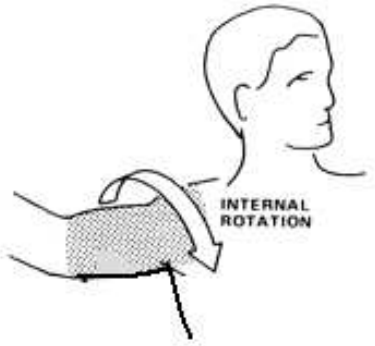

Figure 48: Rotation of the arm around its longitudianl axis.

The main movements allowed by the elbow:

- Flexion and extension between the humerous and the ulna creating an hinge-joint. This articulation allows the bending and straightening between the arm and the forearm (Figure 49).

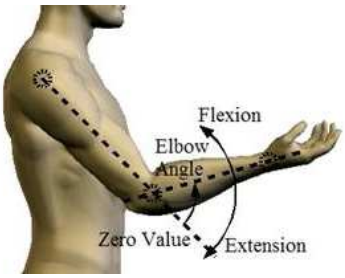

Figure 49: Elbow flexion and extension movements.

- Pronation and supination: Rotational movement of the forearm around its longitudinal axis allowing the palm to face down or up respectively (Figure 50). Both, the elbow and the wrist articulation take part in this movement [24].

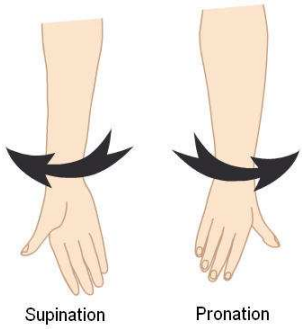

Figure 50: Pronation and supination forearm movements.

As we have seen, the human arms are really complex limbs which have a wide range of possible movements. However the robots created have often more simple arm structures. In order to map the human positions to a robot we have to take into account which movements are common between the robot and the human, which are similar, which ranges have these movements and which strategy we will use to allow the robot performing the imitation.

On section 5.5 we have seen the possible Bioloid movements. This robot has only 3 servos per arm, allowing three different DOF. On Figure 51 we observe the basic structure of the Bioloid arm with its three DOF. As we can see in this figure, Bioloid does not allow any scapular movement at the shoulder level, nor the internal rotation along the arm longitudinal axis. Moreover, the pronation and supination movements allowing a human to rotate its forearm along its longitudinal axis are neither possible by Bioloid.

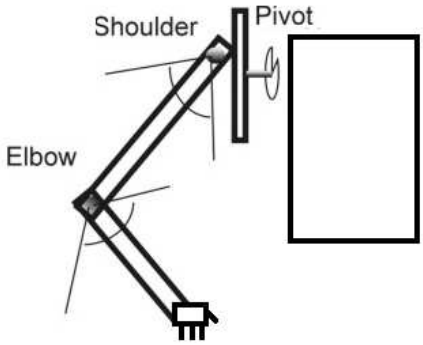

Figure 51: DOF of the Bioloid robot arm.

The Bioloid movements are reduced to the arm flexion and extension, the arm abdcution and adduction, and the shoulder flexion and extension. The shoulder 3D movements of the robot for the right arm can be seen on Figure 52, where the first servo in the arm chain allows the flexion/extension movements represented by the angle between the A and the L axis. The second servo allows the angle between the L and the T angle, this movement could be seen as the abduction/adduction, however the plane where this movement occurs depends on the flexion/extension angle, because it is dependent on the first servo of the chain. Thus it does not always occurs in the coronal or in the transversal plane.

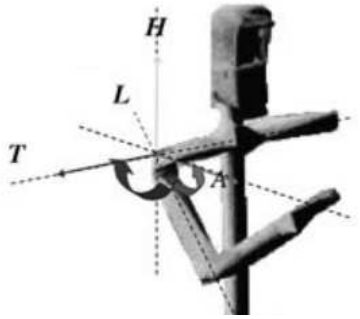

Figure 52: Allowed Bioloid shoulder movements.

With the Bioloid DOF configuration it would be possible for the robot end effector, its hand, to reach all the positions inside the sphere centered in the shoulder and with radius equal to the arm length plus the forearm length. However we have to take into account two important aspects:

1. The Bioloid end effector can reach most of the positions inside this sphere, but this does not implies that the robot arm will preserve the human position. In fact, the relative position of the arm in regard to the shoulder can be maintained by dint of losing the perfect imitation of the arm position because of the low number of the Bioloid arm DOF.

2. The angular ranges of the Bioloid servos are limited by its own angular range (between 0^o and 300^o) and by its mechanical constraints and collisions with the robot itself. Above all, the ranges are imposed by the mechanical constraints between a robot limb and its next limb.

Knowing **1**, we have chosen to perform a goal oriented imitation which will keep the relative position of the hand in regard to the shoulder of the same arm. The structural position of the robot arm with the angles of the servos will be determined thanks to this relative position. Because of the only 3 DOF per Bioloid arm, the possible arm configurations reaching a certain hand position is low. In most of cases only four different configurations are possible to reach a certain Bioloid hand position. On Figure 53 we can see a diagram showing the possible Bioloid arm configurations for a specific hand position. In these positions we can see how Bioloid allows to bend the elbow to both sides, however the human arm cannot be bended to both sides, thus this helps us to eliminate two of the four configurations by preventing the Bioloid arm to bend backwards reaching non natural positions.

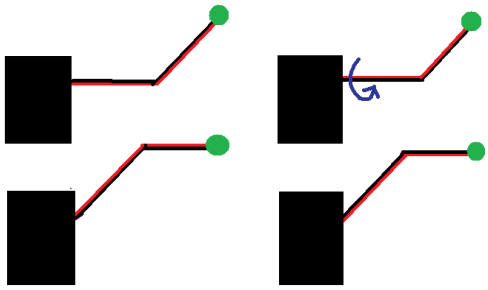

Figure 53: Possible arm configuration for a given end effector position.

As seen on section 6.2.1, the tracking is carried out in front of the coronal plane. This means that we will not study movements where hands are behind of this plane. Hands are most of time performing its actions in front of this plane because we have visual feedback of this actions. This fact will also allow to lower the range of movements in order to find a unique solution for the hand position on each instant of time.

Mapping to Bioloid:

After this introduction on the human and on the Bioloid arm movements we will present the method which uses imitaSoft to find the Bioloid arm angles using the 'Shoulder-Hand' vectors determined with the human tracking on section 6.2.1. These vectors indicating the relative human hand positions, will allow us to find the angles by inverse kinematics.

To do the calculations we have used a model of human arm with a length of 35cm for the forearm and a length of 35cm for the arm. All the calculations are done at the human scale but only taking into account the possible Bioloid movements with its 3DOF. Thus, we will calculate the angles necessary to reach a specific human hand position in regard to the shoulder of the same arm by using only 3 angles (equivalent to the Bioloid angles). Using the found angles and transforming them into the Bioloid nomenclature the imitation will be performed by the robot.

In order to calculate the Bioloid angles we have used a method based on basic trigonometry and vectors projections on the main Cartesian planes. For that, we have used the coordinates system shown on Figure 54. Where the **z** axis points up, the **y** axis is perpendicular to the

coronal plane and it points backwards, and the **x** axis is the vectorial product of **y** by **z**. It is important to note that these axis used on imitaSoft are not the same that those used by the SR3000 camera. For this reason we have to adapt the vector shoulder-Hand components to this new Cartesian system.

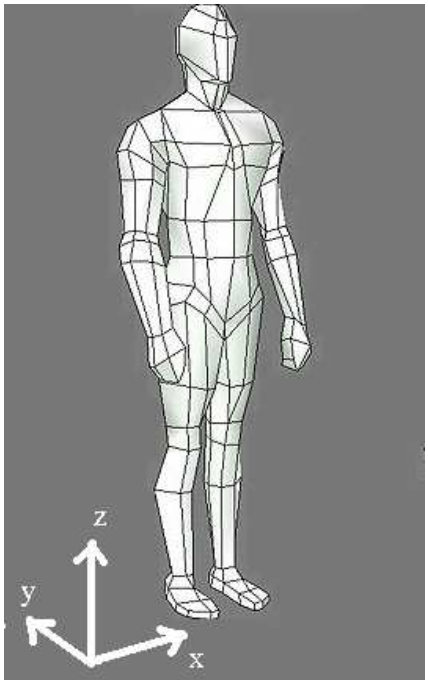

Figure 54: System of Cartesian axes used for the Bioloid arm angles calculation.

The relation between the components expressed in the SR3000 system and the components expressed in the imitaSoft system are:

$$shoulderHand_{x,\ in\ SR3000\ system} \rightarrow -shoulderHand_{x,\ in\ imitaSoft\ Cartesian\ system}$$

$$shoulderHand_{y,\ in\ SR3000\ system} \rightarrow shoulderHand_{z,\ in\ imitaSoft\ Cartesian\ system}$$

$$shoulderHand_{z,\ in\ SR3000\ system} \rightarrow shoulderHand_{y,\ in\ imitaSoft\ Cartesian\ system}$$

From now on we will only work with the axis and components of the imitaSoft Cartesian system of coordinates.

The first angle that we will deduce is the elbow angle. The elbow angle is defined by the angle formed between the arm prolongation and the forearm as represented on Figure 55.

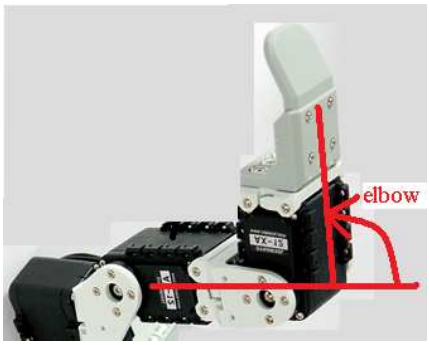

Figure 55: Bioloid elbow angle: 3rd servo of the arm chain.

The elbow angle has been calculated using the cosine theorem thanks to the following data:

1. The human arm length: a_{arm}
2. The human forearm length: b_{fore}
3. The length of the vector 'shoulderHand' extracted thanks to the hand tracking: c_{sh}

With these three vector lengths forming a triangle as shown on Figure 56 we can find the internal elbow angle (e_{int}) between the arm and the forearm, and subtracting this internal angle to 180º we find the elbow external angle that we were looking for.

Using the cosine theorem we have that:

$$c_{sh}^2 = a_{arm}^2 + b_{fore}^2 - 2 \cdot a_{arm} \cdot b_{fore} \cdot cos(e_{int}) \Rightarrow e_{int} = arccos(\frac{-c_{sh}^2 + a_{arm}^2 + b_{fore}^2}{2 \cdot a_{arm} \cdot b_{fore}})$$

And thus the elbow external angle that we are looking for is:

$$elbow = Pi - e_{int}$$

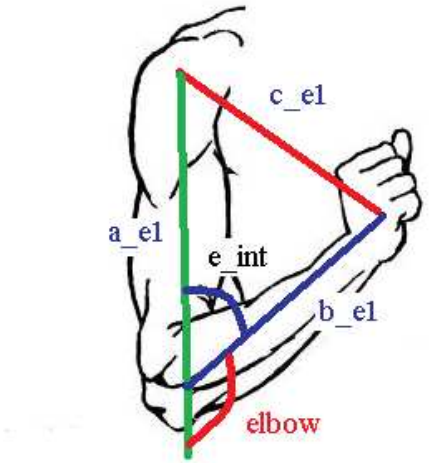

Figure 56: Elbow internal and external angles.

We have to take into account that the Bioloid external elbow angle, once prevented to bend backwards, has a range which oscillates between 0º and 90º, which is much less than the range allowed by the human elbow. In the case of calculating a human elbow angle greater than 90º, we will set the Bioloid elbow external angle to its maximum value, that is to say 90º, and we will modify the arm pitch angle (second servo in the arm chain) in order to compensate as much as possible the difference on the imitation.

The vector shoulder-hand has been calculated from the tracking with the input information of the SR3000 camera. Thus, errors on the SR3000 camera information or in the tracking can appear. The cosine theorem can only be applied if the absolute value of the vector shoulder-hand is lower that the sum of absolute value of the vector arm plus the absolute value of the vector forearm that we have defined, the opposite has no sense. During the mapping, when imitaSoft detects that this condition is not fulfilled, a warning message appears in the imitaSoft GUI indicating that the mapping can not be done for this specific instant of time. In this case the specific arm for which the warning message appears will retain the current position for this instant. Otherwise the mapping continues.

The next angle that we will deduce is the shoulder rotation angle given by the first servo-motor. This angle allows the flexion/extention of the Bioloid arm and we have named it the yaw angle. The yaw angle will be given by the first servo on the arm chain. On Figure 57 we observe comparison between the three angles (yaw, pitch and roll) between an airplane and the Bioloid arm. From these angles, the Bioloid shoulder only allow the firsts two DOF. Roll is not allowed by the servos configuration of Bioloid.

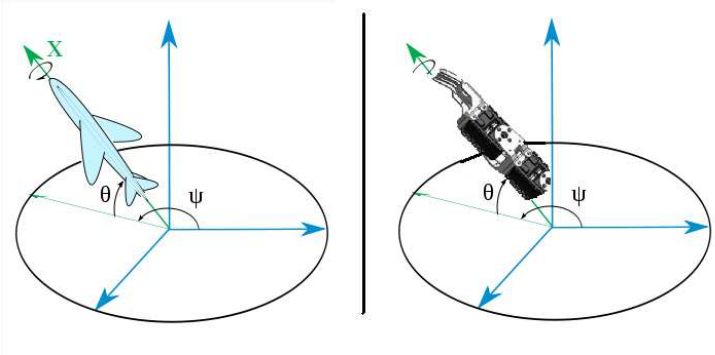

Figure 57: Yaw (Ψ), pitch (θ) and roll (X) angles.

The axis of the first shoulder servo is perpendicular to the sagittal plane (plane **yz**). The second shoulder servo and the elbow servo are always parallel, and are coplanar to the plane **yz**. One of the implications of this servomotor structure is that the arm and the forearm projections on the **yz** plane will always be colinear. These two projections are also collinear to the shoulderHand vector projection on the plane **yz**. On Figure 58 we observe the vectors arm (red), forearm (red) and shoulderHand (green), their axis (black) and their projections (orange) on the sagittal plane. The angle Ψ is the yaw angle.

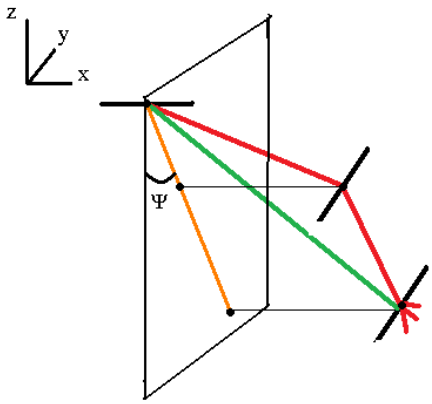

Figure 58: Yaw angle and arm projections on the sagittal plane.

The yaw angle is referenced in regard to the **-z** axis, when the arms are hanging near to the body, and will take values between 0 and Pi, in front of the coronal plane. The yaw angle can be calculated using the **y** component of the vector shoulderHand and the module of the projection of the same vector on the sagittal plane, applying the following trigonometric equation:

$$\Psi_1 = arcsin\left(\frac{-shoulderHand_y}{\sqrt[2]{shoulderHand_y^2 + shoulderHand_z^2}}\right)$$

The minus is used because we are working in front of the coronal plane, and the **y** axis is pointing backwards of this plane. The denominator cannot be 0, thus if obtain 0 when calculating the projection, we will force the value of this projection a really small number.

The yaw angle (Ψ) can be situated on two different quadrants in front of the body. On Figure 59 we have represented the first quadrant (bottom quadrant) in yellow and the second quadrant (top quadrant) in red. If the **z** coordinate of the vector shoulderHand is negative, implies that the projection of this vector is situated on the first quadrant. Thus, the angle Ψ_1 will have a value between 0 and $\pi/2$ and it will be equal to Ψ. However, when the **z** coordinate of the vector shoulderHand is positive, the projection of this vector is situated on the second quadrant. In this case the value of Ψ will be between $\pi/2$ and π radians, and its value is calculated as:

$$\Psi = \pi - \Psi_1$$

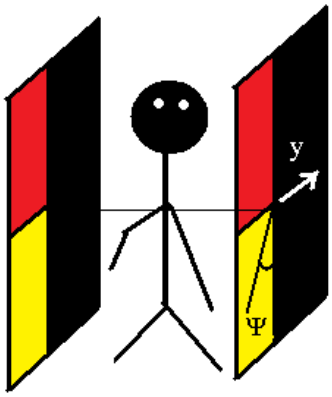

Figure 59: Quadrants where we can found the Ψ angle.

The last angle to be found is the arm pitch angle (θ angle). The arm pitch is achieved thanks to the second servo in the arm chain. This angle cannot be calculated independently of the others because it will depend on the elbow angle. Thus, to calculate the pitch angle we will not only have into account the pitch of the vector **shoulderHand**, but we will also consider the value of the elbow angle.

The first thing to do is to calculate the angle that exists between a vector perpendicular to the sagittal plane, in our case the **x** axis, and the vector **shoulderHand**; we will called this angle λ. To compute λ, we have used the cosine theorem creating a triangle between an **X** vector of 10cm in the x direction, the vector **shoulderHand** and the vector obtained when subtracting **X** to **shoulderHand**, the vector obtained from this subtraction is called **H**. On Figure 60 we observe the human arm and forearm (black), and the triangle formed by the vector **shoulderHand** (green), the vector **X** (red) and the vector **H** (blue).

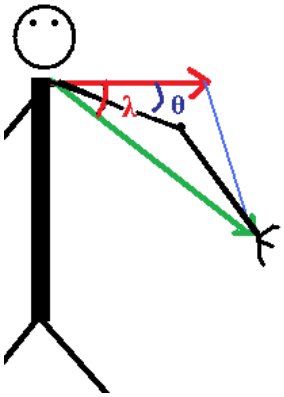

Figure 60: Diagram representing λ and the arm pitch angle.

Having this triangle we can deduce the angle λ by using the magnitudes of the vectors forming the triangle and applying the cosine theorem:

$$\lambda = arccos(\frac{X^2 + shoulderHand^2 - H^2}{2 \cdot X \cdot shoulderHand})$$

The following angle that we have to know in order to compute the pitch, is the angle formed

between the arm vector and the vector shoulderHand. We will call this angle as δ. This angle δ, as λ and the pitch angle, is situated on the plane formed by the vectors **shoulderHand** and **X**. We will call this plane the pitch plane, and it is the plane inside which the Bioloid arm and forearm will move for a given yaw angle. On Figure 61 we represent λ, δ and the pitch angle θ.

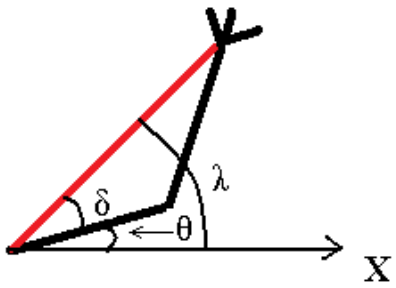

Figure 61: λ, δ and the arm pitch angle.

δ has also been computed using the cosine theorem between the lengths of the human arm vector (a_{arm}), the human forearm vector (b_{fore}) and the shoulderHand vector (c_{SH}).

$$\delta = arcos\left(\frac{c_{SH}^2 + a_{arm}^2 - b_{fore}^2}{2 \cdot a_{arm} \cdot c_{SH}}\right)$$

As imitaSoft only allows the tracking and mapping when the hands are working in front of the coronal plane, only two different cases are possible. On Figure 62 we can observe both cases. The first case is when both, the elbow and the hand, are in front of the coronal plane. The second is when we have the hand in front of the coronal plane and the elbow behind it. In this image the plane represented is the pitch plane, which is the plane inside which the arm and the forearm moves.

λ and δ angles will always be positive as they have been computed using the cosine theorem. Figure 62 shows that the pitch angle (θ) can be computed in both cases by subtracting the δ angle to the λ angle. Having the origin of the x axis at the shoulder this axis divides the pitch plane in two parts, if the hand and the elbow are on the same side of the x axis, we will obtain a positive θ; else if the x axis crosses the forearm vector we will obtain a negative (θ).

$$\theta = \lambda - \delta$$

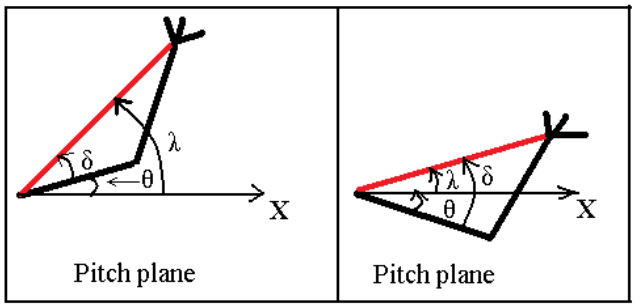

Figure 62: λ, δ and the arm pitch angle for both cases.

The pitch angle calculated until now is calculated regarding the x axis. From this axis the servomotor giving the pitch angle to the Bioloid arm cannot take values higher than $\pi/2$ because of the Bioloid mechanic limitations. Having that, if we obtain a pitch angle higher than $\pi/2$ after our calculations we will have to set its value to its maximum value $\pi/2$ allowed by the mechanics of Bioloid. Modifying this value would totally modify the position of the hand, as this shoulder servomotor is the second in the arm chain, and has a repercussion on the arm and the forearm position. The relative position of the hand regarding the shoulder of the same arm would be modified. However we will minimize this fact by adapting the elbow angle in consequence and we will set the pitch angle to its maximum value of $\pi/2$.

On Figure 63 we observe how the limitation of θ is affecting the hand position (at left). At right we minimize the difference by readjusting the elbow angle. With this modification we will obtain a forearm vector parallel to the forearm direction before to set the pitch angle to $\pi/2$. Thus, the difference in the hand position is reduced and a better quality of goal oriented mapping is achieved.

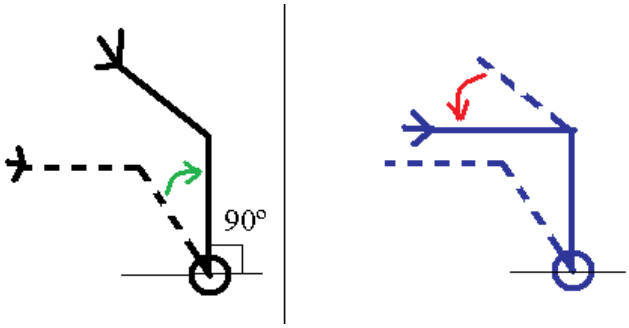

Figure 63: Limitation of θ to $\pi/2$ and readjustment of the elbow angle.

On Figure 63 we see that the angle that we have to add to the previous elbow angle (in red), is equal to the angle that we subtract to θ (in green) in order to limit it to $\pi/2$. Thus, the new elbow external angle in function of the previously calculated elbow angle, when the pitch angle calculated in relation to X was higher than $\pi/2$, will be:

$$elbow = elbow_(previous) + (\theta - \pi/2)$$

And we will force the new θ to the value $\pi/2$:

$$\theta = \pi/2$$

If θ was not greater than $\pi/2$ then the θ calculated previously is kept. Finally, we will compute the pitch angle regarding the sagittal plane using the pitch angle found regarding the x axis. On Figure 64 we appreciate the pitch angle regarding x (θ, in red), and the pitch angle regarding the sagittal plane (θ_s, in green), which is equal to:

$$\theta_s = \pi/2 - \theta$$

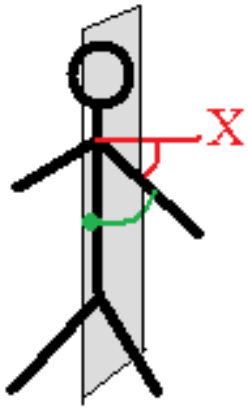

Figure 64: Pitch angle (θ) regarding the coordinate x (in red) and pitch angle (θ_s) regarding the sagittal plane (in green)

The process to deduce the angles both arms is almost equivalent with only some difference in the signs. The symmetry of the body regarding the sagittal plane gives us the opportunity to apply the same reasoning.

To synthesize we can say that the main process to find the Bioloid angles for a given iteration has been to use the hand position regarding its respective shoulder position for an instant of time. Then we have looked for a Bioloid arm configuration allowing to keep this relative hand position regarding the shoulder, having into account the Bioloid mechanical constraints. It is often possible to keep this relative position by adapting the Bioloid joint angles in consequence. When the Bioloid mechanical constraints does not allow to keep the relative position we search alternative positions close to the original position that we were looking for, by modifying the pitch and elbow angles.

Mapping the angles to Bioloid: from degrees to AX-12 values

Using as data the relations between some angles in degrees and the correspondent AX-12 servo positions from Bioloid (with its correpondent ID) that we have seen in section 5.5, we can deduce some simple equations relating every angle expressed in degrees with its correspondent AX-12 value. The relation used has the form: $ax+b=y$; where 'x' is the angle expressed in degrees and 'y' is the correspondent AX-12 value for this servo angle. With two pairs of 'x' and 'y' we can find the constants 'a' and 'b' for a given servomotor.

Once we have the three angles in radians (Ψ, θ_s and elbow), we use a small function to express them in degrees, and then we apply the relations deduced for the corresponding arm, allowing to express these angles in Bioloid AX-12 intelligible positions. For the left arm we have:

$$\Psi_{Bio} = anglesBio[0] = (-3.42 * \Psi + 820);$$

$$\theta_{Bio} = anglesBio[1] = (-3.36 * \theta_s + 815);$$

$$elbow_{Bio} = anglesBio[2] = (3.33 * elbow + 500);$$

And for the right hand we have:

$$\Psi_{Bio} = anglesBio[0] = (3.42 * \Psi + 205);$$

$$\theta_{Bio} = anglesBio[1] = (3.36 * \theta_s + 210);$$

$$elbow_{Bio} = anglesBio[2] = (-3.33 * elbow + 500);$$

For example, for the left arm and for an elbow angle of 90^o we would have

$$elbow_{Bio} = (-3.33 * 90^o + 500) \approx 800,$$

which is coherent with the data from section 5.5 deduced thanks to the Bioloid Motion Editor software.

The values added (820, 815, 500, 205, 210, 500) are the offsets that we have to apply. As we have said, the factors have been found experimentally using different Bioloid arm positions. It is for this reason that we found small differences between the multiplicative factors (3.42, 3.36 and 3.33) but knowing that we are using the same model of servo (AX-12) for each robot joint we would have to found the same values. In our software we have used the average of these values (3.375) as multiplicative factor.

Mapping: The Real Time

Above we have seen how we have done the mapping in order to deduce the Bioloid angles for a particular instant of time. What we have done is to generalize this process by applying it every time that we have new data coming from the tracking. As we have seen, the tracking obtains new information about the vectors shoulder-hand every 70ms. Thus, every 70ms the mapping algorithm is feed with this new information which allows to find the three angles for every iteration. Moreover, we use the angles found for two consecutive iterations separated by 70ms in order to deduce the velocities for each servo between a given position and the next one. In this way, having a good computing capacity, we can create continuous and fluid Bioloid arm movements. To calculate the servo velocities we use the following equation that we apply to the angles of the three different servos (arm yaw Ψ, arm pitch θ_s and elbow angle):

$$\frac{angle_{goal} - angle_{current}}{0.07s}$$

In order to translate the velocities from degrees per second to intelligible Bioloid values (AX-12 servos values) we have performed an experiment on section 6.5.3 to deduce the coefficients allowing this translation. The relation found is:

$$Velocity_{\ in\ degrees\ per\ second} = Velocity_{\ in\ Bioloid\ units} \cdot 0,6$$

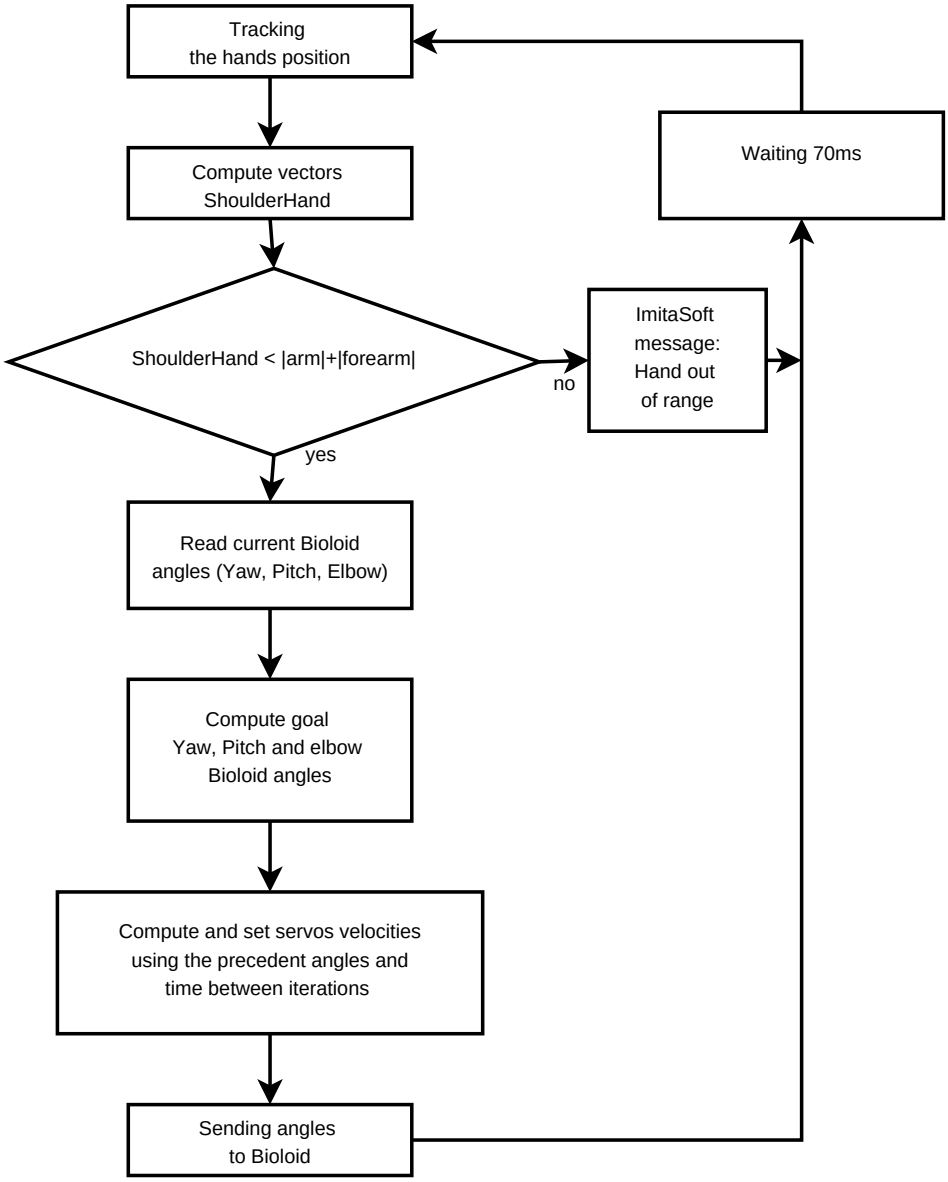

Figure 65: Algorithm diagram of tracking + mapping

Figure 65 is the block diagram showing the main structure of the imitaSoft algorithm. As we see, we use the tracking presented on subsection 6.2.1, we compute the shoulderHand vectors thanks to the shoulder coordinates and the hand coordinates for each iteration, we get the current positions of the servos and we calculate the goal positions to be attained thanks to the mapping of angles presented in this chapter. Doing the subtraction of these pairs of angles and dividing by the time between iterations (70ms), we obtain the servos velocities for the current iteration. After setting the velocities in the Bioloid notation we send the goal angles to Bioloid, and the we restart a new iteration after 70ms. The same process is used to perform the mapping of one arm or of both arms. When working with both arms we perform first the mapping of one arm and just after the mapping of the other arm by using the tracking information from the same iteration.

This normal process can be modified if a hand is not found inside its logical range, that is

to say, if a shoulderHand vector longer than the length of the arm plus the length of the forearm is found. In this case the process restarts after 70ms without modify the servos angles for the current iteration. A warning message indicating the arm which is out of the range appears on the imitaSoft screen indicating this fact. When the hands reenter the logical range the mapping continues with normality.

6.3 Imitation in deferred: mapping using VooDoo XML sequences

In this subsection we present the deferred tracking using the extracted information from the VooDoo software described on section 5.3. This software allows to map the human arm movements to a human computer model and from there we can will be able to map the computer model movements to Bioloid robot movements.

The first thing that we need to know is the kind of data that we have as starting point, in order to deduce the angles for the three servomotors of the Bioloid arm, later on. In our case these data have been obtained by two different ways. As we have seen on subsection 6.2.1, for the real time imitation we have used as input information the data retrieved from the Swiss Ranger camera, processing its images and carrying out a shoulder-hand tracking. In this part, we will perform a deferred imitation using a graphic interface already designed to carry out a human tracking and allowing to model a 3D human body. This program that we have presented on section 66 is called VooDoo. VooDoo represents the body limbs using a human model consisting of ten cylinders. One of them is used to represent the torso, another represents the head, four of them are for the legs and lower legs, and four more for the arms and forearms. The VooDoo human model made of cylinders is displayed on Figure 66. We have to take into account that the VooDoo computer model of the human has 9 DOF per arm. Our goal will be to map the arm movements from the 6 DOF from VooDoo arm to the 3DOF of the Bioloid arm. The mapping done in order to do that will also be goal oriented and what we will proceed by finding the vectors *shoulerHand* from the VooDoo model at each instant of time and finding the 3 angles of the Bioloid arm from the information given by the vector *shoulderHand* and the VooDoo arm and forearm lengths.

Figure 66: VooDoo model with the body limbs made of cylinders.

VooDoo has its information organized in the form of sequences of XML documents. On Figure 67 we observe a set of XML files for a given sequence where the model pulls back a hand. VooDoo uses these files to reproduce model movements in deferred after having been recorded previously. Each single XML file represents a given position (a frame) and contains the rotation matrices, used to position the model limbs regarding the contiguous limbs, for a given instant of time. A time stamp present on each XML indicates the relative instant of time at which a given position has been recorded inside the whole sequence (68). The frame sampling is done every few milliseconds (between 20ms and 70ms approximately) in order to give us the impression of a continuous model movement recording. Every complete model movement has its own set of

XML files describing the entire movement with a finite number of discrete positions.

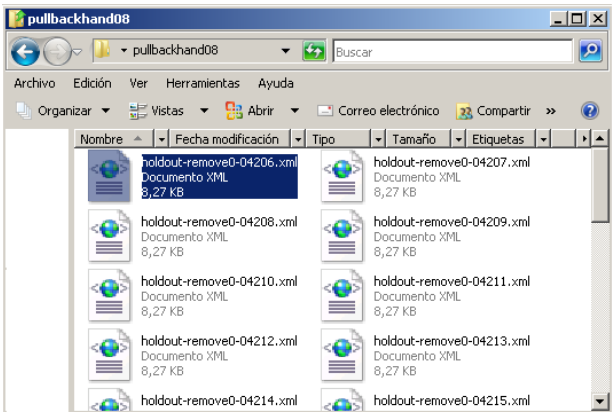

Figure 67: XML VooDoo files forming a given VooDoo sequence in which the computer model pulls back its hand.

```
- <VDBodyModel modeltype="0" timestamp_s="1165222121" timestamp_us="124763">
  - <GlobalMatrix>
      <Row0 Col0="0.719137" Col1="-0.693683" Col2="0.040578" Col3="75.925955" />
      <Row1 Col0="0.044972" Col1="-0.011811" Col2="-0.998918" Col3="367.571312" />
      <Row2 Col0="0.693411" Col1="0.720184" Col2="0.022703" Col3="2800.994665" />
      <Row3 Col0="0.000000" Col1="0.000000" Col2="0.000000" Col3="1.000000" />
    </GlobalMatrix>
  - <Limb type="0" children="5">
      <Dimension rba="150.480011" rbb="83.599998" rta="150.480011" rtb="83.599998"
    + <nextChild index="0">
    + <nextChild index="1">
    + <nextChild index="2">
    + <nextChild index="3">
    + <nextChild index="4">
    </Limb>
  </VDBodyModel>
```

Figure 68: Time stamp present on each XML file.

Thanks to the VooDoo information in form of rotation matrices between human limbs, we are able to deduce the set of vectors going from the shoulders to the hands for a given deferred sequence previously recorded using VooDoo. These vectors that have been called *shoulderHand vectors* are used afterwards to find out the Bioloid servomotors angles and velocities by applying the same mapping algorithm used for the real time imitation, that we have seen on subsection 6.2.2. As we have done on subsection 6.2.2, using the projection of the arm vectors on different plans and applying different trigonometric relations, we have computed the Bioloid arm angles with a very low computational cost, in a quite intuitive way.

On Figure 69 we present a schematic image showing the different limbs forming the VooDoo arms and the systems of coordinates attached to each limb used to represent the movements. To be brief and clear, we will study here only the left arm. We will use the number of each limb as sub-index of the vectors expressed in a given coordinates system.

- The torso (will be our base: Limb number 0)
- The arm (Limb number 1)
- The forearm (Limb number 2)

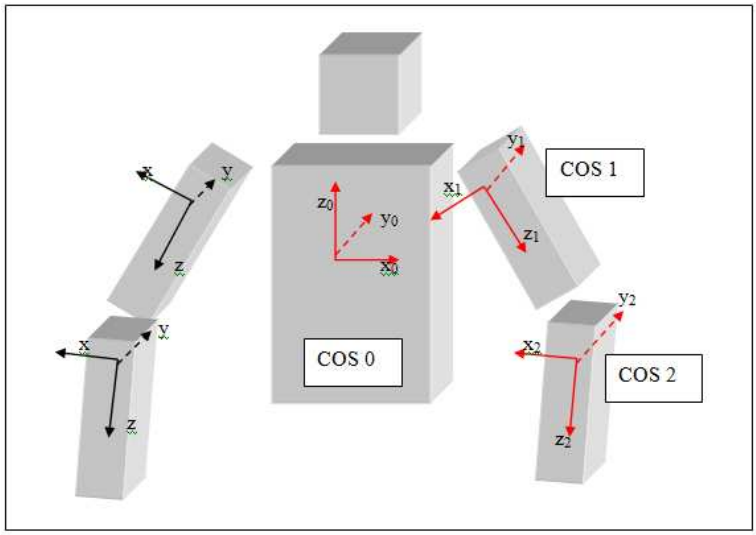

Figure 69: VooDoo limbs with the coordinates systems used by the program.

Each XML VooDoo file store the transformation matrices used to go from one parent limb to a child limb (for example: from the torso to the arm). These matrices contain information about the rotations and the translations necessary to go from a limb to another. From those matrices of 4x4 components we will only be interested in the first 3x3 components which are the components describing the rotation necessary to go from the coordinates system of one limb to the coordinates system of the next limb.

VooDoo uses a total of four transformation matrices for each arm. Two of them are invariant and are used to indicate a given standard position of the arms; we will call them joint matrices. The others vary over time, indicating the position of the limbs in relation to the standard position defined by the joint matrices; we will call them, movement matrices. Each articulation (shoulder and elbow) is formed by a joint matrix plus a movement matrix. In order to obtain the total transformation matrix allowing us to go from a limb to the following, what we do is to multiply the joint matrix by the movement matrix defined between two consecutive limbs. In the case of the elbow, the joint matrix is defined as the identity matrix inside VooDoo; this means that its standard position of the forearm is to be aligned with the arm. Moreover, in this case, the total transformation matrix between arm and forearm is equal to the movement matrix between them, because the identity matrix is the identity element for the matrix multiplication. In contrast the shoulders joint matrices are different to the identity. These matrices are presented hereafter:

Right shoulder joint matrix (RSJM):

$$RSJM = \begin{pmatrix} -0.866 & 0 & -0.5 \\ 0 & 1 & 0 \\ 0.5 & 0 & -0.8666 \end{pmatrix}$$

Left shoulder joint (LSJM):

$$RSJM = \begin{pmatrix} -0.866 & 0 & 0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & -0.8666 \end{pmatrix}$$

The precedent transformation matrices are used by VooDoo like shoulder joint matrices and will be useful to find the hand position in relation to shoulder.

Other important information is the length of the arm and the length of the forearm in order to define the arm and forearm vectors. The VooDoo models normally use an arm length equal to the forearm length because in fact those two values are quite close in the reality and as we will see later on, this fact eases some of the calculations when studying the geometry of the arm configurations.

Hereafter, we will give an example of the left hand coordinates calculation in relation to the left shoulder. We will express the vector in the torso coordinates system. This shoulderHand vectors will be necessary afterwards to deduce the Bioloid servos angles applying the same mapping reasoning that we have used for the real time imitation on section 6.2.2.

The input data used to deduce a shoulderHand vector are the VooDoo transformation matrices for the corresponding frame. The transformation matrices for this specific frame describe the VooDoo model position for the human in a given moment referenced by a time stamp in microseconds. A given XML from a VooDoo sequence give us the following joint and movement matrices for the left shoulder. As said before, the shoulder joint matrix (Figure 70) is always the same for all the VooDoo sequences and for all the XMLs presents on a given sequence; it represents a given standard position between the body joints. The shoulder movement matrix represented on Figure 71 is specific of a given instant of time and presents the relative position between the torso and the arm limb for each different time stamp.

```
−<JointMatrix>
   <Row0 Col0="-0.866025" Col1="0.000000" Col2="0.500000" Col3="150.480004"/>
   <Row1 Col0="0.000000" Col1="1.000000" Col2="0.000000" Col3="0.000000"/>
   <Row2 Col0="-0.500000" Col1="0.000000" Col2="-0.866025" Col3="589.380014"/>
   <Row3 Col0="0.000000" Col1="0.000000" Col2="0.000000" Col3="1.000000"/>
 </JointMatrix>
```

Figure 70: Left shoulder joint matrix from a specific VooDoo XML.

```
−<ChildMatrix>
   <Row0 Col0="0.743722" Col1="-0.641370" Col2="0.188474" Col3="-7.482903"/>
   <Row1 Col0="0.645111" Col1="0.762505" Col2="0.049158" Col3="1.712902"/>
   <Row2 Col0="-0.175239" Col1="0.085027" Col2="0.980848" Col3="19.742119"/>
   <Row3 Col0="0.000000" Col1="0.000000" Col2="0.000000" Col3="1.000000"/>
 </ChildMatrix>
```

Figure 71: Left shoulder movement matrix from a specific VooDoo XML.

The multiplication of those two matrices gives us the matrix allowing us to change from the left arm coordinates base to the torso coordinates (TA):

$$TA = \begin{pmatrix} -0.7318 & 0.5967 & -0.3272 \\ 0.6450 & 0.7620 & 0.0490 \\ -0.2205 & 0.2464 & -0.9427 \end{pmatrix}$$

The joint elbow rotation and movement submatrices for the left elbow are presented on figures 72 and 73. As said before the elbow joint matrix used by VooDoo is the identity matrix.

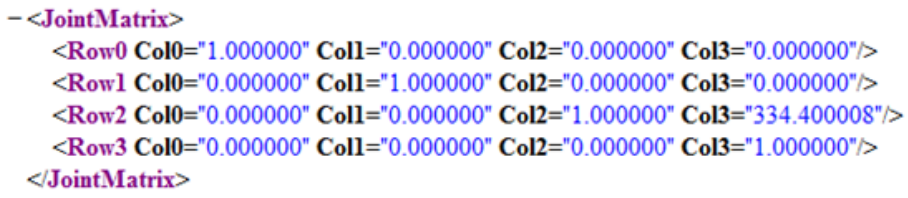

Figure 72: Left elbow joint matrix from a specific VooDoo XML.

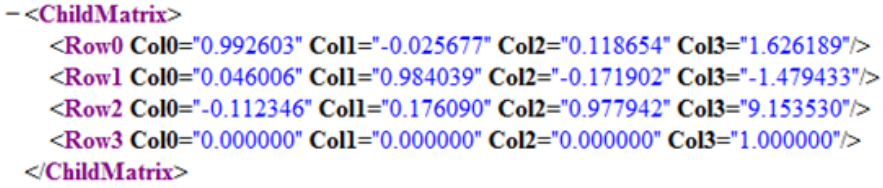

Figure 73: Left elbow movement matrix from a specific VooDoo XML.

In this case the movement rotation matrix directly gives us the transformation matrix between the left arm and the left forearm (AFa) because the joint matrix is the identity:

$$RSJM = \begin{pmatrix} -0.866 & 0 & -0.5 \\ 0 & 1 & 0 \\ 0.5 & 0 & -0.8666 \end{pmatrix}$$

Left shoulder joint (LSJM):

$$AFa = \begin{pmatrix} 0.9926 & -0.0256 & 0.1186 \\ 0.0460 & 0.9840 & -0.01719 \\ -0.1123 & 0.1761 & 0.9779 \end{pmatrix}$$

Finally, when we multiply the transformation matrix allowing going from the torso to the arm (TA), by the transformation matrix allowing going from the arm to the forearm (AFa), we obtain the total transformation matrix allowing us to pass from the torso to the forearm (TFa):

$$TFa = TA \cdot AFa = \begin{pmatrix} -0.7357 & 0.6635 & 0.1306 \\ 0.6698 & 0.7419 & -0.0066 \\ -0.1016 & 0.0821 & -0.9903 \end{pmatrix}$$

Thanks to the precedent matrix we will be able to express in the base of the torso, the vectors that were previously expressed in the forearm base.

As indicated on Figure 69, the arm vector is oriented following the z1 axis and the forearm vector is oriented following the z2 axis. Thus, knowing that VooDoo use a length of 335mm for the arm and the forearm, these two vectors expressed in their own base are presented here:

Arm vector (vArm) expressed using the own arm coordinates system (vForeArm):

$$\mathbf{vArm} = TA \cdot AFa = \begin{pmatrix} 0 \\ 0 \\ 335 \end{pmatrix}_1$$

Arm vector (vArm) expressed using the own forearm coordinates system:

$$\mathbf{vForearm} = TA \cdot AFa = \begin{pmatrix} 0 \\ 0 \\ 335 \end{pmatrix}_2$$

Using the matrices that we have calculated in this section, we are already able to express the precedent vectors in the torso coordinates base, and so, to find the shoulder-hand vector, in the given instant, also expressed in the torso base.

Vector arm expressed in the torso system of coordinates:

$$\mathbf{vArm}_0 = TA \cdot \mathbf{vArm}_1 = \begin{pmatrix} -0.7318 & 0.5967 & -0.3272 \\ 0.6450 & 0.7620 & 0.0490 \\ -0.2205 & 0.2464 & -0.9427 \end{pmatrix}_{0\to1} \cdot \begin{pmatrix} 0 \\ 0 \\ 335 \end{pmatrix}_1 = \begin{pmatrix} 109.61 \\ 16.42 \\ -315.80 \end{pmatrix}_0$$

Vector forearm expressed in the torso system of coordinates:

$$\mathbf{vForeArm}_0 = TA \cdot \mathbf{vForeArm}_2 = \begin{pmatrix} -0.7357 & 0.6635 & 0.1306 \\ 0.6698 & 0.7419 & -0.0066 \\ -0.1016 & 0.0821 & -0.9903 \end{pmatrix}_{0\to2} \cdot \begin{pmatrix} 0 \\ 0 \\ 335 \end{pmatrix}_2 = \begin{pmatrix} 43.75 \\ -2.203 \\ -331.77 \end{pmatrix}_0$$

Thanks to the precedent vectors we can deduce the components of the vector shoulderHand in the torso base (base 0), for this particular instant of time. We do so computing the addition of the arm and forearm vectors, both expressed in the base 0

$$\mathbf{shoulderHand}_0 = \mathbf{vArm}_0 + \mathbf{vForearm}_0 = \begin{pmatrix} 109.61 \\ 16.42 \\ -315.80 \end{pmatrix}_0 + \begin{pmatrix} 43.75 \\ -2.203 \\ -331.77 \end{pmatrix}_0 = \begin{pmatrix} 153.36 \\ 14.21 \\ -647.57 \end{pmatrix}_0$$

The information given by this vector, which is specific for each frame, is used to deduce the angles that we are looking for by applying the same mapping process explained on the real time imitation (section 6.2.2).

To deduce the Bioloid velocities, we use the triplets of angles for two consecutive instants of time (one angle for each of the three servos forming a Bioloid arm) as we have done on the real time imitation. Doing the subtraction between both triplets of angles and dividing by the elapsed time between the consecutive XML files, which is deduced thanks to their time stamps, we obtain the velocities for each of the Bioloid servos to go between a given position to the next one. We apply the same process for each loop cycle.

Finally, we summarize the key steps that are present on the deferred imitaSoft mapping using as input information the VooDoo sequences. In a first place, as we can see on Figure 74, imitaSoft allow us to choose a given VooDoo XML sequence representing a set of movements of the VooDoo computer model. Afterwards the imitation begins following the diagram represented in Figure 75. On this figure we can see the evolution of the algorithm, from choosing a given VooDoo sequence, to deducing the Bioloid angles for each time stamp until the VooDoo XML sequence finishes.

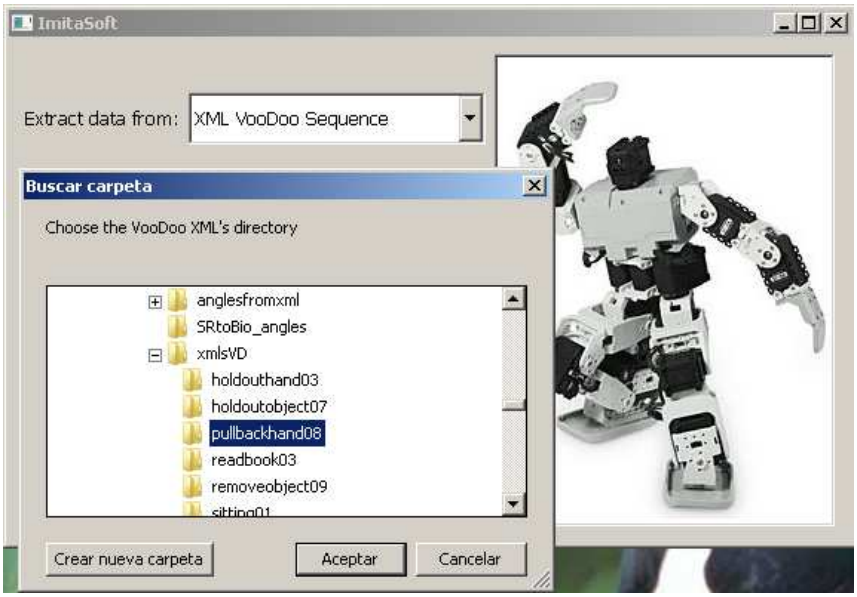

Figure 74: Menu window allowing to choose a given VooDoo sequence.

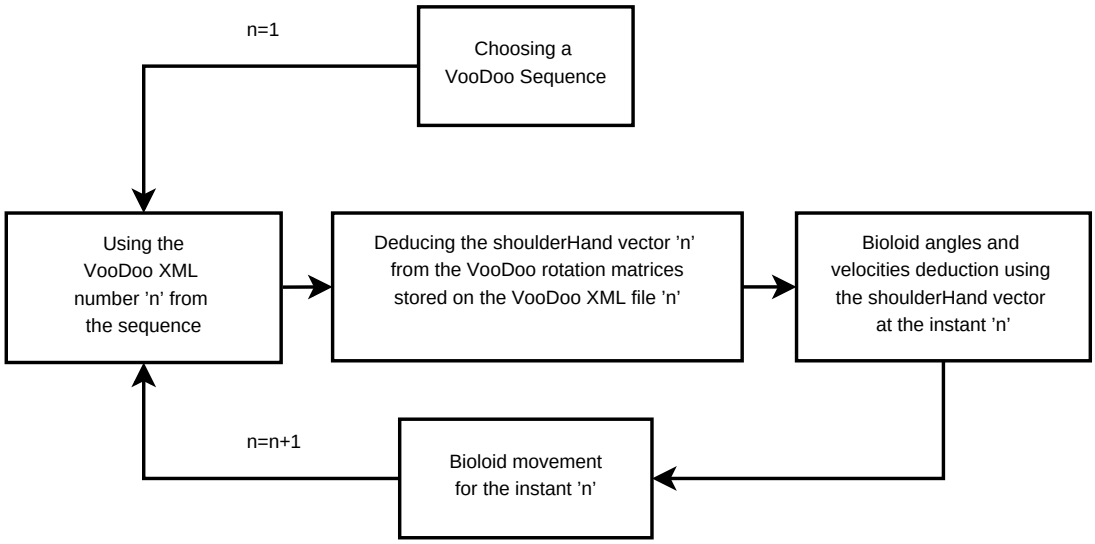

Figure 75: Diagram representing the mapping using the VooDoo information.

6.4 Learned actions: Angles and Velocities Files

Another important functionality which incorporates imitaSoft is the ability to "learn" new actions by imitation. ImitaSoft is able to memorize Bioloid actions that have been performed previously by imitation and reproduce them with the robot whenever we desire.

Every time that we perform an imitation using imitaSoft, the program shows the option to memorize the Bioloid movements that are done when performing a given imitation for the first time. If we enable this option, the Bioloid servomotor sequence of angles and velocities is stored and Bioloid will be able to use this information in the future in order to reproduce a given imitation. On Figure 76 we apreciate the option *'Angles and velocities files'* which allows Bioloid to perform the reproduction of previously learned actions.

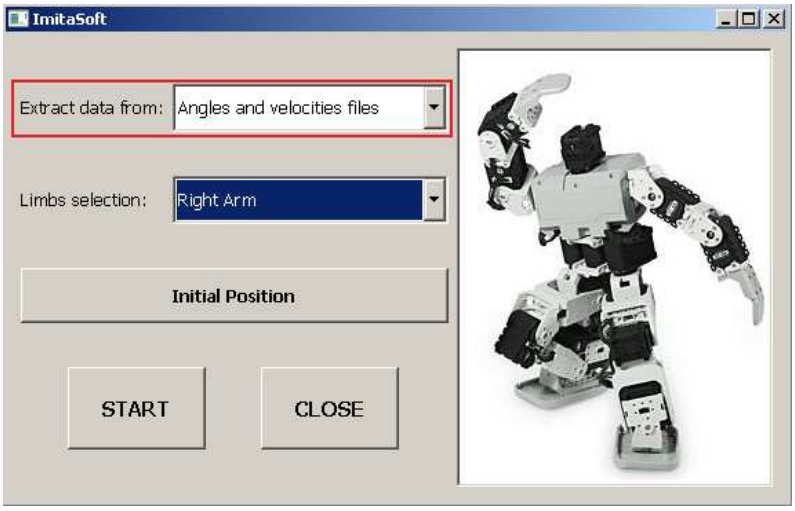

Figure 76: ImitaSoft option to reproduce previous learned actions with Bioloid.

This learning can be enabled both using the real time human imitation functionality (6.2) and using the imitation in deferred functionality (6.3). This sequences of velocities and positions are stored in 'txt' files. Once the imitation has been performed 5 'txt' files are created for each hand. The first file stores the triplets of angles in function of the time for the 3 servos of the arm; the second file stores the triplets of velocities in function of the time for the 3 servos of the arm; and a third file stores the interval of time that is elapsed between consecutive triplet of angles, which is of the order of magnitude of some tens of milliseconds. Two more files are stored in order to have the angles and the velocities in the Bioloid notation. This set of informations is combined afterwards when we want to reproduce an imitation that has been performed previously. On Figure 77 we observe the way in which the triplets of angles for the right Bioloid arm have been stored after the learning of a given VooDoo sequence. The triplets of angles in function of the time are stored in degrees (left) and in the Bioloid's own notation (left). The first angle of the triplet corresponds to the first shoulder servo in the arm chain, the second angle corresponds to the second servo in the arm chain and the third servo corresponds to the elbow angle which is the third servo in the arm chain.

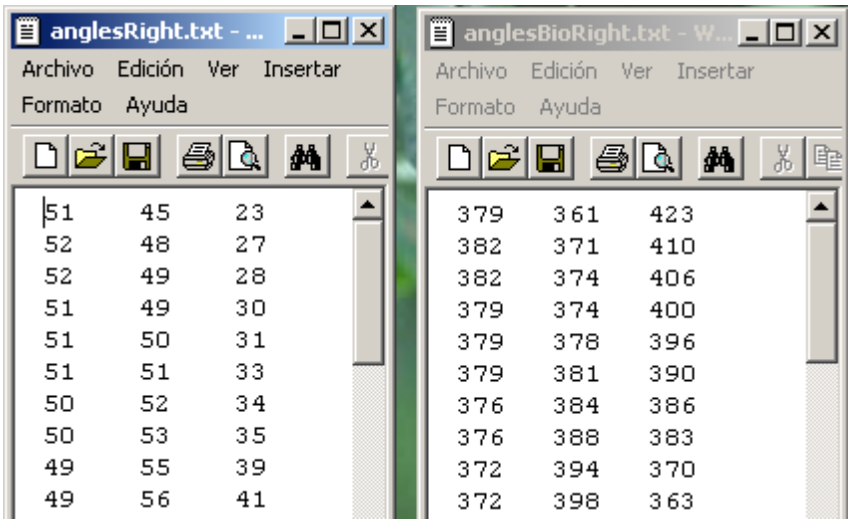

Figure 77: Triplets of angles stored in degrees and in Bioloid notation.

One important question is why we can consider that imitaSoft allows Bioloid 'learning' new actions by imitation. In fact, the first time that an imitation is performed, imitaSoft performs a mapping in order to make a link between the human movements (or the VooDoo human model movements) and the Bioloid movements. But once this first imitation has been performed, the files stored by imitaSoft are recorded in the Bioloid angles notation. Thus, once the learning has been done, there is no need to perform the mapping from the beginning, instead the robot only reproduces the actions that it has performed previously during the original imitation. ImitaSoft has allowed Bioloid to learn new actions because it has internalized these actions in the Bioloid's own notation.

As we have seen above, thanks to the data storage allowed for imitaSoft when performing a real time imitation or a deferred imitation, we can later on reproduce the same actions that the robot has imitated for the first time. ImitaSoft has allowed to learn the actions and from now on we can teach new actions to Bioloid without touching at any code, only by using imitaSoft software.

6.5 Experiments

6.5.1 Experiments 1 and 2: Choice of a SR3000 integration time and minimization of the error in depth measurements

Introduction

The following two experiments have been designed in order to minimize the random errors and to detect possible systematic errors in depth measurements given by the SR3000 camera. The objective is to give solutions in order to minimize these errors and to improve the imitation in real time. Firstly we will choose an appropriate SR3000 integration time reducing the random errors on measurements inside the tracking area. Secondly we will identify and reduce the systematic errors by approximating the error function and compensating these errors on the SR3000 measurements with the function found. The imitaSoft software will be adapted in consequence implementing the chosen solutions.

Experiment 1

In this experiment we will study the depth information obtained using the SR3000 camera in order to determine to what degree it is reliable. The main objective is to find an adequate SR3000 integration time for the range of distances where the performer must be positioned during the robot control when he is using imitaSoft.

To perform this study we have used a large number of images of a box with dimensions 50x25x25 cm (Figure 78). This box has been photographed at different distances over the range going from 0.80 meters to 3.2 meters, taking the camera sensor as the origin for the distance measurements. From these images we have calculated the average depth distance and the distance standard deviation from the pixels in the front face of the box. With all this information we will be able to choose an integration time helping to reduce the standard deviation for the range of distances inside the tracking area.

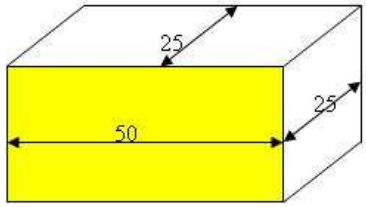

Figure 78: Box used to determine depth precision in SR3000 distance measurements.

Given that one of the most important aspects of the project is the imitation in real time, we will be interested into use a low integration time allowing a time reduction between frames during the tracking. It is important to reduce this inter-frame time because a local hand tracking needs similar positions of the hands between one frame and the next one, otherwise the hand could go out from the local tracking window, and thus, lose the track.

The total range of the camera going from few centimeters to 7.5 meters and the integration times going from 1 to 50ms, we will try to find an adequate integration time useful for the range between 0.80 meters to 3.2 meters. The tracking area where the performer must be placed when using imitaSoft will be a subrange of this range of distances, but study a slightly larger space will give more flexibility and comfort to the performer positioned in the camera field of vision.

On Figures 79, 80, 81 and 82, we can see a sample of the kind of images taken by the

Figure 79: Intensity images for different distances.

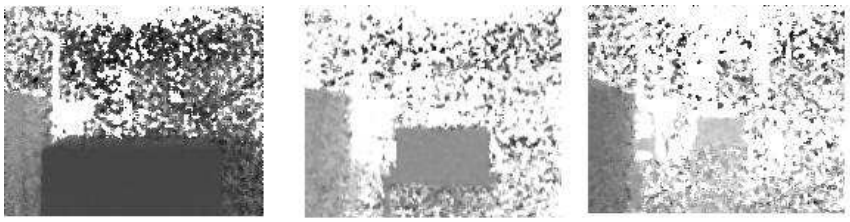

Figure 80: Depth images for different distances and a 2ms integration time.

Figure 81: Depth images for different distances and a 18ms integration time.

Figure 82: Depth images for different distances and a 50ms integration time.

SR3000 camera. On Figure 79 we present the intensity images of the box positioned in different distances. On Figures 80, 81 and 82 we have presented the depth images of the box. The box has been placed on the top of a small cabinet in order to center it on the image.

These images have been taken at three different depth distances which are: 0.80m, 1.60m and 3.20m. The integration times studied here are 2ms, 18ms and 50ms. The depth images have been taken in such a way that we observe the closest points to the camera in darker colors and the farther in lighter colors.

The first thing that we observe when we take a general look to the images is that some zones that appears far in one image, appears close in another image. We can see this fact when

we compare the images form the box at 0.80 meters with the other images. In these images we observe the wall at the end of the room which seems to be really close for the images where the box is at 0.80cm, and actually, it must be the opposite. This is a noisy effect that can be reduced and appears when there are windows or other reflective objects in the scene. We will study it in more detail on 6.5.2.

When we focus on the box, we observe that for an integration time of 2ms, we have almost no noise on the box when it is close to the camera (0.80m), but when we put the box farther we begin to observe that the surface of the box appears wrinkled to the camera. In fact the integration time helps giving more or less illumination to the scene, and we have to adjust it to our range. For the integration times of 18ms and 50ms we observe that the surface of the box seems to be quite smooth. This means that the depth calculated for each pixel in the surface of the box is approximately the same for these integration times.

After the qualitative study above, we will perform a quantitative study. For it, we will use the information from the depth images in order to obtain the standard deviations in depth from the pixels situated on the front face of the box. We will retain the integration times for which we obtain smaller standard deviations in distance, indicating that the camera sensor actually senses the front face of the box as a smooth surface.

In Table 3 we present the depth standard deviations (in meters) for different distances and for different integration times. They have been calculated taking 250 pixels for a box at the distance of 3.05m, to 2600 pixels for a box at the distance of 0.65m. All those pixels were situated at the front face of the box, visible by the SR3000 camera. When the box is close to the camera (e.g. for 0.65m) more pixels of the image are occupied by the box thus we can use more pixels for the standard deviation calculations; in the other hand, when the box is far from the camera we have the opposite effect because the box occupies a smaller angle.

		Distances [m]						
		0.65	**1.05**	**1.45**	**1.85**	**2.25**	**2.65**	**3.05**
Integration Times [ms]	**2**	0.005	0.011	0.017	0.023	0.041	0.044	0.087
	6	0.003	0.008	0.007	0.012	0.014	0.021	0.023
	10	0.004	0.008	0.005	0.009	0.009	0.010	0.017
	14	0.003	0.007	0.004	0.007	0.007	0.009	0.011
	18	0.002	0.006	0.004	0.006	0.006	0.007	0.010
	22	0.003	0.006	0.004	0.005	0.006	0.005	0.009
	26	0.005	0.008	0.004	0.005	0.005	0.006	0.008
	30	0.010	0.011	0.005	0.005	0.004	0.005	0.007
	34	0.014	0.013	0.005	0.006	0.005	0.004	0.007
	38	0.015	0.014	0.006	0.006	0.005	0.004	0.007
	42	0.014	0.014	0.006	0.006	0.005	0.004	0.007
	46	0.015	0.014	0.006	0.006	0.005	0.005	0.007
	50	0.017	0.015	0.006	0.006	0.004	0.004	0.006

Table 3: Standard deviations (in meters) in function of the integration time and the distance to the SR3000 camera

On Figure 83 the depth standard deviations in function of the integration time is displayed for different distances. For low integration times we obtain higher standard deviations when the box is far from the camera; e.g for a distance of 3.05m and for an integration time of 10ms we obtain a standard deviation around 1.7cm. On the other hand for these low integration times we obtain low standard deviations when the box is close to the camera; e.g for a distance of

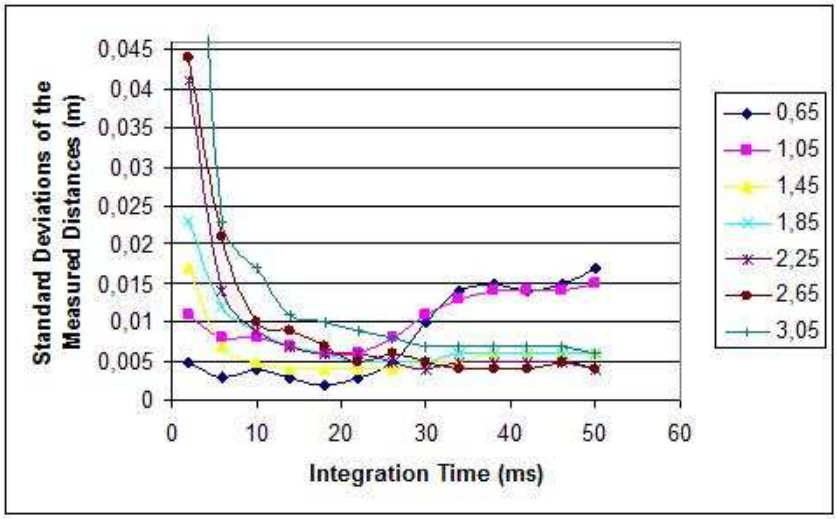

Figure 83: Standard deviations in function of the SR3000 integration times for different distances (in meters).

0.65m and an integration time of 10ms we obtain a standard deviation around 5mm.

When we work with high integration times (around 40 or 50), we get the inverse effect, that is to say, the standard deviations are reduced for longer distances and vice versa. This effect is in accordance with the expected effect because for long distances we need a higher integration times in order to obtain more intensity about the scene, while for lower distances if we have an integration time too high, the photography can be saturated because to much intensity of radiation will be reflected by the scene.

With the information on Figure 83 we already see that the range of integration times for which the standard deviations are lower goes approximately from 17 to 26ms. For this range the standard deviations are smaller than 1cm. As we are looking for a low integration time we have chosen an integration time of 18ms for imitaSoft. With this integration time, imitaSoft will still have 52ms to make the calculations if we work with a frame rate of 14 frames per second (70ms between frames).

A different kind of data has been extracted from the taken depth images. In Figure 84 and 85 we present the standard deviation for a given pixel situated in 15 consecutive SR3000 frames. These pixel is situated in the surface of a white box from which we know its real distance to the camera. We do the same study for 36 different pixels in the surface of the box.

As said above, in Figures 84 and 85, instead of working with pixels of the box in the same frame, we work with the evolution of 36 pixels over the time and each of these 36 pixels is studied separately. The research has been done for two different integration times: 10ms and 18ms.

As shown in Figures 84 and 85 the standard deviations is also bigger over the time for longer distances. We appreciate how standard deviations are always smaller in our range when using an integration time of 18ms compared to an integration time of 10ms. We also appreciate that for 18ms the standard deviation is always smaller than 1.5cm, and for distances shorter than 2.65m the standard deviations the standard deviations are smaller than 1cm. These results are quite similar to those obtained previously when studying the standard deviations of box pixels in the same frame.

As we want to perform an arm mapping, having that the arm and forearm lengths are of

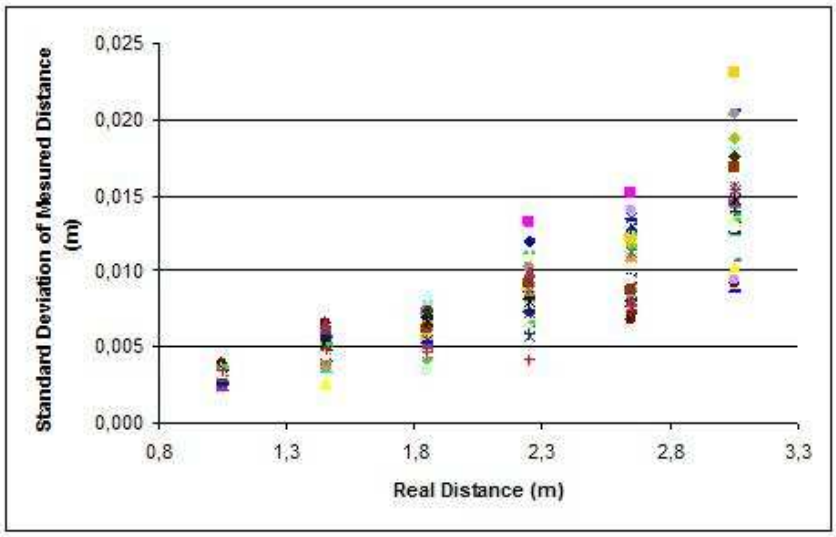

Figure 84: Standard deviations for 36 different pixels taken in 15 consecutive frames for different distances (integration time: 10ms).

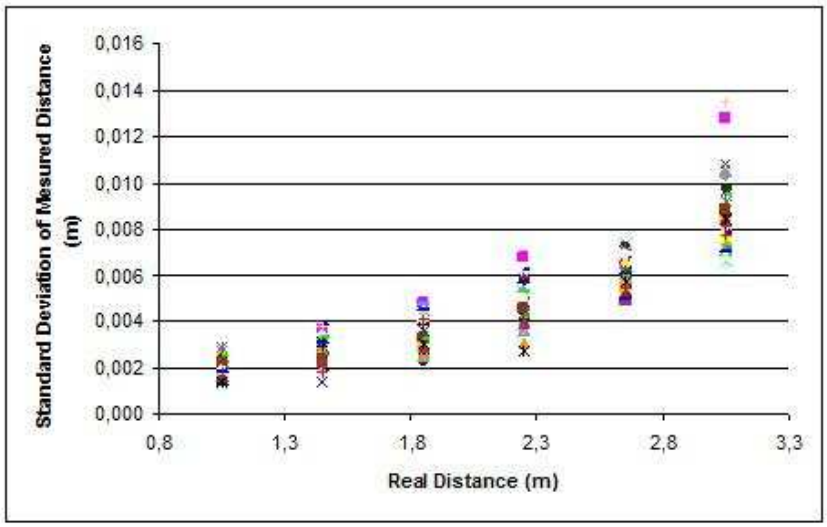

Figure 85: Standard deviations for 36 different pixels taken in 15 consecutive frames for different distances (integration time: 18ms).

the order of 30cm and having that the maximum standard deviation is 1.4cm, we can conclude that this length being only a 5 percent of 30cm would has to allow to perform the tracking and the hand coordinates extraction in sufficiently good conditions to allow the arm mapping.

It is important to remember that we are looking for the vector joining the shoulder and the hand of the same arm in order to extract the useful information to perform the goal oriented mapping. This vector not always has a 'z' coordinate (depth) longer than 30cm, thus the relative error in depth can be higher when the 'z' coordinate is shorter. However, the standard deviations found are quite acceptable for our objectives, even if this coordinate is short.

The results of the experiment have shown that an integration time between 17 and 26ms

would minimize the depth standard deviation inside our tracking range. Knowing that we need a low integration time in order to do not affect the real time tracking and mapping, we have chosen a SR3000 integration time of 18ms to be used by imitaSoft, this integration time will help to improve the precision of the SR3000 coordinates calculation inside the tracking area.

Experiment 2

For this experiment we have used the same scenario used in Experiment 1, that is to say, we have studied the same images that we took from the white box of 50x25x25cm but with a different goal. This time the objective has been to compensate systematic errors produced by the camera on depth measures.

For this study we have used a large number of images of a white box having dimensions of 50x25x25 cm (Figure 1). This box has been placed at different distances over the range going from 0.80 meters to 3.2 meters taking as reference the camera sensor and has been photographed with the SR3000. From those images we have calculated the average depth distance and the standard deviation on depth distance from the pixels in the front face of the box using the information given by the camera.

For this experiment we have used images taken for three different integration times: 10ms, 18ms and 30ms. We have chosen 10ms because it has been the integration time that we have first used in the software until applying the correction done thanks to the results given by the Experiment 1. Secondly, we have chosen 18ms because it has been the integration time retained after having performed the Experiment 1, this is the integration time that we will use in our human to robot mapping software. Finally we have chosen a third integration time in order to know if the average error in the depth measures was approximately independent of the integration time.

On Table 4 we present the results of calculating the average in depth pixel distances for the front face of a white box for three different integration times. These averages have been calculated using a frame for each distance and integration time and using at least 200 pixels situated on the front face of the white box, in such a way that we can obtain the average distance from the camera to the white box. The real distance is indicated in the first column of the table.

In Figure 86 we represent the SR3000 depth measurements in function of the real depth for two different integration times. The difference between real and measured depth is hardly noticeable in this figure because the scale is in meters and the difference is only few centimeters. This means that the measurements given by the camera are quite precise.

In contrast with Figure 86, in Figure 87 we can clearly see that there exists a difference of some centimeters between the real distance and the measured distance. In this figure we present the difference between the real distance and the measured distance in function of the measured distance. This difference is given for the three studied integration times, and we observe that this error follows approximately the same function for the three integration times.

Having the relation

$$Error = (Real\ distance) - (SR3000\ Measured\ distance)$$

we deduce that

$$Real\ distance = (Error) + (SR3000\ Measured\ distance)$$

Real distance [m]	Measured distance using SR3000 camera for:		
	Int. Time: 10ms	Int. Time: 18ms	Int. Time: 30ms
0.65	0.628	0.648	0.654
0.85	0.849	0.849	0.859
1.05	1.052	1.065	1.068
1.25	1.271	1.280	1.286
1.45	1.482	1.498	1.495
1.65	1.699	1.708	1.707
1.85	1.894	1.896	1.889
2.05	2.087	2.091	2.088
2.25	2.273	2.269	2.265
2.45	2.447	2.454	2.453
2.65	2.632	2.634	2.630
2.85	2.820	2.825	2.825
3.05	3.011	3.014	3.010
3.25	3.223	3.221	3.221

Table 4: Real distances to the box and calculated distances using the SR3000 camera for three different integration times

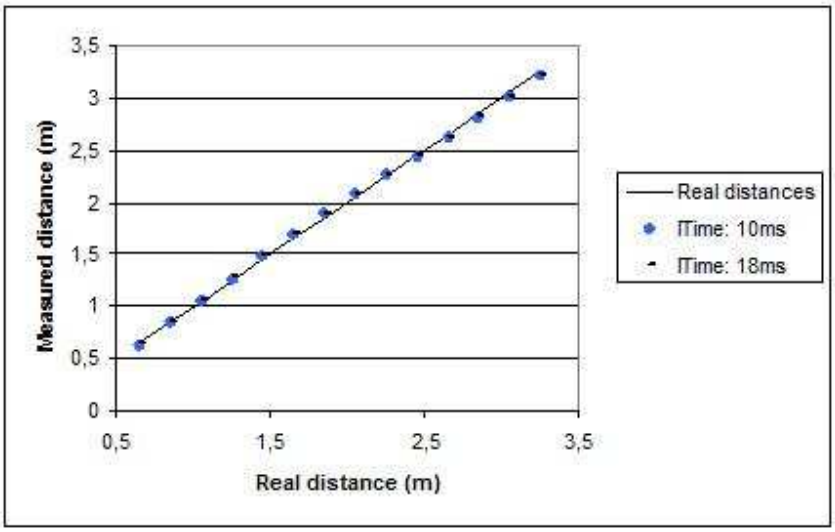

Figure 86: SR3000 measured depth to the box in function of the real depth for two different integration times (ordinate and abscissa: Distances in meters).

As we see in the precedent relation we can calculate the real distance by adding the error to the SR3000 measured distances. Thus, we only need to model the error. As we observe on Figure 87 the error function could be approximated by a sinus function plus an offset inside the studied distance range going from 0.6 to 3.2 meters.

First of all we will determine the offset which maintains the sinus shifted in the 'y' direction. After doing so, we will deduce the sinus formula and the total relation in order to reduce the error between the real and the measured depth distances obtained with the SR3000 camera.

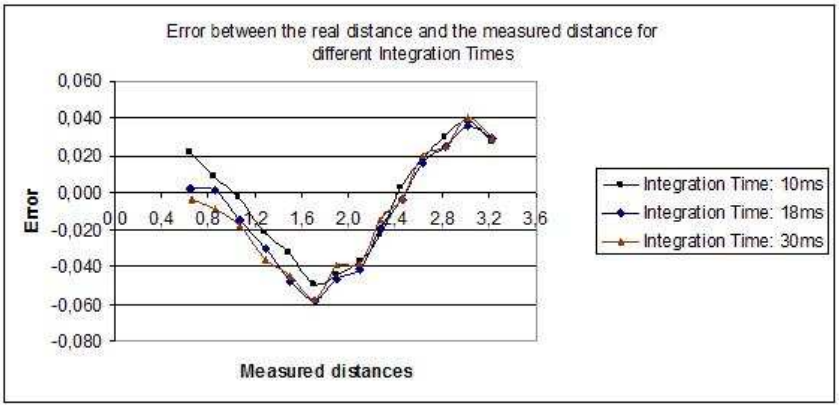

Figure 87: Difference between the real distance and the measured distance for three different integration times (Distances in meters).

From Figure 87 we see that the sinus amplitude can be deduced by subtracting the peak values, using the relation:

$$\frac{38+58}{2} = 48mm$$

From here we can deduce the offset that we must use in order to model the error function. These offset is:

$$-58 + 48 = -10mm$$

The following step is to define the sinus function. In order to do this, we will support in Figure 88 where we have centered the sinus in order to determine the period and phase. As we can see in this figure the semi-period goes from 1 to 2.4 meters, which implies a total period of 2.8 meters. We can also see that the sinus function begins at x=2.4meters; this implies that we have to add a phase of -2.4meters in order to model the sinus approximating the error function. With all that we obtain a sinus which follows the relation:

$$Modelled\ sinus = 0.048 \cdot sin\left(\frac{2\pi}{2.8} \cdot (SR3000\ Measured\ Distance - 2.4)\right)$$

This sinus is also represented in Figure 88.

At the end we obtain the final relation defined by:

$$Corrected\ Distance = (Measured\ Distance) - (0.01) + (Modelled\ sinus)$$

On Figure 89 we show the new error in depth distances after applying the correction. To display this error we have applied the correction to the measured depth distances and we have subtracted the corrected distances to the real distances. As we can see the final error inside the range of 0.8 and 3.2 meters is always smaller than 1cm. Only for 0.6 meters we have an error of 2.2cm in absolute value, however this distance is almost never attained because if we want to have a complete view of the arms the person has to be placed at least farther than 1.5 meters to the camera, and considering that the extended arm of an average person ranges about seventy centimeters, the closest distance of the hand to the camera will be about 0.8 meters.

We can conclude that using the relation that we have deduced in order to correct the distances measured by the SR3000 camera, we have reduced the error from almost 5cm to only 1cm, which is an appreciable reduction.

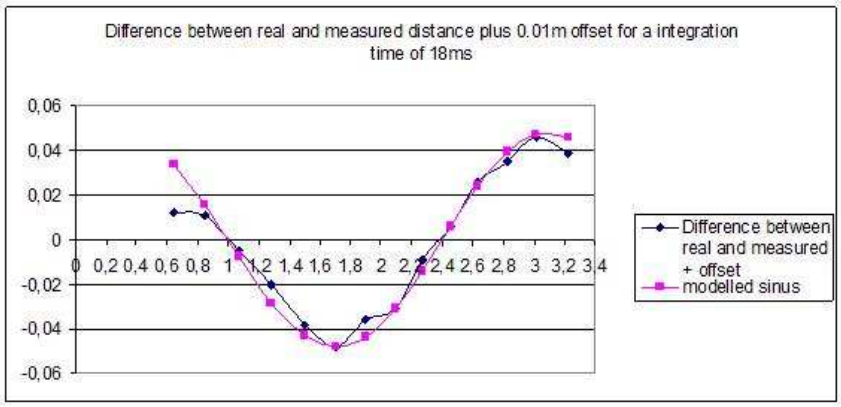

Figure 88: Difference between real and measured distance plus 0.01m offset for the integration time of 18ms and modeled sinus representing this difference (Distances in meters).

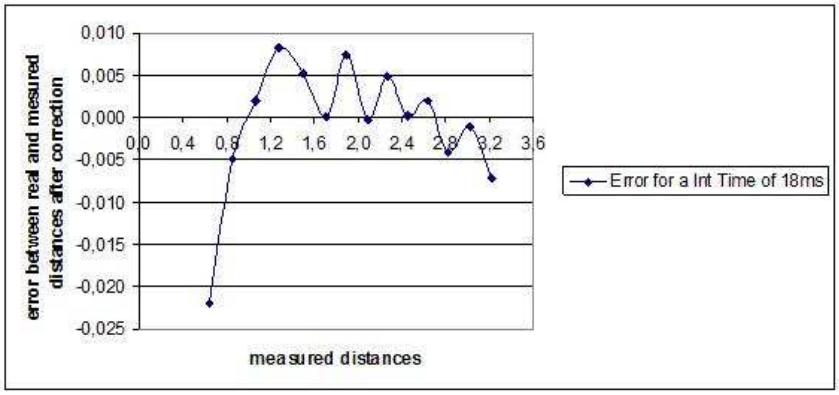

Figure 89: Error calculated after correction using the found regression function (Distances in meters).

Conclusions

With these two experiments we have seen that it could be interesting to set the integration time to 18ms. In one hand this time allow us to get small standard deviations and diminishes the non systematic errors. In the other hand, it is a quite low integration time which in convenient if we want to achieve a fast frame-rate for the tracking and will allow to have a time margin for the computations for each tracking and mapping loop.

In Experiment 2 we have seen that there exists systematic errors on the depth measurements done by the SR3000 camera and we have deduced a function to reduce them by approximating the error function and compensating the measures using this function.

These choices in integration time and the function helping to correct systematic errors, have been integrated into the imitaSoft code in order to improve the quality of the imitation.

6.5.2 Experiment 3: Choice of amplitude threshold for noise reduction in SR3000 images

Introduction

The objective of this experiment is to observe the effect of the sources of noise which affect the SR3000 images and try to reduce these effects using the functions present in the C++ SR3000 library. As observed in 6.5.1 the windows in the scene are one of the elements introducing a high level of noise. But windows are not the only objects introducing noise in the scene; some kind of tissues, often black objects, some kind of metals or other reflecting objects, introduces also noise in the scene [21].

In our project we are processing and filtering the SR3000 coordinates data in order to isolate the person inside a virtual box delimited by a range in the 'x', 'y' and 'z' coordinates. In this way, all objects outside this virtual bounding box, will not be inside the hand tracking zone and will not distort the tracking. But when a noisy object is present in the SR3000 camera field, even if this object is actually placed outside this virtual box, the light reflection can have effects inside the virtual box, propagating the noise inside the tracking zone and affecting the measurements of coordinates consequently.

Mesa Imaging software provides a function which helps to reduce this kind of noise:
`SR_SetAmplitudeThreshold(SRCAM srCam, unsigned short val);`

As said in the mesaSR library documentation (an extract of this documentation is presented on Appendix A) this function sets all distances and intensities to zero if the amplitude signal arriving to the camera is lower than the amplitude threshold. Using this amplitude threshold we have a method to discard pixels representing noisy objects. In this experiment we will observe the effect of applying different amplitude thresholds and we will chose one of these thresholds helping to reduce the noise but bearing in mind that we do not want to lose relevant information for the tracking.

Experiment

Next, we have taken some depth images of a plant applying different amplitude thresholds. On Figure 90 we observe the intensity image given by the SR3000 camera, which corresponds to the scene presented on the six depth images of Figure 91. On these depth images, the objects which are close to the camera appears in darker colors, while the objects which are far from the camera appears in lighter colors. The amplitude threshold is used to remove noise from the images; all the information from the pixels which are below the amplitude threshold is removed and they are represented on black. Our objective is to find the maximum amplitude threshold that can be applied to remove noise and irrelevant information, without losing relevant details of the main image, in this case the plant. The scale used by Mesa Imaging to express the values of the amplitude threshold does not have any units associated.

The amplitude thresholds used on the six images on Figure 91 correspond to the following six values:

Image 1: 10	Image 2: 40	Image 3: 80	Image 4: 120	Image 5: 160	Image 6: 200

For the first of the six depth images we have used an amplitude threshold value of 10. It is a low threshold value and because of that we can see the image without almost no modifications done by the threshold applied. We can appreciate that there exist a pointillism effect on the floor, this indicates that the floor appears on the image as a rough ground. It is a noisy effect because it does not correspond to the reality of the laboratory floor. This happens because the

floor is quite glossy and introduces noise on the image. In the following images we can see that the first thing which becomes black by the effect of the threshold is the floor. As we increment the threshold we also observe the appearance of a circle and the parts of the image far from the center becomes black. The chosen threshold has to avoid as much as possible this effect because we could be losing information when performing the human tracking if the hands reach the borders of the image. We observe that this effect begins to be visible in the upper corners of the image for a threshold of 80, thus we will choose an amplitude threshold lower than 80.

Figure 90: Intensity image of a plant in a low noise scene with a matt wall in the background.

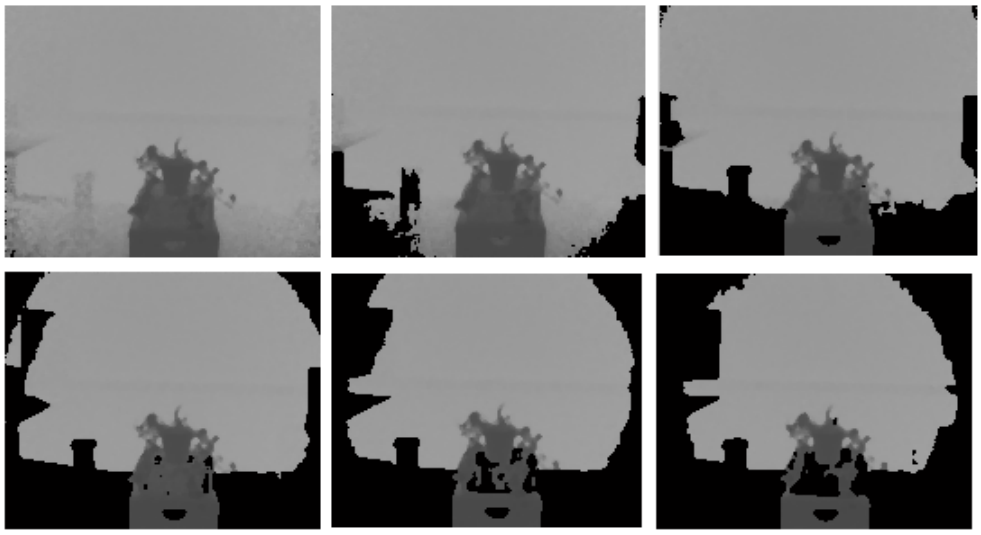

Figure 91: Images with little noise taken using different amplitude thresholds.

Figure 93 presents three depth images with the same plant and windows in the background. The three images represented on this figure are taken for the following threshold values:

Image 1: 0	Image 2: 40	Image 3: 80

Figure 92 corresponds to the intensity image representing the scene on the depth images of Figure 93. On this Figure 93 we observe that the background is much more noisy that the background on Figure 91. In fact the windows introduces a lot of noise as they are also external sources of light. SR3000 camera responds much better when the scene is poorly illuminated by external sources of light and is mostly illuminated by its own source of infrared light. In the

first depth image from this series we can see the pointillism effect in all the image indicating variations in depths that are not present on the real scene. We also see that some distant areas appear in dark gray colors; this is also a noisy effect, because with the color gradient used the more distant areas to the camera must appear in lighter colors. Given that almost the whole image becomes really noisy because of the windows, when we increase the amplitude threshold a little (from 0 to 40), a lot of noisy objects disappears and most parts of the image becomes black. In these images we can see other objects introducing noise as it is the case of the two chairs at the end of the room, which also becomes black with a threshold of 40. The conclusion drawn is that the background has to be as matte as possible in order to reduce the level of noise.

Figure 92: Intensity image of a plant in a noisy scene with windows in the background.

Figure 93: Highly noisy images taken using different amplitude thresholds.

The wall that we have used as a background when performing the real time imitation, is the same background from the 91. This is a quite good background as it does not reflect a lot of light and thus the noise is reduced.

On Figure 94 we observe a performer which will be imitated by Bioloid in front of the matte background. We have chosen a threshold of 60 which allow to keep the useful depth information of the main image and allow to eliminate the main sources of noise. At the same time we do not lose the image borders, which is an effect produced when we increase to much the threshold. In this image the performer appears quite clearly in the depth image and there is not noise present in the zone of the arms.

Figure 94: Performer in front of a low noisy background using the value of 60 as the SR3000's amplitude threshold.

Conclusion

The quantity retained to be used on the imitaSoft software as amplitude threshold has been the value of 60. With an amplitude threshold of 60 we keep the main useful information during the tracking, that is to say, the hands and shoulders depth information and Cartesian coordinates. Choosing this value we have deleted the main sources of noise. Higher values for the amplitude threshold could imply the loss of important information from the performer, while lower values could increment too much the noise level and compromise the quality of tracking by losing the hands track.

6.5.3 Experiment 4: Conversion of Bioloid velocities into AX-12 units

Introduction

This experiment has been designed with the objective to find the relationship between the servo (AX-12) velocities expressed in the AX-12 convention and the servo angular velocities expressed in degrees per second. As indicated in the AX-12 manual [25] the servos angular velocity in the AX-12 convention can be set between 1 and 1023; 1023 corresponding to the maximum velocity. ImitaSoft first calculates the velocities in degrees per second and then they have to be adapted to the AX-12 velocity convention. The conversion factor between degrees per second and the AX-12 units could be directly deduced theoretically, but with this experiment we want to corroborate the linearity between the velocities in degrees per second and the velocities in the specific AX-12 units.

With this study we will deduce the conversion factor necessary to express the velocities in the servos convention as well as in degrees per second. ImitaSoft uses the intermediate step to calculate the velocities in degrees per second because, in future works, it could be interesting to adapt the software to other robots using different conventions for the velocities. Given this fact, it is important to have the velocities in an understandable units just before to make the conversion to AX-12 units. We have to keep in mind that we are not taking into account the communication times with the computer and the acceleration and deceleration times but these times are insignificant compared to the measured times that we will use to compute the velocities.

Experiment

To accomplish our goal, we have developed a small program which uses the function *clock()* and the variable *CLOCKS_PER_SEC* from the library *time.h*. The function clock() returns the number of clock cycles elapsed since the program was launched. The macro constant CLOCKS_PER_SEC specifies the relation between a clock cycle and a second (clock cycles per second). On the computer used for this experiment, the value of the constant CLOCKS_PER_SEC was 1000.

What we have done is to set a given velocity to an AX-12 servo while making it rotate from 0^o to 180^o. We have executed the function clock() at the beginning and at the end of the movement and with that we have measured the increment of clock cycles between this two instants. Dividing the precedent difference by the constant CLOCKS_PER_SEC we have obtained the elapsed time needed by a AX-12 servo to carry out the rotation of 180^o. With this time and the angle increment, we can deduce the angular velocity of the servo, and following the same procedure for different velocities we can find the relation between the velocities in the AX-12 scale and the velocities expressed in degrees per second. On Table 5 we observe the data that we have obtained with this experiment.

To find the relation between Bioloid velocities expressed in the AX-12 scale and in degrees per second we use the data presented on Table 5 with the following equation:

$$Servo\ velocity\ in\ degrees\ per\ second = x \cdot Servo\ velocity\ in\ the\ AX12\ scale$$

With x being the conversion factor that we want to find,

$$x = \frac{Servo\ velocity\ in\ degrees\ per\ second}{Servo\ velocity\ in\ the\ AX12\ scale} = \frac{300}{500} \Rightarrow x = 0.6$$

For the range of velocities going from $15^o/s$ to $300^o/s$, the relation between velocities is

Velocity in AX-12 units	Angle (º)	Elapsed time (s)	Velocity (º/s)
25	180	12	15
50	180	6	30
100	180	3	60
200	180	1.5	120
300	180	1	180
400	180	0.75	240
500	180	0.6	300

Table 5: Velocities and times to perform a rotation of 180º with a AX-12 servo.

linear, and we only need to multiply or to divide by 0.6, in order to express the angles in one or in another convention.

To avoid damaging the servos in case of losing the control we have limited the velocity to 300º/s. This range limitation will allow us to increase the reaction time if we envisage a collision of the Bioloid arms with its own body and reorient the movement in real time in order to do not damage the servomechanisms. At the same time a maximum velocity of 300º/s for each arm articulation is adequate for our purpose of imitating the arms in real time and allow us to perform actions at a quite normal velocity.

Conclusion

Thanks to the scale factor and to the relation found, we can deduce the angles in the AX-12 scale by dividing the angles in degrees by the constant *0.6*.

7 RESULTS

7.1 Introduction

In this chapter we will present the results on imitation that we have obtained with the software imitaSoft. The results have been displayed for the two main parts of the software: results on the imitation in real time and results on the imitation in deferred time using some VooDoo sequences as input. All these actions are learned by imitation as new actions and can be replayed by the robot later on. As we have seen on chapter 6.4, the new actions learned are the actions that have been previously imitated on real time or that have been imitated on deferred. First, the robot performs the imitation of the demonstrator; thanks to that, the software internalizes the information of the actions. Then, the software imitaSoft can transmit this information to Bioloid in order to make it able to reproduce these new actions that has been learned by imitation.

To present the results we have firstly used some simulations on learned actions thanks to deferred imitations of the VooDoo computer model. To perform these simulations we have elaborated a model of the Bioloid arms using Matlab and the robotics toolbox.

After that, we have performed real experiments using the robot by recording some videos of the imitations in deferred time and in real time. Sequences of frames of these recordings are exhibit in this chapter. Besides presenting these sequences, we discuss the positive and negative points of the results and we expose which aspects can be improved.

7.2 Results: Deferred imitation using VooDoo

In order to check the quality of the mapping we have performed simulations on the imitations of some VooDoo sequences that had been recorded by the creators of the VooDoo software and are available at the VooDoo website [7]. We compare these simulations of the Bioloid robot arms with some frames of the VooDoo sequences. We remember that Bioloid and its simulation have only 3DOF per arm, while the VooDoo model has 6 DOF per arm; thus, reducing the number of possible configurations and making it necessary the mapping.

On Figure 95 we can appreciate the VooDoo model in a given instant from a sequence where the performer was reading a book (at left). The Matlab Bioloid arms model that we have developed to perform the simulations is displayed at right. This robot simulation has two servos on each shoulder and one servo on each elbow, as well as the Bioloid robot. In this figure, the robot simulation is imitating the VooDoo model when it is reading a book. We can appreciate how the end effectors position of the robot simulation (the hands) are representing quite well the VooDoo hands position. We see for example that the elbows are bended to allow having a book between the hands, as well as the VooDoo model. In the simulation we can also observe the shadow of the arms which give us more information about the arms position. We see for example how the left arm is crossing the sagittal plane allowing the left hand to be placed at right of this plane (from the robot simulation point of view); on the other hand, the pitch angle for the right arm is greater than the pitch angle for the left arm allowing this arm to be more opened to the right. All these observations reproduce well the VooDoo computer model configuration at left.

Figure 96 shows a frame of a video of Bioloid recorded while it was imitating the 'reading a book' VooDoo sequence and shows the mapping from the VooDoo model to the robot for a given frame. This position corresponds to the position showed on the simulation on figure 95 .

[7] VooDoo sequences at: http://wwwiaim.ira.uka.de/users/loesch/har/

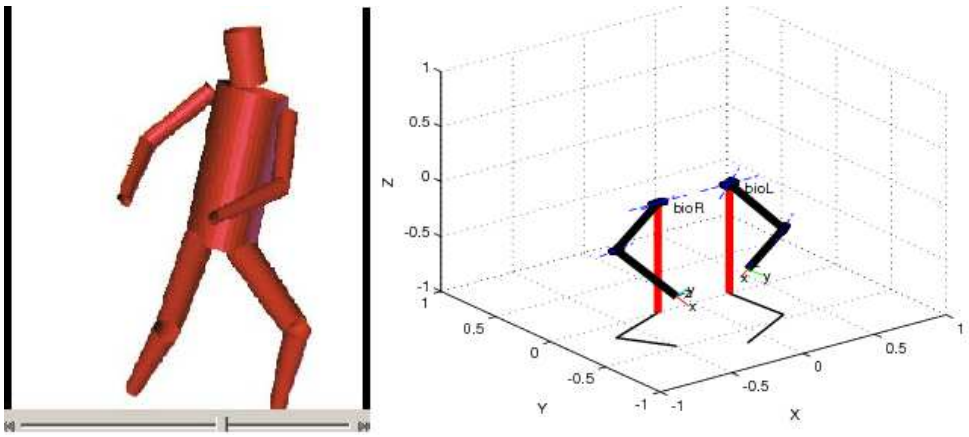

Figure 95: VooDoo model and mapping simulation of a frame from the 'reading a book' sequence.

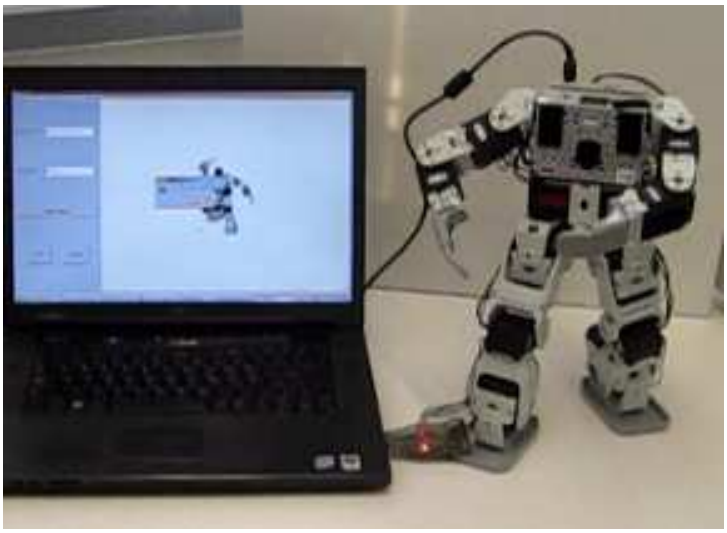

Figure 96: Bioloid video frame during the imitation of the 'reading a book' VooDoo sequence.

On Figure 97 we present some frames of a VooDoo sequences where the performer was removing an object. This VooDoo sequence is on-line at the VooDoo website. On Figure 98 some frames of the mapping simulation are represented for this 'remove object' VooDoo sequence. As we can see on the original VooDoo sequence, the VooDoo model begins with the right arm extended forward and little by little goes backwards to perform the action of removing an object. As we can observe by comparing the Bioloid simulation with the VooDoo computer model the imitation is quite faithful to the VooDoo action as a whole. While the left arm is almost not moving along the sequence, the right arm performs a movement from front to back by rotating around the first servo of the right arm chain. On Figure 99 we have displayed some frames of a recording from the imitation performed by Bioloid from the 'remove object' VooDoo sequence. This is a deferred human imitation, given that, the VooDoo sequence has been extracted from a previous human imitation.

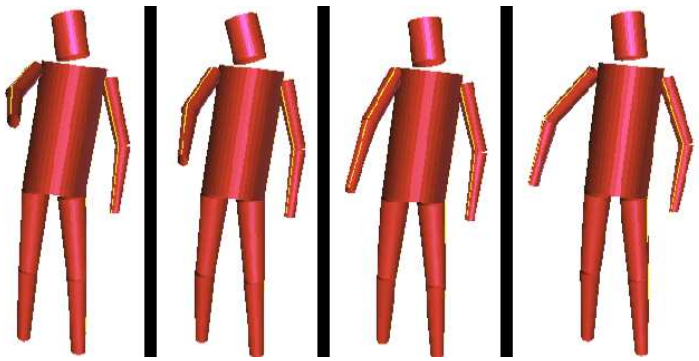

Figure 97: Some frames of the VooDoo model imitation of a 'removing an object' action.

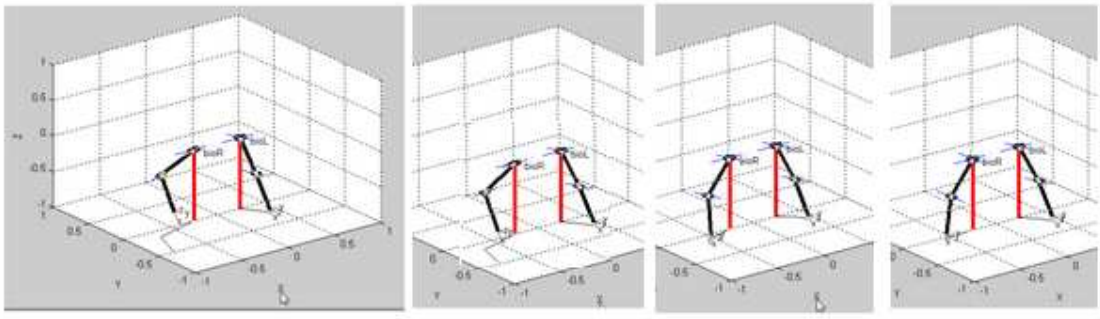

Figure 98: Mapping of the action 'removing an object', using a robot simulation with 3DOF for each arm.

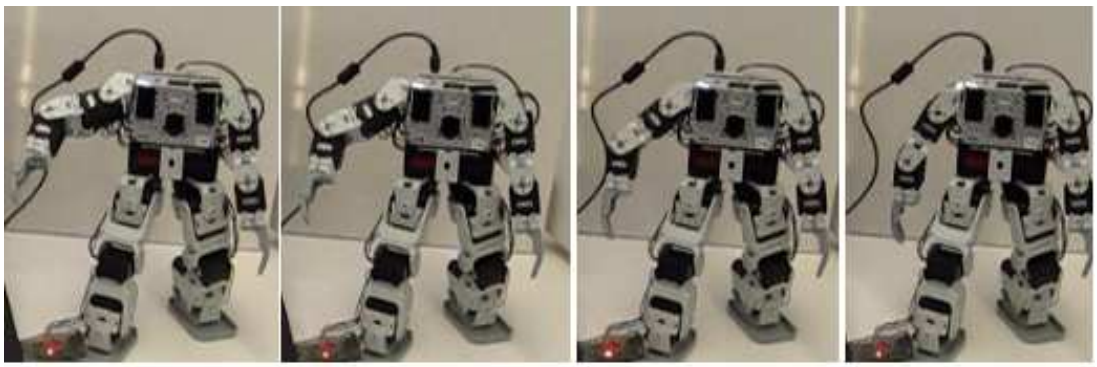

Figure 99: Imitation of the action 'remove object' from a VooDoo sequence, using Bioloid.

The experiments done using Bioloid shows that the robot actions imitating the VooDoo model movements are quite faithful to the model actions. However we have observed that the movement is not as fluid as it could be. A possible solution in order to improve the fluidity would be to reduce the velocities by scaling them in order to better splice the differentials of movement. Reducing the velocities would slow down the entire movement but we would achieve a better synchronization.

Once the imitation from the VooDoo model has been performed a first time, the angles and

velocities are stored because imitaSoft allows to 'learn' the new actions. The following times that the same action is performed with Bioloid, the stored information is used. Thus, the number of computations to be done is smaller and, consequently, the movements from these learned actions gain fluidity.

On Figure 100 we have displayed some frames of the VooDoo sequence 'hold out object' and the corresponding imitation performed by Bioloid. In this action we can see the right arm rotating around the first servo of the arm chain; thanks to this rotation the right arm goes from back to front. At the end of the sequence the elbow (third servo in the arm chain) is not so bent and we observe the arm extension thanks to this fact. In the last frame we also observe how the pitch angle diminishes and the arm approaches a parallelism with the sagittal plane.

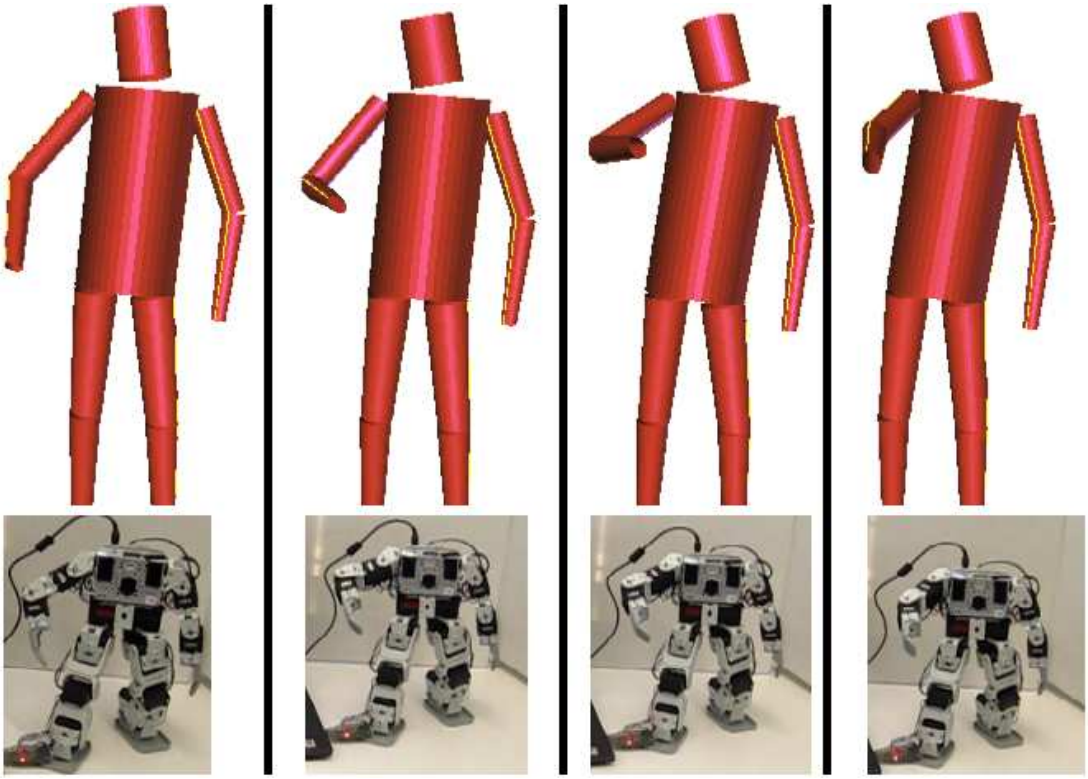

Figure 100: Four frames of the VooDoo model imitation of a 'holding out an object' action and the corresponding imitation performed by Bioloid.

7.3 Results: Real time imitation results

The real time imitation algorithm is composed of two main parts: the tracking and the mapping. This imitation can be done for one or for both arms. Qualitative results are presented for the tracking and the mapping, and we will present the main strengths and weaknesses of the imitation results.

First, a tracking sequence where both hands are tracked is presented. On Figure 101 we can see a tracking frame with the imitaSoft GUI. As indicated on the main screen of imitaSoft, the shoulder coordinates have been extracted based on their fixed position superposed with the green squares and the hand tracking is being carried out. From now on, the demonstrator has to stand at the same place and he can move freely their hands which will be tracked. However the hands must always remain in front of the coronal plane as imitaSoft uses the proximity information to track them. Both hands are tracked with red circles.

On Figure 102 are displayed some frames following the first frame presented on Figure 101. On this frames we can see how the hands are tracked. The choice of a matt background together with the choice of an adequate amplitude threshold helped to reduce the noise and allowed a good parsing of the demonstrator by using the depth information. On this image we can see the notification which inform us that the coordinates of the shoulders have been found. One aspect of the tracking that could be improved is the fact that the tracking is not centered on the middle of the hand, but it oscillates along the hand because we only take the information of one pixel at each instant of time. This fact makes the red circles to shake along different positions along the hand and does not allow a smooth tracking. Some kind of average using the position of the pixels on the hand could be performed in future works in order to smooth the tracking.

Figure 101: ImitaSoft main screen with the tracking images: intensity and depth images

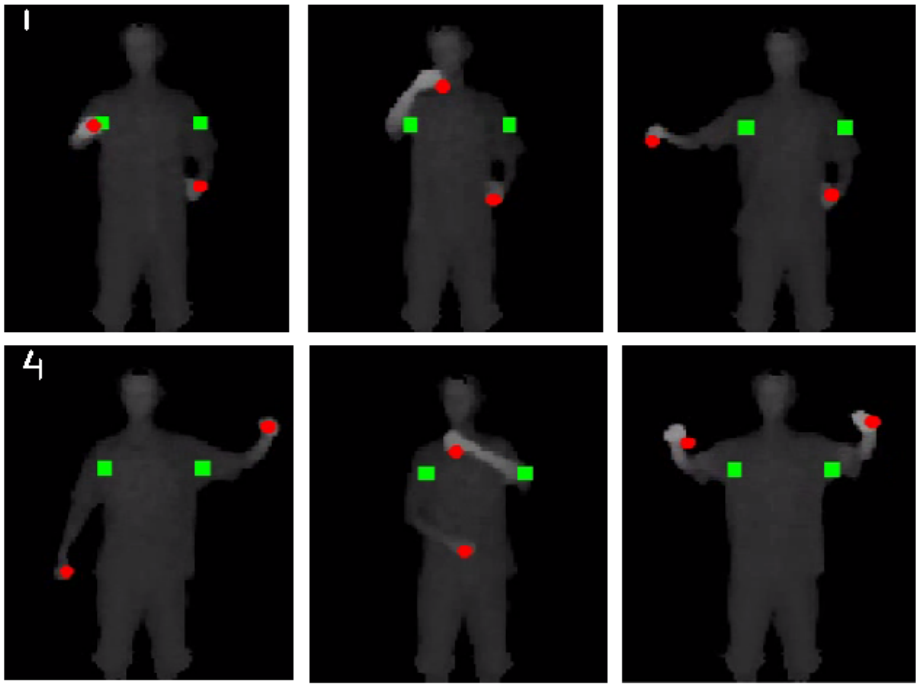

Figure 102: Six depth frames of a tracking sequence where both hands are tracked

An aspect that has to be taken into account is that the demonstrator cannot join hands during the tracking of both hands. In fact, the tracking uses a local area for each hand; if both local areas becomes superposed, both red circles will be track the hand which is closer to the camera, and from there we only have the option to restart the tracking in order to separate the red circles which tracks the hands. A system which takes into account this fact has not been studied in this work and the tracking is only valid when the hands moves independently one of the other.

Below, we present some frames of a real time imitation that we have performed using imitaSoft. On Figure 103 we observe the organization of the scene; we can partially observe the SR3000 camera mounted on a tripod at left, the robot at the middle and the demonstrator at right. The SR3000 camera is always recording the demonstrator actions while the robot imitates the demonstrator actions in real time. Figure 104 shows four of the frames of a sequence where Bioloid performs the imitation of the left arm movements. On this sequence we can observe that the movements are quite well imitated in general, we always have a goal oriented mapping, thus it is the hands position which are imitated.

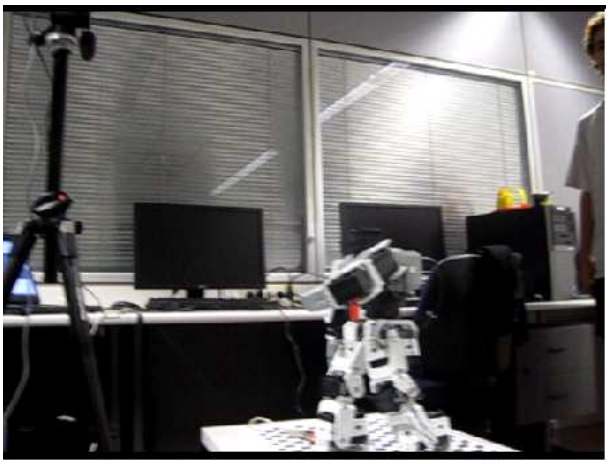

Figure 103: Distribution and organization of the scene. Demonstrator, camera and robot placement.

One of the things that could be improved during the imitation in real time is the precision of the SR3000 coordinates measures during the tracking. We can see for example in the fourth image of the sequence that the demonstrator has its arm almost totally extended while the robot has its arm quite bended. This is because we have used lengths for the arm and forearm vectors longer than the real arm and forearms of the demonstrator. When trying to use the real lengths for these vectors we often obtained measures of the vector shoulderHand longer than the sum of the lengths of arm plus the forearm because of the inaccuracy of the coordinates obtained. If the length of the vector shoulderHand measured with the SR3000 during the tracking is longer than this sum a warning message indicating that the hands are out of range appears in the imitaSoft main screen and the mapping stops temporarily, because the vector lengths are not consistent (the length of the vector shoulderHand cannot be longer than the length of the forearm plus the length of the arm). In order to solve that and allow the mapping we have used longer measures of arm and forearm, with the result of obtaining the arm more bent than necessary during the real time imitation.

On future works later models of ToF cameras could be integrated as input sensors for imitaSoft in order to improve the precision of the measures and the tracking could be improved to allow a smoother hand track.

In general, the real time imitations shows good results as we can see on the image sequences and a video has been posted on youtube showing how Bioloid imitates the movements of a demonstrator using the imitaSoft software. This video is posted at the url displayed on the footnote [8].

[8] http://www.youtube.com/watch?v=4HPes4LBWOY

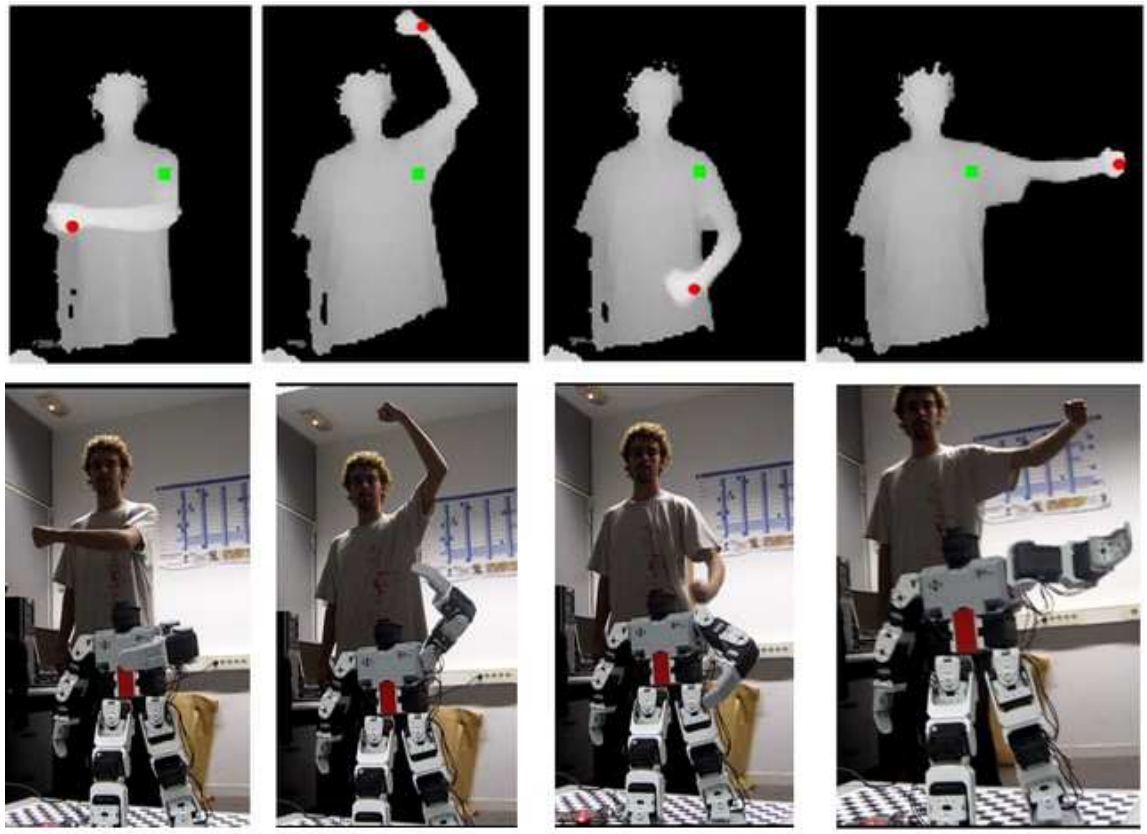

Figure 104: Human imitation on real time of left arm movements by Bioloid.

8 CONCLUSIONS

8.1 Conclusions

Along this project we have presented a software called imitaSoft that has been created with the goal to allow the imitation of a human upper torso thanks to a Bioloid robot.

We present the conclusions point by point following the objectives on section 2. We describe the strengths and the weaknesses of imitaSoft and on section 8.2 and we present possible improvements for future work.

General conclusions:

- A software called imitaSoft has been developed in order to address the imitation problem. This software allows the imitation of the human arms that are imitated using a Bioloid humanoid robot.

- Bioloid is the robot which is used in the imitations by using imitaSoft. Thanks to this software, Bioloid is able to imitate the human arms movements in deferred time by using an intermediate tracking software called VooDoo and it is also possible to perform an imitation in real time using the imitaSoft internal system of tracking. Moreover it gives the possibility to store the angles and velocities during the first imitation and reproduce later on the imitated movements. Thus, it is not only possible to control the robot in real time, but the software also allows Bioloid to "learn" new actions by imitation.

- The system created is totally unobtrusive from the demonstrator point of view. We have used a SR3000 camera as input sensor which allows to sense the demonstrator using depth information, and a hands tracking system based on proximity information has been created thanks to the information provided by this sensor.

- The goal oriented mapping has shown reliable results for a 3DOF robot. Moreover, the same mapping could be used with robots with more DOF given that most of humanoid robots and robot arms have at least 3DOF structured following the Bioloid's servos configuration. The mapping created allows us to reproduce the hand position but not its orientation. To reproduce the hand orientation 3 additional DOF would be needed in the humanoid robot hand and a new tracking system able to sense the hand orientation would have to be created. The goal oriented imitation is interesting because is highly related with the method used by humans to learn new actions by imitation and we have provided the robot with this kind of imitation.

- The software imitaSoft, when performing the tracking and the mapping, first calculates the angles and velocities in human readable unities (the angles in degrees and the velocities in degrees per second) prior to do the final transformation to the Bioloid nomenclature for angles and velocities. Independent functions are created to do this transformation from degrees to Bioloid unities. This architecture would easily allow to control other robots with few modifications of the software by developing new independent functions able to do the transformation between the human readable angles and velocities to the specific unities used by the new robot that we would like to use. Thus, the system architecture has been created to be extended in the future, with few modifications, to other robots with similar arm structure. However, we have to take into account that self-collisions have not been addressed in this project and for some robots it is important to address them in order to avoid damaging the robot.

After having explored the possibility to use the tracking provided by the VooDoo software for the real time imitation we have seen that this tracking was little reliable if we wanted to use this information to implement a mapping in real time. Given that, we have decided to implement our own tracking system, by creating a simpler tracking adapted to our needs of a goal oriented mapping. At the end we have only used VooDoo as an intermediate program able to provide imitations in deferred time from previous recorded reliable VooDoo sequences.

Thanks to the tracking created, we have developed a mapping algorithm which has provided a mean to transform the human movements into similar robot movements in real time, by adapting the different number of joints between a human and the Bioloid robot.

In order to establish the link between the directions proposed in the objectives (section 8.2) and the conclusions, we can say that the direction number two from the objectives has been retained by implementing a complete GUI application able to perform the human imitation in real time but we have integrated some aspects of the point number one by creating a deferred imitation using previous recorded VooDoo sequences. Moreover we have implemented a learning module which provides the ability to store data form actions that have been imitated previously and that can be reproduced by Bioloid later on.

On section 7 we have seen the qualitative results in terms of imitation sequences performed by Bioloid. The goal oriented imitation gives quite good results as we have seen on the frame sequences displayed on section 7 that have been extracted from the imitation video recordings using the software imitaSoft. One of the imitation videos can be seen on Youtube [9].

[9] http://www.youtube.com/watch?v=4HPes4LBWOY

8.2 Future Work

In this section we indicate possible lines which could be followed as future work in order to improve the software with aspects that have not been addressed or that have been little addressed in the project. Some of these works could be achieved with few work, while others would need a long period of work and would introduce a totally new approach to the software. We have found refreshing to give ideas in different directions that could improve different aspects of the project, even if the future work is a really open field and everyone interested in the project is free to apply their ideas to the project.

One of the most interesting works to be done in the future would be to develop a real time shoulder tracking. As said in precedent chapters and contrarily to what have been done to track the hands, shoulders coordinates are found at the very beginning of the software execution. Doing so, once these coordinates have been found, the demonstrator is bound to stand at the same place while he is moving their arms. If he walks and change its position, the vector shoulder-hand will not be well calculated as the algorithm is not taking into account the new shoulders position. This error will be reflected in the robot causing a totally distorted imitation. However if a shoulder tracking in real time is done the vector shoulder-hand will always be calculated even if the demonstrator move inside the field of view of the camera. As in our case we have been tracking the hands, we have worked with the information obtained from the closest limbs to the camera. In order to deploy a shoulders tracking, a different tracking technique would have to be found.

Following with the idea of working with the tracking in order to improve the mapping, another interesting thing to be done would be to allow the tracking of the hands when they are behind the coronal body plane. Being able to track the hands behind this plane would enlarge the range for the arm imitation.

In our imitation experiments we have observed that when we are performing the tracking of a demonstrator in a static position, the circles which tracks the hands presents a shaking movement. This issue could be solved at the tracking level by doing an average of the closest pixels to the camera representing the hand. This issue could be also solved at the mapping level by processing data using windows of few frames in order to reduce the higher frequencies in the movements, that makes these movements less natural. We could do this by applying a low pass filter to the coordinates evolution or by finding a trade-off between the movement observation and the movement prediction evolution by means of a Kalman filter for example.

Another aspect that could be developed in the future is all about to the collision of the robot arms with the robot itself. This problem can be addressed by defining a working space around the robot which will mostly depend on its geometry. This working space will represent the space kept by the robot as it moves and it must not be accessible by the robot arms and hands, preventing the robot to self-collide. Whereas the arms moves, the rest of the Bioloid body is static, this fact could facilitate the work to define a simple working space which could also be static. However a more complete self-collision algorithm would have to take into account the mobile parts of the robot. This implies that the working space around the arms would have to be dynamic and follow in real time the arms positions in order to avoid self-collisions between an arm and the other as they can be moving at the same time. This mobile working space would have to be extended to the rest of the whole robot body if we were working with a fully mobile robot, for example with a robot able to walk. In this case it would be mandatory to define a fully mobile working space around the whole body following in real time the robot positions. As the working space would not be accessed by the mobile parts of the robot, the self-collisions would be avoided. If we wanted to work and integrate other robots different to Bioloid to our software,

working spaces adapted to each of these robots would have to be defined as the working space depends on the geometry of the robot. The fact of designing a software able to take into account the self-collision problem would offer a safer environment for the robot.

One of the most attractive works would be to adapt the software to make it possible the control of other robots arms different than Bioloid arms by means of imitation. In order to implement the imitation control with other robots, the primary condition of having a similar robot arm configuration compared to that of Bioloid must be satisfied. Thus, the servomotors have to occupy the same relative positions following the same axes, that is to say, we need to have two servomotors on the shoulder and one servomotor on the elbow following the same orientations as the Bioloid servomotors. Other robots with higher number of servomotors per arm can be used with the condition of having at least three servomotors specifically oriented allowing to preserve a similar arm structure to that of Bioloid. Finding robot arms with these features is not difficult in the available humanoid robots in the market as well as in the industrial robot arms, this is because the Bioloid robot arm configuration is one of the simplest allowing a qualitative mimic good enough to evoke the image of a human. Some examples of these robots are Robonova, for the humanoid robots, Staubli Unimation for the industrial robot arms, or the WAM arm from Barrett that is a robot arm really used in different research centers. This work could be quite interesting and probably not really complex as the output of our software are the angles and angular velocities that has to be given to each of the three servomotors conforming the arm. The two works to be done in this case would be in one hand to adapt the angles and velocities to the new robot convention and in the other hand to modify the way in which we send these information to the robot by adapting the communication system to the new robot.

A more ambitious work would be to integrate the legs' imitation into the software. This would be a possible subject of work for a full project. The legs' movement imitation would possibly be even more complex because in this case a trade-off between the imitation and the robot balance has to be found. Also the tracking part would be more complex because legs cannot be tracked as we have done with the hands using the proximity information. Probably, the simplest way to perform the mapping and avoid the balance problems would be to create some stable steps primitives in advance, in this way the more complex part would be to track and deduce which kind of step is performing the demonstrator and to chose the primitive that imitates at best the demonstrator action.

Our first idea had been to work with VooDoo and use its own tracking system. This software created in the Kalsruhe University has been designed to offer a computer 3D model of the human body imitating a human in real time. At this moment its compilation is a little unstable and the tracking is being improved currently by their creators. In our laboratory conditions we have observed that we needed a more reliable hand tracking than that introduced by VooDoo to perform the mapping, but with an improved version of VooDoo giving more reliable information it could be possible to use the VooDoo tracking in order to have a representation of the whole body and be able to perform a mapping from here in order to work with a full robot imitating a human. In this case we could extract information not only from the arms and the legs, but also from the head movements and the torso movements.

Nowadays, a new camera able to extract depth images has appeared on the market: it is the Kinect. New algorithms exploiting this technology are being developed very fast and a complete body tracking system is already available as open source software. Using the output information from this tracking and integrating the imitaSoft mapping process would be another direction that could be followed in future works in order to develop a real time imitation using the Kinect camera which is a much cheaper technology.

A APPENDIX

Function presented on the Experiment 3 (6.5.2) and used on ImitaSoft software in order to reduce the noise of the SR3 images:

In libmesaSR.h from mesaSR library we find the following function used to set the SR3000 amplitude threshold with the following comments:

*/** Sets the Amplitude Threshold of the camera. The default amplitude threshold is 0. Setting this value will set all distance values to 0 if their amplitude is lower than the amplitude threshold */*

SR_API(int) SR_SetAmplitudeThreshold(SRCAM srCam, unsigned short val);

References

[1] Wikipedia, "Human-robot interaction — wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Human_robot_interaction, 2010. [Online; accessed 19-October-2010].

[2] elPeriodico, "El gasto en i+d en españa cae por primera vez desde 1994." http://www.elperiodico.com/es/noticias/ciencia-y-tecnologia/20101117/gasto-espana-cae-por-primera-vez-desde-1994/590658.shtml, November 2010. [Online; accessed 25-November-2010].

[3] M. V. del Río, "El gobierno recortará la inversión en i+d+i casi un 20% en 2010," September 2009.

[4] Wikipedia, "Kinect — wikipedia, the free encyclopedia." http://es.wikipedia.org/wiki/Kinect, 2010. [Online; accessed 22-Novembre-2010].

[5] T. C. B. Riek, L.D.; Rabinowitch and P. Robinson, "Empathizing with robots: Fellow feeling along the anthropomorphic spectrum," in *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pp. 1–6, September 2009.

[6] A. Bekkering, H.; Wohlschlager and M. Gattis, "Imitation of gestures in children is goal-directed," in *The Quarterly Journal of Experimental Psychology*, pp. 153–164, 2000.

[7] U. de Guadalajara, "Robotica." http://proton.ucting.udg.mx/materias/robotica/. [Online; accessed 21-October-2010].

[8] RoboSavvy, "Bioloid robot kit." http://robosavvy.com/site/index.php?option=com_content&task=view&id=82&Itemid=81, 2010. [Online; accessed 19-October-2010].

[9] C. Breazeal and B. Scassellati, "Robots that imitate humans," *TRENDS in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[10] B. G. Galef, *Social Learning and Imitation*, ch. 4, pp. 263–269. 2000.

[11] A. Billard and M. M.J, "Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture," *Robotics and Autonomous Systems*, vol. 37, pp. 145–160, 2001.

[12] C. Nehaniv and K. Dautenhahn, "The correspondence problem. in imitation in animals and artifacts," in *MIT Press*, pp. 41–61, 2002.

[13] M. Cabido Lopes and J. Santos-Victor, "Visual transformations in gesture imitation: what you see is what you do," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, September 2003.

[14] A. Y. K. H. H. Nakaoka, S.; Nakazawa and K. Ikeuchi, "Generating whole body motions for a biped humanoid robot from captured human dances," in *IEEE International Conference on Robotics and Automation (ICRA)*, September 2003.

[15] Y. Wataru, "V-sido, real-time control system for humanoids." http://kinecthacks.net/official-kinect-drivers/, 2010. [Online; accessed 8-March-2011].

[16] Y. Wataru, "V-sido, real-time control system for humanoids." http://vsido.uijin.com/en/index.html, 2010. [Online; accessed 8-March-2011].

[17] S. Foix, G. Alenya, and C. Torras, "Lock-in time-of-flight (tof) cameras: A survey," *Sensors Journal, IEEE*, 2011.

[18] S. D. R. Knoop, S; Vacek, "A human body model for articulated 3d pose tracking." `http://wwwiaim.ira.uka.de/data/File/Publications/chapter-knoop.pdf`, 2007. [Online; accessed 8-December-2010].

[19] F. Vanden, "Kranf site." `http://www.applied-mathematics.net`, 2009. [Online; accessed 8-December-2010].

[20] M. Imaging, "Mesa imaging - the leading provider of 3d time of flight cameras." `http://www.mesa-imaging.ch`, 2010. [Online; accessed 12-December-2010].

[21] S. A. Guomundsson, "Robot vision applications using the csem swissranger camera," 2006. page 51.

[22] P. Kormushev, S. Calinon, and D. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 3232 –3237, 2010.

[23] H. Rodriguez, C. F.; Quintero and A. Bogota, "Arm flexion and extension," in *Movements of the Upper Limb*, pp. 7–18, November 2002.

[24] Wikipedia, "Elbow — wikipedia, the free encyclopedia." `http://en.wikipedia.org/wiki/Elbow`, 2010. [Online; accessed 24-January-2011].

[25] Robotis, "Dynamixel ax-12 user's manual," 2006.