Bachelor's thesis

Grau en Enginyeria en Tecnologia Industrial

User tracking and haptic interaction for robot-assisted dressing

MEMORY

Author:	Raimon Padrós Valls
Director:	Aleksandar Jevtić
Co-director:	Carlos Augusto Ocampo Martinez
Call:	June 2017

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Abstract

The aim of this work was development of visual and haptic interaction model for a robot dressing assistant. The work was performed as a part of the I-DRESS project whose goal is to develop an interactive robotic system that will provide proactive assistance with dressing to disabled users or health-care workers whose physical contact with garments must be limited to avoid contamination.

In this work, haptic and vision were explored as modalities for human-robot interaction with a final goal to develop a model of the dressing task in order to recognize user's intentions and hazardous events during dressing. All the developments were performed on a Barrett WAM robot equipped with a Kinect camera for user tracking and a force sensor. The integration of hardware and software was done in Robot Operating System (ROS).

Contents

List of Figures

List of Tables

1 Introduction

The work in this thesis was performed as a part of a EU project called I-DRESS [2], whose goal is to develop a system that will provide proactive assistance with dressing to disabled users or users such as high-risk health-care workers, whose physical contact with the garments must be limited to avoid contamination. The proposed final robotic system will consists of two highly dexterous robotic arms, sensors for multi-modal human-robot interaction and safety features. An example of the robot-assisted dressing scenario is shown in Figure 1.

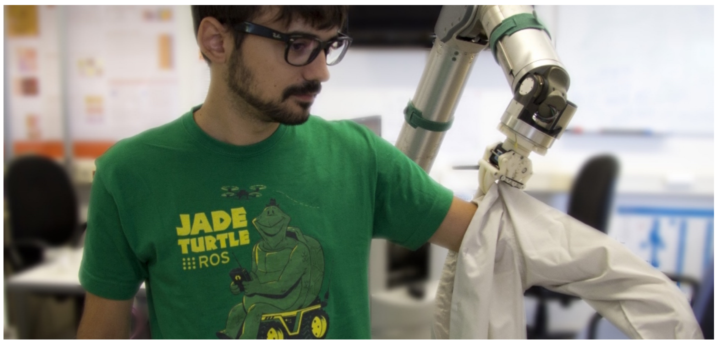

Figure 1: Example of the robot-assisted dressing scenario

The main goal of this work is to develop interaction abilities for the overall I-DRESS robotic system, which are based on user visual tracking and force and torque detection. The mentioned interaction modalities are used to recognize user's intentions while being dressed and detect any hazardous event that can occur during the dressing. The proposed system consists of a Barrett WAM robot equipped with a force sensor and a Kinect One camera. Implementation is done in Robotic Operating System (ROS) [3] running on Ubuntu platform.

The application scenario was divided in two tasks: jacket dressing and surgical gown dressing. The first task focuses on users with reduced mobility; the second task focuses on medical staff and is expected to reduce the possibility of contamination.

The project is divided into four phases. It started with literature overview and study of the state of the art. It was followed by training in necessary software and hardware tools available in the laboratory at IRI. Human-robot based on visual and haptic data was developed using these previously mentioned tools, which is the main contribution of this work. And finally the proposed interface and the associated model of the dressing task were tested on the real robot in experiments with users.

The Institut de Robòtica i Informàtica Industrial is research institute jointly founded by Universitat Politècnica de Catalunya and Consejo Superior de Investigaciones Cientificas (CSIC). It is involved in several world-wide projects including projects like the Socrates project and the Imagine project. Not only it focus its resources to research but also it is involved in some graduate courses from the UPC. As the IRI is using open source software, all the software they develop is open source too. So all the software developed for this project is made available to a wide community of robotics developers.

2 State of the art

Assistive robotics is a fast-developing field and many institutes and laboratories are allocating their resources to this area. The research in assisted dressing is relatively novel, most of the relevant research have been published in the last 5 years. They are summarized here.

In [4], Gao et al. purpose an online iterative path optimization using vision and force information. In the experiments, they show that with this method the robot finds the optimal path for different users after few iteration. However, this method was implemented only on a sleeveless jacket, since the dynamics of the clothes were not considered.

Chance et al. [5] proposed to differentiate between different layers of clothes during dressing using machine learning techniques. The experiments were limited to tests with a mannequin, without involvement of human users. At the end, the method they applied resulted in a 90 % accuracy n differentiating layers.

Additional aspects of the assisted dressing tasks were analysed in [6] and [7]. It is important to mention that this work also provides a novelty; it analyzes the ability of the robot to track user's arm joints under occlusion while the dressing task is performed. Also, the force and torque models presented here come as a result of experiments with real users.

3 Methodology

In the first part of the chapter all the hardware, software and its specifications are presented. The second part focuses on the developments and how the existing hardware and software tools were applied. The final part describes the implementation of the overall system in Robot Operating System (ROS).

3.1 Resources

Before doing any development, first, one has to get familiar with all the hardware and software expected to be used. To achieve this, I have done basic tutorials on C++, ROS, Kinect One and WAM robot programming. I performed a set of programming exercises, formulated as simple robot servoing problems.

3.1.1 Ubuntu

Ubuntu is a free distribution of the operating system (OS) Linux. It is commonly used as an environment for development of robotics applications because of its transparency and stability. Kinect camera is developed for Windows, but it's operation is also supported in Ubuntu. By using the Kinect in Ubuntu the complexity of the system is reduced because it avoids the use of an additional computer running Windows.

3.1.2 Robotic Operating System

Another advantage of using Ubuntu is better support for implementations in Robotic Operating System (ROS) which basically offers the possibility to interact easily with more than one device at the same time. Furthermore, the

ROS community has developed several packages containing functionalities such as hardware-drivers, datatypes, perception and other algorithms that were useful for the project. Moreover, the researchers from IRI had also developed several packages in the ROS framework that allow easier integration and operation of the hardware available in the laboratory.

To learn how to program in ROS, two different tutorials have been done. One was provided on the official ROS website, and the other one was developed at IRI. The tutorials introduce to the main concepts of ROS and how to use them: nodes, topics, publishers, subscribers, servers, actions, etc. [8, 9].

3.1.3 Programming Language

The ROS framework offers the possibility for development in two different programming languages: C++ and Python. The programming language chosen is C++. It is true that programming in Python is simpler, but on the other hand, using C++ is preferable because it is a language easy to compile, which means that the execution of the nodes is faster than in a interpreter language. This feature is interesting when one wants to synchronize more devices in real time. On the other hand, using C++ is a good opportunity to learn a new programming language as Python was already used in some university courses. To learn about C++ the tutorial of cplusplus website [10] has been done. This tutorial goes through the basic concepts of C++.

3.1.4 Kinect One and OpenNI

To perform the posture recognition it is necessary to know the position of the user, i.e., its joints. This interaction is done through Kinect One[1] and OpenNI

[1]Kinect One website: http://www.xbox.com/en-GB/xbox-one/accessories/kinect

software[2].

Kinect One, also known as Kinect v2, is a sensor equipped with a RGB camera, depth sensor and a microphone array (see Figure 2). What makes it attractive to use it in this project is its low price (it was designed for video-games console) and the combination of the RGB camera and the depth sensor because combined with the appropriate software one can track and obtain joints positions of the users in front of the camera. The depth sensor has a detection range between 0.5m and 4.5m and a viewing angle of 70° for the horizontal axis and 60° for the vertical one. Kinect is connected to the computer with a USB 3.0 so a computer with such port is needed.

The OpenNI [11] is an open source software used to develop middleware libraries and applications. Specifically, it reads the data from sensors and publishes it inside the ROS framework. In this case it uses a ROS topic to publish positions of the user in the camera frame of reference.

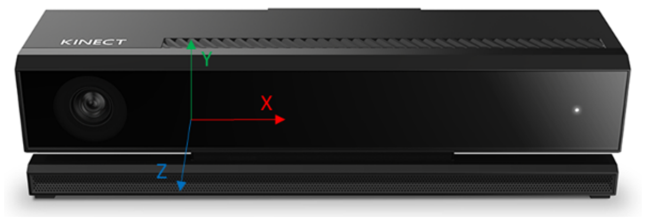

Figure 2: Kinect 2 camera [1]

3.1.5 Barret WAM robot

The assisted-dressing motion is performed by a Barret WAM robot shown in Figure 4. It is a 7-DOF robot with a workspace that has a close-to-spheric shape of approximately 1m radius. The high number of degrees of freedom allows the robot

[2]OpenNI website: http://openni.ru/index.html
[3]http://forum.openframeworks.cc

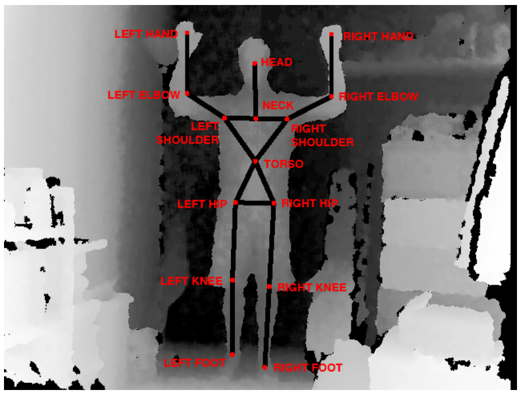

Figure 3: OpenNI real time skeleton tracking[3]

to reach any point of the work-space with the end effector in any orientation. Also it offers the possibility to work with different degrees of stiffness, which makes it suitable for scenarios such as assisted dressing that involve physical human-robot interaction (HRI).

As shown in Figure 4, the WAM robot requires a gripper to be attached to the last joint to be able to grasp the garment. A custom-made, 3D printed gripper was developed in the IRI laboratories for this work.

More information about the specifications of the robot can be found in its webpage [12].

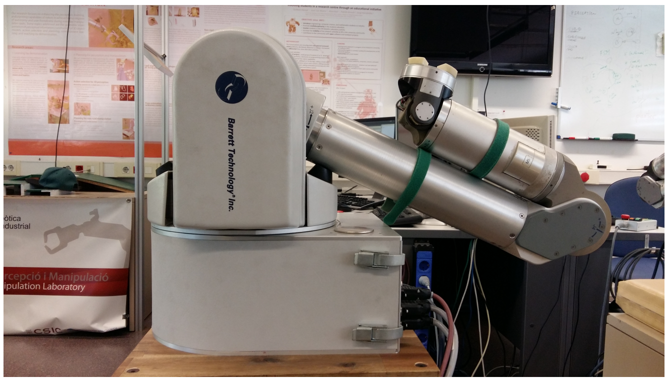

Figure 4: Barrett WAM robot in the IRI lab

3.1.6 Force sensor

The ATI force sensor was used for the assisted dressing task. This sensor is a 6-DOF sensor, which means it provides force and torque values for each axis. A driver developed at IRI waas used in this work. The sensor is connected to a computer via Ethernet. The gripper used has four fingers able to grasp clothes and can be opened and closed using a servo motor. The force sensor can be attached to the gripper, as shown in Figure 5.

As you can see in the following chapters, the force expected during a regular dressing will be lower than 20N in absolute value and the maximum force recorded during a snagging scenario is around 45N. As the limit of this sensor is around 700N for X and Y axis and 900N for Z axis, it fulfills the requirements for this task. Regarding the torque, the values obteined during the dressing are lower than 2 Nm which is also within the sensor's range. More details about the force sensor specifications can be found in Appendix A.

It is important to notice that the axes of the sensor are not the same as the robot ones. Also, the sensor frame of reference is moving around the space during the dressing so it is necessary to transform the values obtained in the sensor frame of reference to a static frame of reference in order to have useful data. Otherwise the data obtained won't be useful.

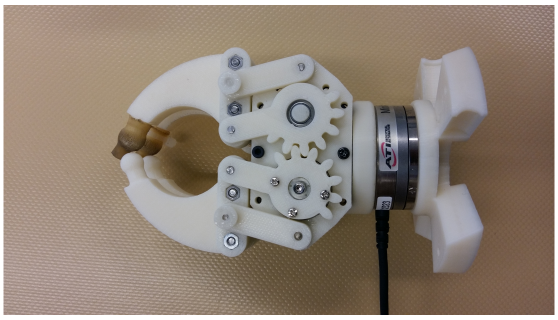
Figure 5: Force sensor attached to the gripper

3.2 Project development

3.2.1 Assisted dressing scenario

In the scenario one can differentiate three different elements: the Kinect One camera, the Barret WAM robot equipped with the force sensor and holding a garment and the user (see Figure 6). The distance between the robot and the camera is 3m, chosen empirically for optimal detection.

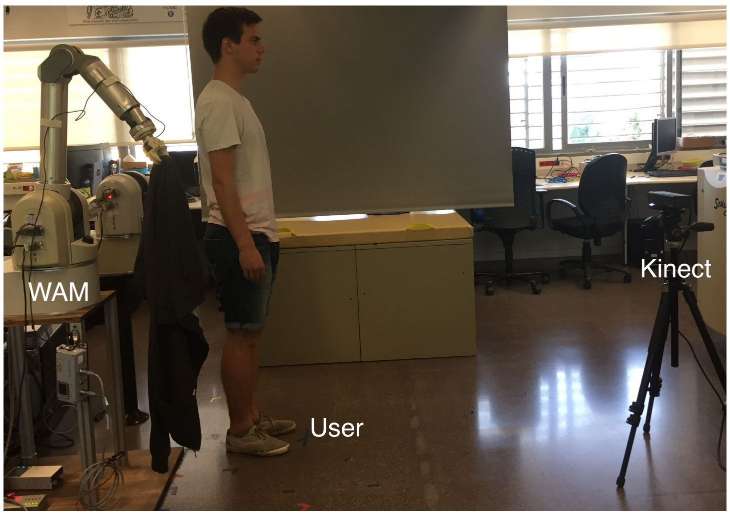

Figure 6: Dressing scenario

To simplify the implementation, the dressing task is decomposed in 4 different segments: *waiting for user*, *waiting to dress*, *dressing* and *finished*. In the first segment, body posture recognition via Kinect is used to let WAM know when the user wants to be dressed. The key posture will be one of the arms extended as shown in Figure 7. The WAM will be waiting at the home position holding the jacket until Kinect detects the key posture. Then, the robot gripper will approach the extended hand and the state will be changed to *waiting to dress*. In this state, the user is supposed to put his hand inside the sleeve and then, using a force sensor, detect such action and change the state to *dressing*. During the *waiting to* *dress* state the robot is supposed to stay in a fixed position, and just go to another position if the user does a huge posture change (depending on this change the robot should go another time near the hand or to the home position). During the *dressing* state the robot will move slowly up towards the shoulder. In this state, we want to use a force sensor to detect if the task is progressing properly;for this, it is necessary to define a force and torque profile of a successful dressing task. The state will be changed to *finished* when the robot is near the shoulder and a force in the horizontal direction is detected. In the last state, *finished*, the WAM robot should open the end effector, i.e., the gripper, to release the grasp of the garment and return to the home position.

3.2.2 Human-robot interaction

In this section, human-robot interaction modalities developed in this work are described, namely, visual tracking and force and torque detection.

User tracking and posture recognition

Simple body posture recognition is used to define when the robot should start the assisted-dressing task and when it shouldn't move. Key postures are the right arm extended and the left hand raised, respectively (see Figure 7). Also, robot trajectory will be specified in terms of the positions of the hand, elbow and shoulder. To perform such interactions the joint positions provided by Kinect v2 camera are used. Preliminary tests ensure that the upper-body tracking was sufficiently reliable to perform this task.

To recognize the postures explained previously, the elbow angle, the armpit angle and the angle between the arm and the body plane (assuming the user has always his body orientated to the camera) will be used. When measuring a body angle, we get some variability due to Kinect noise. So, assuming this variance, an interval

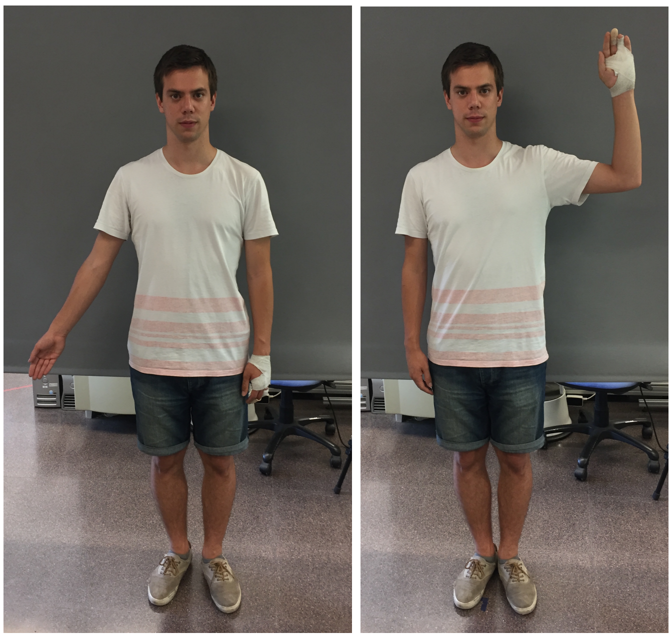

Figure 7: Postures. Left: extended arm, Right: hand raised

of confidence is set for each angle to ensure that the position is recognized (Table 1).

Angle	Extended arm	Rise hand
elbow	135^o-no maximum	70^o-110^o
armpit	30^o-80^o	70^o-110^o
arm-normalbody	60^o-120^o	60^o-120^o

Table 1: Posture recognition: angle range.

One disadvantage of using Kinect through Ubuntu platform is that it doesn't offer the orientation of the different joints and any detection reliability value, as it does when using the manufacturer's library for Windows. This information is not necessary but it would make the programming phase simpler.

Force sensor implementation

To achieve the physical HRI the ATI force sensor has been used. To integrate it with WAM robot and Kinect a ROS node developed in IRI was used. This node takes the data from the sensor and publishes it to a ROS topic. So, for the implementation and use of the force sensor data, a node that subscribes into this topic should be created. The topic message contains 2 vectors of 3 dimensions containing the force and torque values along each axis. The force sensor driver doesn't offer the possibility to zero it once it is run. To solve this issue, one just need to do extra calculations during the dressing that makes the code a little bit more complex.

The sensor will be used during the *dressing* and *finished* states, specifically to determine three different cases. First, to determine when the dressing should start. Secondly, to check if the dressing is progressing correctly. Finally, to define when the task is over. During the *waiting to dress* state the robot will approach the hand at a safe distance and wait for the user to start the dressing task by putting the hand inside the sleeve. While doing this action a gain on the force module will be detected, which indicates that the dressing started. During the dressing, a force value above a certain threshold, suggests that the garment got stuck or that the user has changed body posture. Also it will be used to make corrections to the trajectory during dressing to allow smoother performance and improve user's comfort. And finally, when the sensor detects a force in the horizontal direction at a shoulder level, this would indicate that the task is finished.

Sensor-to-robot frame transformation

Before doing any measurements with the force sensor, the pose of the force sensor is changing all the time during the dressing so it is important to associate the force sensor frame of reference with a fixed frame such as robot's base. As the last joint of the WAM robot has its own frame a transformation between this frame and the force sensor is sufficient.

It is important to make sure that the transformation between the force sensor frame of frame of reference and the WAMs' frame of reference is correct. To do such thing, WAM has been fixed in a certain position and then a force on the end effector has been applied. First on the x axis, then on the y axis and finally on the z axis (in the robot frame of reference). The results are plotted in Figure 8. The force plot shows that the force recorded was at the same direction where we applied it so we can conclude that the transformation works properly, although some noise due to user and sensor error can be noted along the remaining two axes.

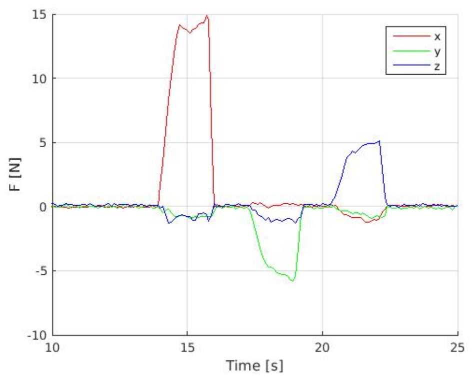

Figure 8: Force transformation validation

3.3 Kinect-WAM calibration

In order to be able to use the different body joints positions to guide the WAM robot calibration between the robot and the camera is needed. The transformation was done using the ROS TF package [13]. Doing in this way we are now able to obtain user joints positions in the robot frame of reference without extra computing.

To do the calibration between the Kinect v2 camera and the WAM robot a node has been written. This node, using a body frame that is tracked with few noise (even if it has the robot near it) like the shoulder and the rotation between the camera and the WAM will give you the exact distance between the robot world frame and the camera global-space frame. One just has to be in front of the Kinect and put the end-effector in the same position of the body frame, in this case the shoulder (like in Figure 9). Then in the screen where the node is run the values for the transformation are displayed.

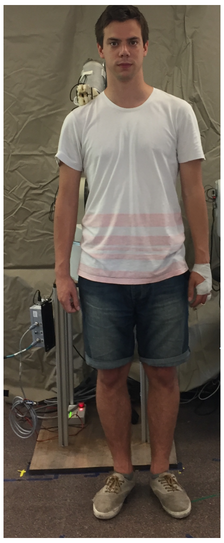
Figure 9: User position during calibration

3.4 WAM control

To control WAM robot some nodes developed at IRI will be used. The first one is an action client-server called dmp-joint-tracker. It is necessary to activate WAM and basically in order to move WAM robot a message containing the rotations for each robot joint (a total of seven) has to be published to its topic (called

DMPTrackerNewGoal). The second node provides a service containing the inverse kinematics of the robot. By giving a 3D point in the robot frame of reference and a certain rotation expressed in quaternion it calculates the rotation of each of the seven WAM joints that should be applied in order to reach that point with the end effector in the quaternions direction (if possible). When sending the robot to a new goal if the action server is activated, it takes the data from the topic and makes the robot move automatically, otherwise the robot won't move. If you try to send the robot to a point out of the workspace, the robot not move.

For safety reasons, WAM is equipped with a control pendant (Figure 10). The pendant is necessary to activate WAM after running the launch file from the Ubuntu terminal, in this way, WAM can't be activated by accident. Also it has an emergency stop button which will stop the robots actuators instantly providing enough power to fall slowly.

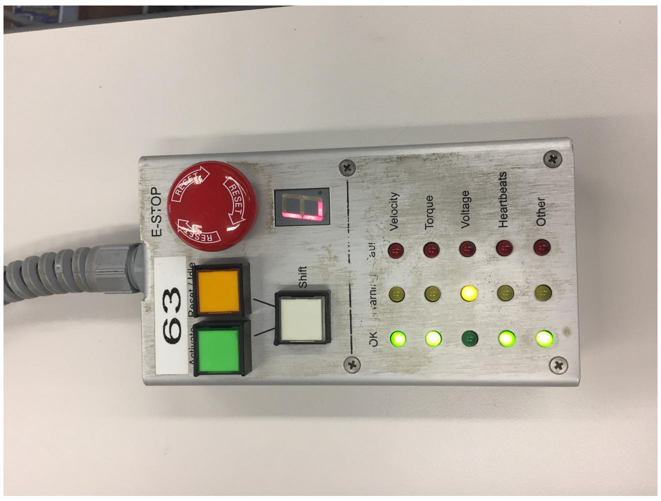

Figure 10: WAM pendant

3.5 ROS Implementation

To implement the task in the ROS framework some nodes have been developed. Several nodes developed at IRI are necessary to operate the robot.

First of all, a node performing the posture recognition was developed. This node uses the information of the left and right shoulder, elbow and hand positions in order to define the angles between these joints. When the detected angle was within the range of values of a certain posture, a message of the boolean type True was posted to the topic dedicated for the recognition of the respective posture. Otherwise it published the boolean False.

A second node allowed the robot following of the right hand. This node is activated only when the left hand is not raised and the right arm is extended, i.e, if in the respective topics a False and a True value are published. This node has two subscribers, one for each topic, and also uses Kinect to obtain the hand position. For safety reasons, the robot is sent to 5 cm from the user's hand (in the forearm direction).

An extra node to make easier the communication with WAM robot was created. By publishing to a certain topic a message containing the 3D point and the gripper orientation desired it makes the WAM robot move to this position using the nodes explained in the previous section. It is based in one node already created in the IRI repository.

To run the system, the described nodes must be running along with the supporting algorithms: the Kinect tracker, the sensor and WAM robot drivers. It is important to mention that two different personal computers are needed: one running WAM robot drivers and another one running the force sensor, the Kinect tracker and the implemented algorithms. The two are connected via Ethernet.

4 Experiments and results

To be able to propose a model of the dressing task based on force and torque measurements, some initial experiments were performed. to define the range of values for a normal operation for different segments of the dressing task, and detect common patterns.

Furthermore, to be able to guide the robot from the hand to the elbow and finally shoulder, it was important to find out the reliability of user tracking during dressing (due to occlusions).

4.1 Experimental setup

The proposed experimental setup consist in performing the dressing along a pre-recorded trajectory. During the experiment force and torque measurements from the force sensor and positions of the arm joints in the robot frame of reference were recorded. Of course, the Kinect, WAM robot and force sensor drivers were used too to perform the experiments.

First the kinesthetic teaching has been used to record a dressing trajectory. Kinesthetic teaching lets you record a trajectory into a .csv file by moving the robot with your hands, and its points are stored in a .csv file. The trajectory follows the arm angle and performs a final push at the shoulder height to pull the garment over the shoulder (see Figure 11). Additional ROS node was developed for the experiments that reproduces a trajectory from the .csv file and stores the measurements obtained from the force sensor and the Kinect and save them into two different text files. It also stores the position of the end effector and the time when each sample was taken. The position of the user with respect to the robot and the camera is shown in Figure 6.

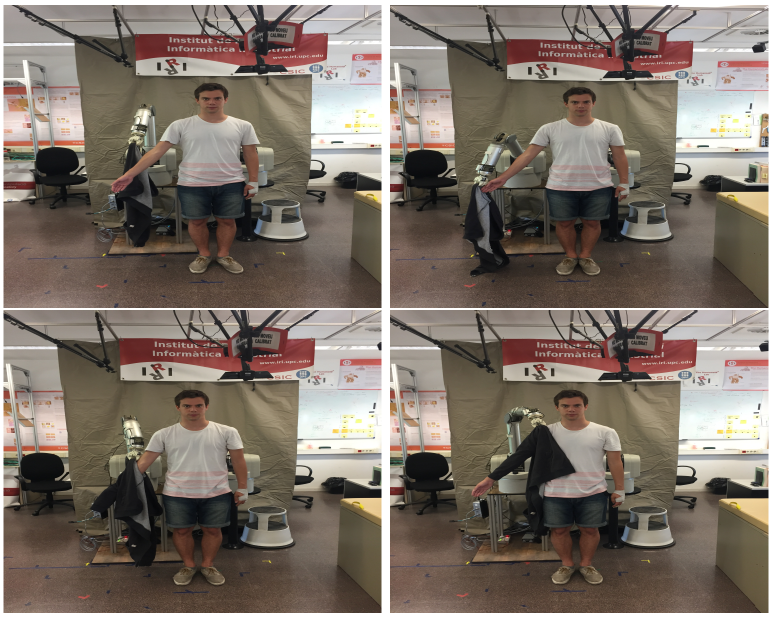

Figure 11: Dressing sequence

The experiments were performed with 3 users of approximately the same height (183 +/- 2 cm) in order to be able to use the same pre-recorded trajectory. Each user performed 10 trials.

The data from the force sensor were used to define force and torque profiles of the dressing task. On the other hand, the data from the Kinect provides insight into the reliability of user tracking during the dressing. Also, the position of the WAM end-effector was compared to the positions of the hand, elbow and shoulder joints detected by Kinect, to recognize the segment of the dressing task and associate the force and torque measurements with that segment.

Finally, 20 additional experiments had been done: 10 performing the dressing without user (simulating the case when user's hand misses the sleeve) and 10 more without a user and without garment (to understand the level of inertial forces caused by the robot motion and possibly detect if the garment was accidentally dropped by the robot). This will let us define a profile for this two different cases and compare them with the real case scenario when a user was involved. Additional experiments were performed simulating a snagging of the garment.

It is important to mention that in the trajectory recorded (Figure 12), the WAM robot is moving along the Y-Z plane. More significant and representative forces measurements were detected along these axes.

4.2 Results analysis

Matlab Software was used to obtain relevant statistical values such as means, variances and confidence intervals. These results were also plotted in Matlab.

First, the position of the WAM end-effector was compared to position of the hand, elbow and shoulder joints to label these segments on the plots. The timestamps associated with different segments are shown in Table 2. The timestamp for the final push has been obtained through analysis of WAM trajectory. Please note that the start of the dressing task is at 3.6s.

	Begin	Hand	Elbow	Shoulder	Push	End
time [s]	3.6	4.2	6.48	8.5	9.9	10.7

Table 2: Timestamps

4.2.1 Variability and stability of the joint positions analysis

Assuming that the user maintained the position during all the experiments, the variance for each joint and each axis along the time has been calculated. The results are the plots in Figure 13. As during the dressing the robot is moving really close to the user, it is considered that a standard deviation greater than 4

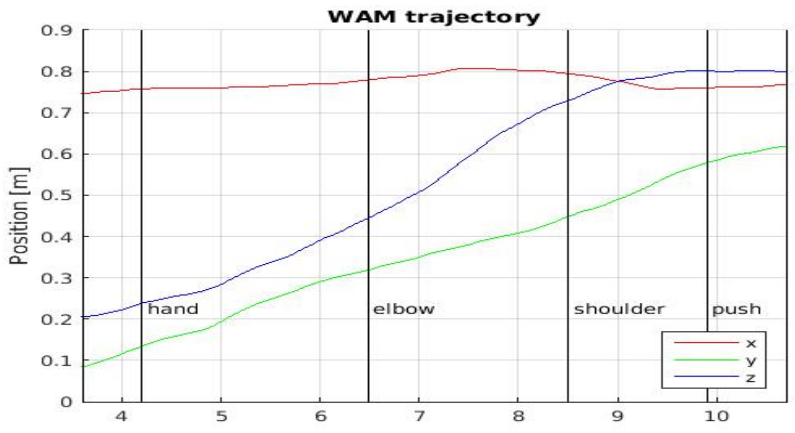

Figure 12: WAM trajectory

cm is too big to rely on this joint. Using a joint with such variance can lead to hit or stretch the user.

As seen in Figure 13, the variation of the hand and elbow position along all three axes goes over the threshold and that the position of the shoulder is below this threshold. So one can conclude that the shoulder joint, unlike the hand and the elbow joints, can be used as a robot guiding point during the dressing.

4.2.2 Force and torque analysis

Force and torque mean values for each user along each axis were computed. The measurements from all three users are shown in Figure 14.

Analyzing the data, we can assume that the maximum force in absolute value for the X axis will be around 4.58, for the Y axis 15.13 N and for the Z axis 11.24 N. The maximum torque value in absolute value for X axis is around 1.443 Nm, for Y axis 0.586 Nm, and for Z axis 0.331 Nm.

Missing sleeve and no jacket test

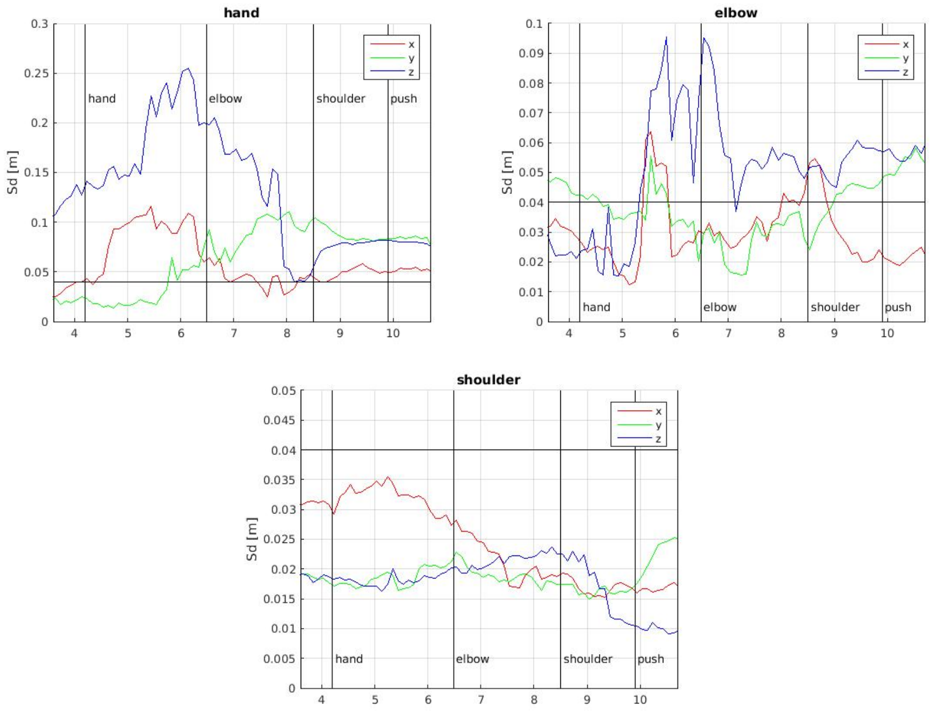

Figure 13: Joints standard deviation

Detection of the missed sleeve error during dressing will be detected by comparing the actual force and torque measurements with the ones obtained from the model. The separation of confidence intervals (when they do not overlap) was used as the metric for successful detection of the missed sleeve error. The objective is to detect the error in the earliest stage of the dressing task. The obtained results are shown in Figures 15, 16 and 17 and Tables 3, 4, 5 and 6.

Force	Fx (in %)	Fy (in %)	Fz (in %)	Time y	Time z
User 1	37.5	87.5	68.055	4.307	5.407
User 2	40.277	84.72	65.277	4.407	5.007
User 3	48.611	86.11	90.277	4.607	4.407
Average	42.129	86.11	74.5370	4.441	4.941

Table 3: Missing sleeve force: Overlap intervals and earliest detection times

Torque	Fx (in %)	Fy (in %)	Fz (in %)	Time y	Time z
User 1	83.333	63.888	79.166	4.307	5.407
User 2	72.222	59.722	79.166	4.407	5.007
User 3	88.888	77.777	76.388	4.607	4.407
Average	81.481	67.129	78.240	4.441	4.941

Table 4: Missing sleeve torque: Overlap intervals and earliest detection times

Force	Fx (in %)	Fy (in %)	Fz (in %)
User 1	91.666	100	100
User 2	84.722	100	100
User 3	84.722	100	100
Average	87.037	100	100

Table 5: No jacket force: Overlap intervals

Torque	Fx (in %)	Fy (in %)	Fz (in %)
User 1	100	100	88.888
User 2	88.888	100	86.111
User 3	100	100	87.5
Average	96.296	100	87.5

Table 6: No jacket torque: Overlap intervals

As seen from the results, for Y and Z axis there is significant separation of confidence intervals (specially for the torque). The earliest instance when you can find significant differences between the force profile and the missing sleeve is at 4.4 s for Y axis and 4.9 s for Z axis (for both force and torque). Remembering that the dressing begins at 3.6 s, the real times are 0.8 and 1.3 s respectively. Regarding the test without jacket, as you can see directly from the plots from Figure 16 and 17, there is no coincidence between both profiles at any time. In conclusion, if the robot is not holding the jacket the system can detect directly such thing, on the other hand, it the user misses the sleeve, looking at Y axis, the system will detect such event in less than 1 s.

From the plots, it is also possible to detect when the final push happens, which is important to recognize the measurements at the end of the dressing task. For the force on the Y axis you can see that it produces a local minimum and on the Z axis it produces a global maximum.

Snagging test

From the plots on Figures 18, 19 and 20 the force and torque values obtained from the snagging are completely out of range of values for the normal operation. So it is an event easy to identify. On the Z axis the maximum value obtained is between 40 and 50 N while it should be less than 5N. On the Y axis the difference is not that huge but it is noticeable too: the value obtained is between 5 and 10N and it should be lower than 4N.

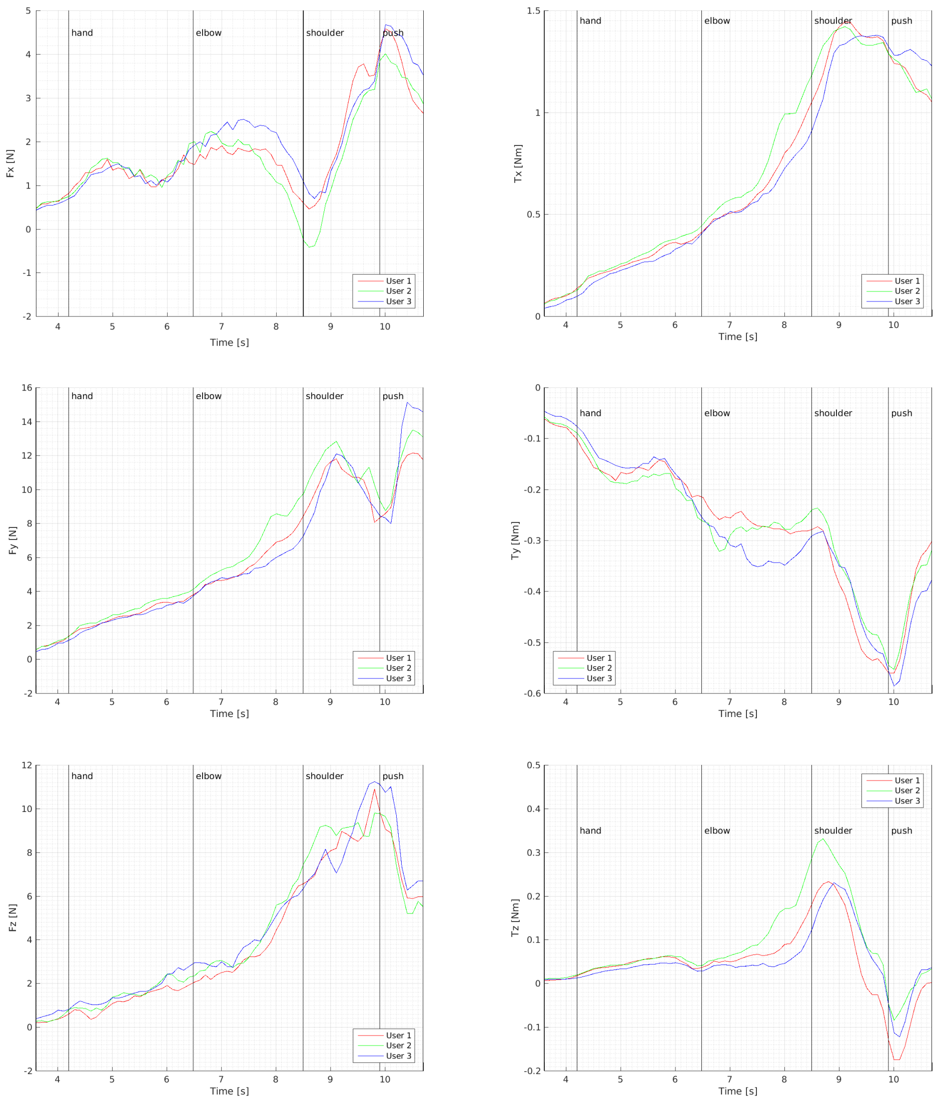

Figure 14: Force and torque averages

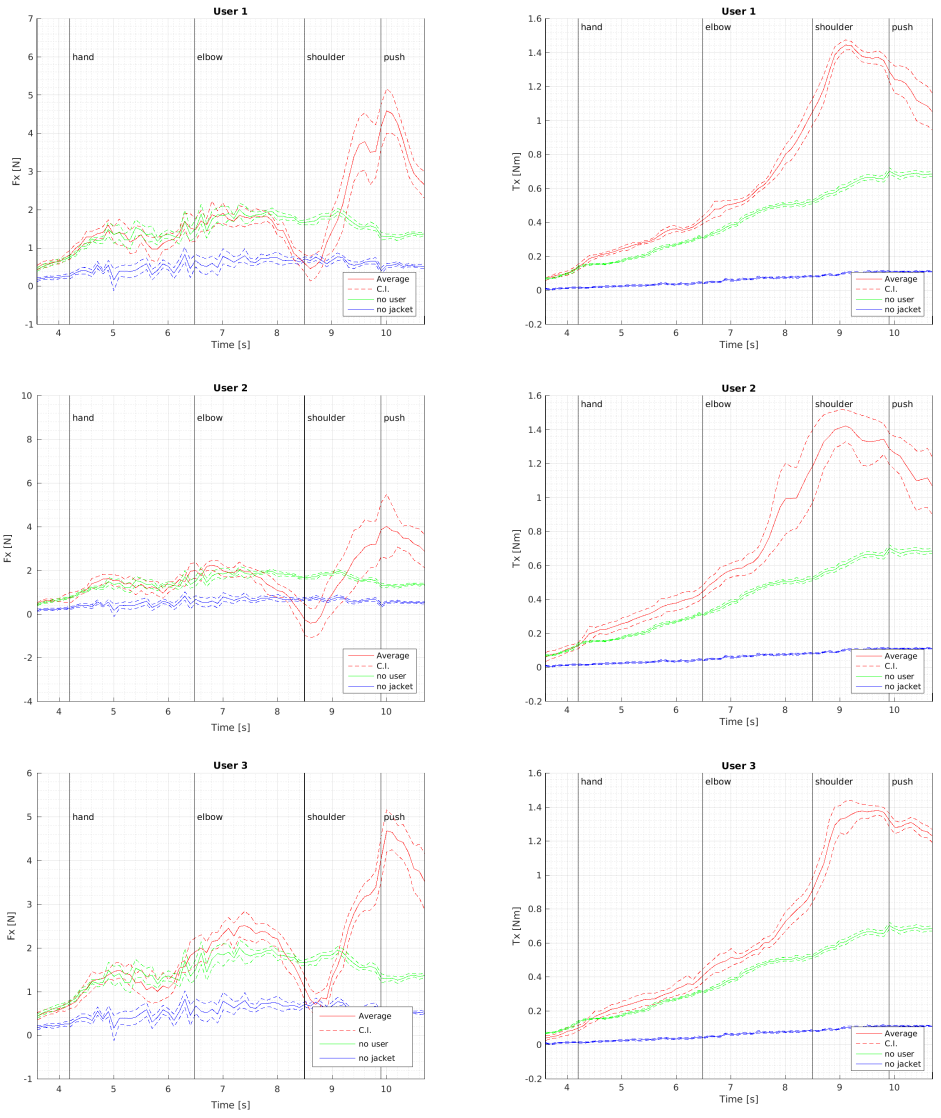

Figure 15: Missing sleeve and no jacket: Force and torque X axis

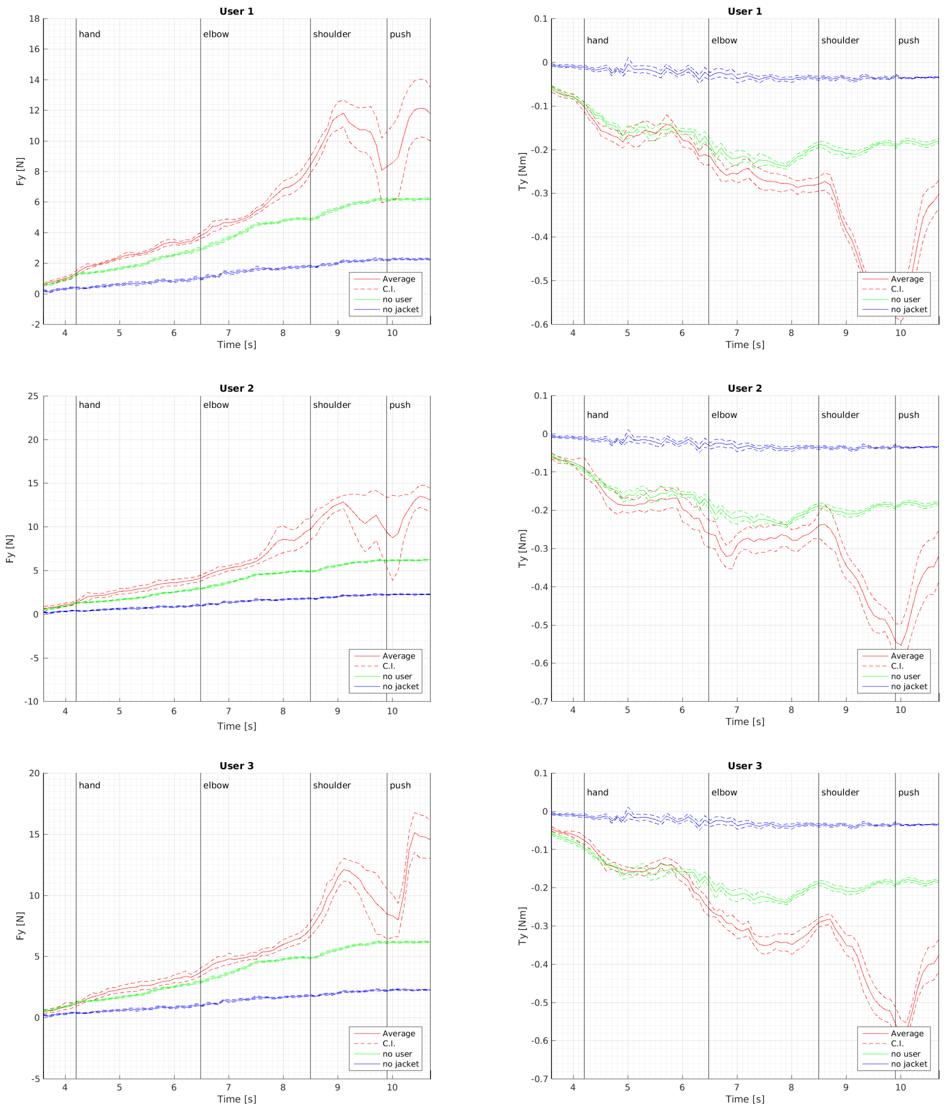

Figure 16: Missing sleeve and no jacket: Force and torque Y axis

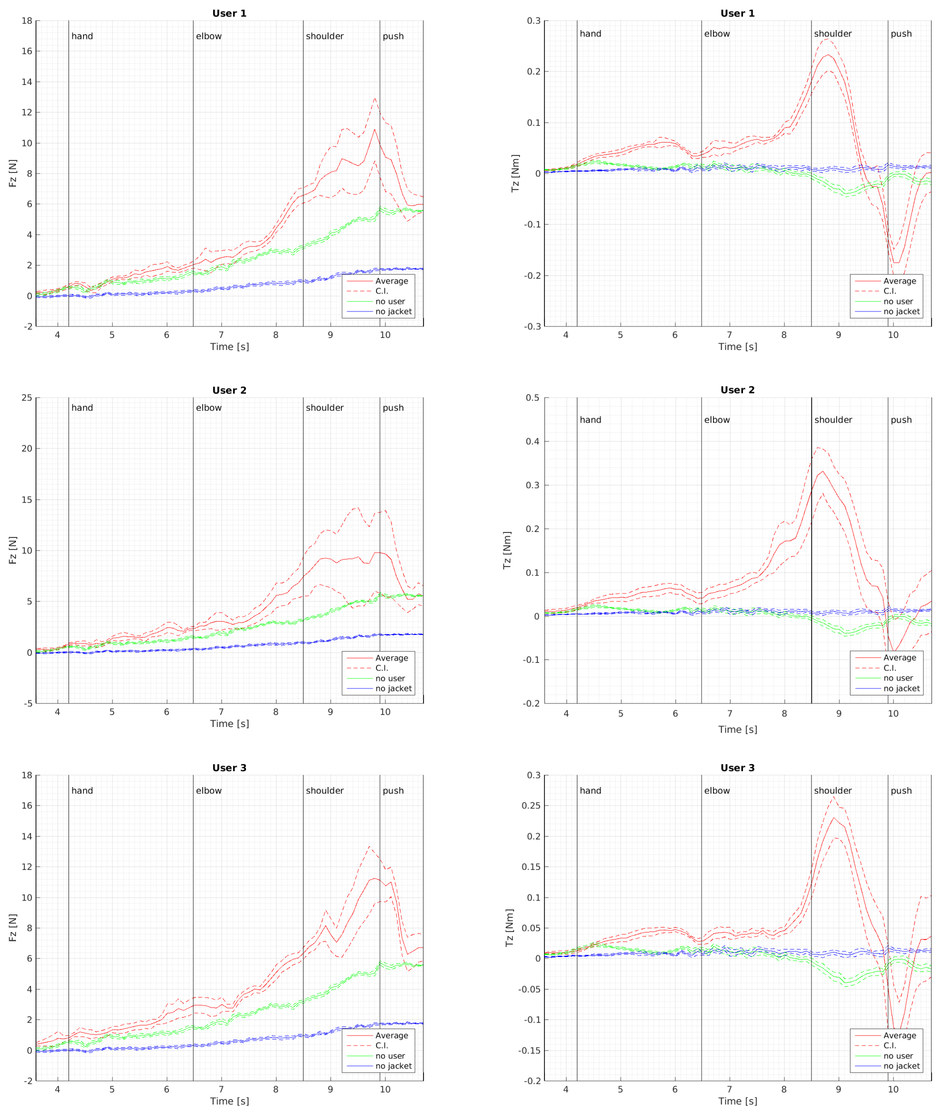

Figure 17: Missing sleeve and no jacket: Force and torque Z axis

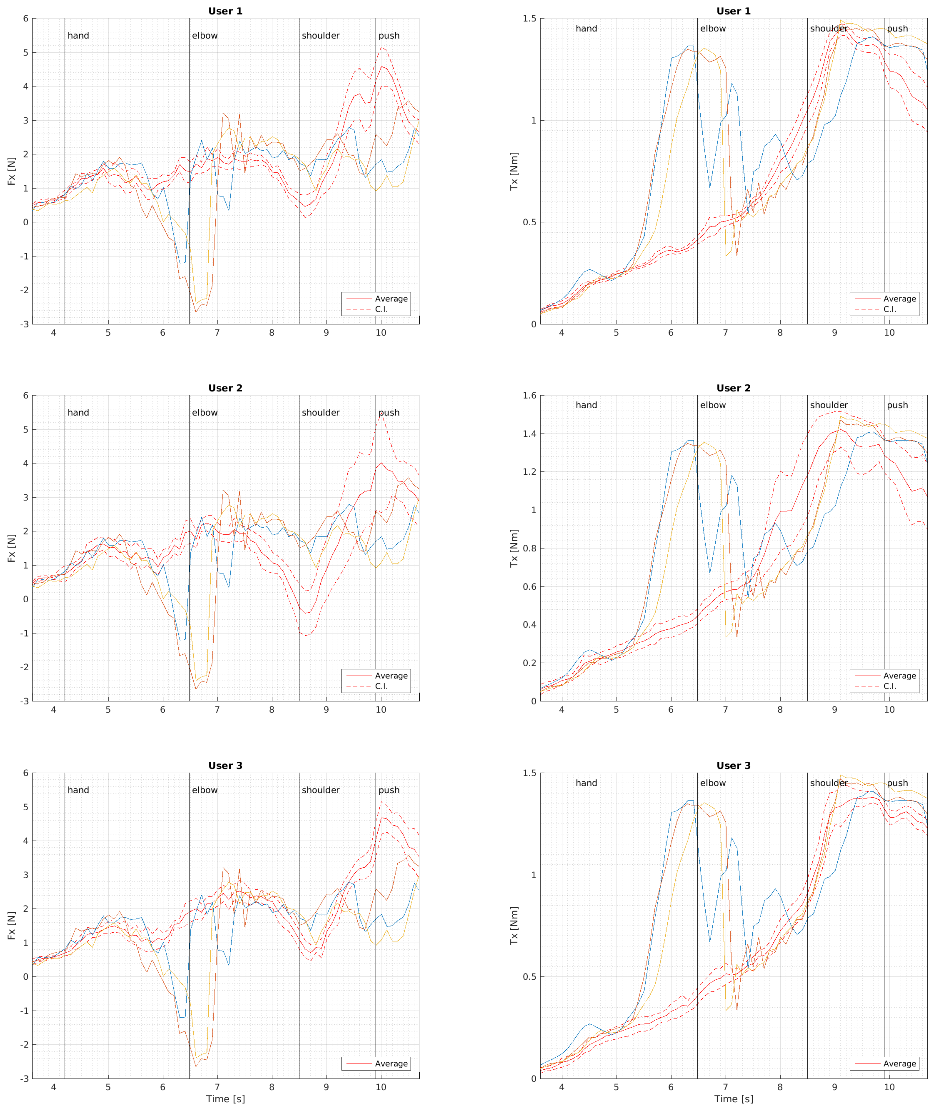

Figure 18: Snagging: Force and torque X axis

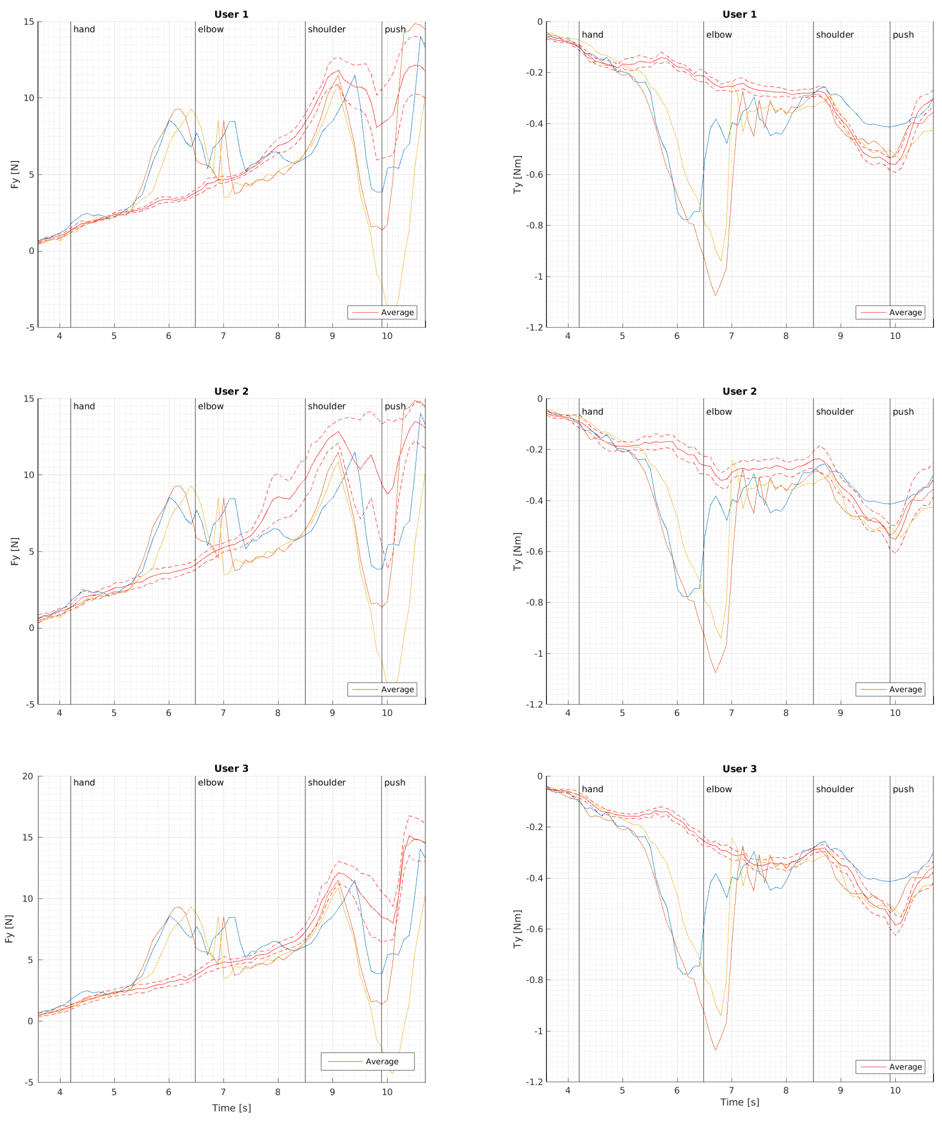

Figure 19: Snagging: Force and torque Y axis

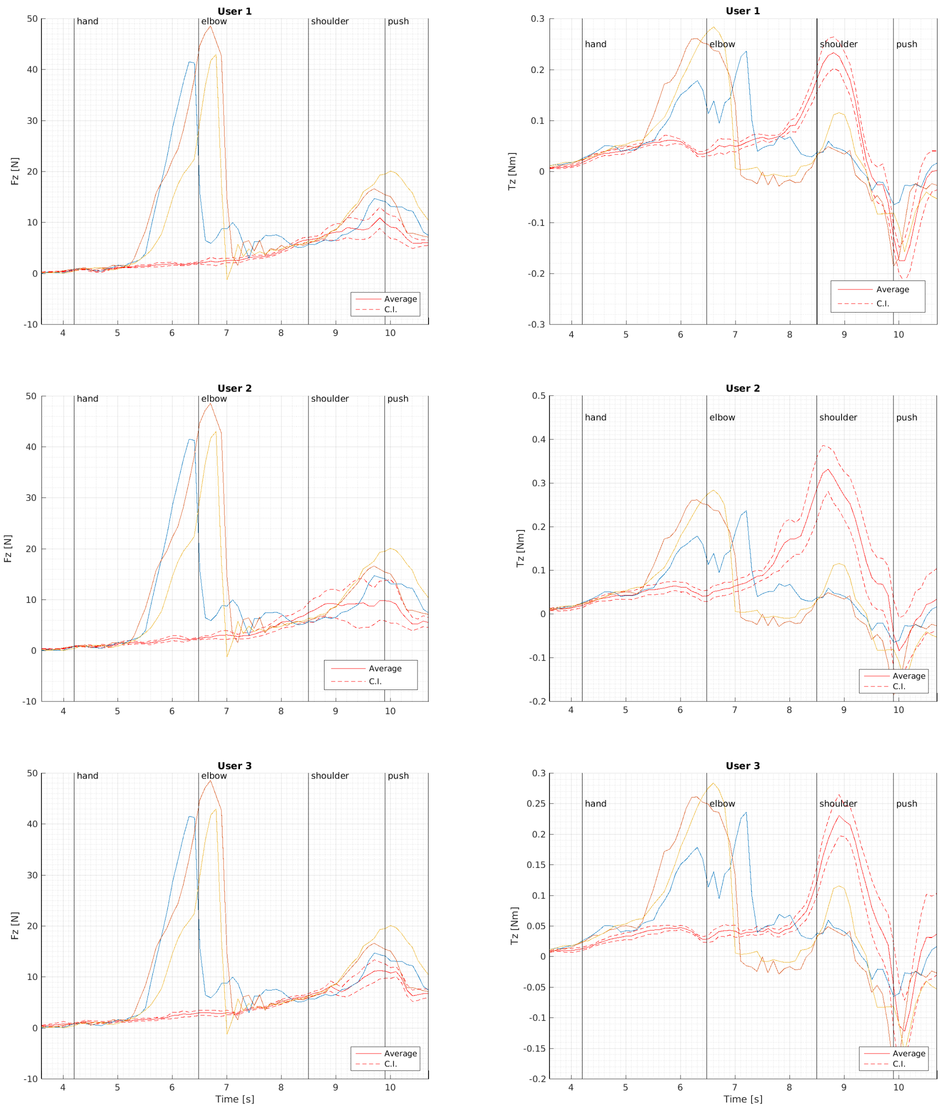

Figure 20: Snagging: Force and torque Z axis

5 Budget

In this chapter the cost of the hardware, software and human resources is analyzed. Remind that is just a estimation, so the real cost of the project may differ from the calculated.

As human resources, I've been working as a part time job for 4 months what means a total amount of 340 working hours approximately. On the other hand, the supervisor has work on the project on average 5 hours a week, what makes a total amount of 86 working hours. Discriminating by role, the total costs associated to this two workers are presented in Table 7. It is supposed that both, engineer and project manager, are payed just for the amount of hours expected to do and not the ones invested at the end.

Role	Salary (per hour)	Working hours	Total wage
Project manager (Supervisor)	58,96€	86	5070,56€
Industrial engineer	22,43€	340	7626,2€
Total	-	-	12.696,76€

Table 7: Human resources costs

The estimated amortizations for each device used during the project is shown in Tables 8 and 9.

Regarding the software resources, most of them (Ubuntu, ROS, ROS algorithms, Latex) are open source software and free. The unique one to consider is Matlab. The total cost associated to the software resources is shown in Table 10.

[4]Price per hour = (Unit price)/(Amortization period * 250 working days per year * 8 hours a day)

Resource	Units	Unit price	Amortization period	Price per hour[4]
Laboratory PC	2	1.000€	4 years	0.125
WAM robot	1	97.500€	10 years	4,875
Force sensor	1	6000€	4 years	0.75
Kinect v2	1	99,99€	2 years	0.025
Total	-	-	-	-

Table 8: Hardware resources costs. Part 1

Resource	Hours of use	Amortization
Laboratory PC	340 + 60	50€
WAM robot	60	292,5€
Force sensor	40	30€
Kinect v2	80	2€
Total	-	374,5€

Table 9: Hardware resources costs. Part 2

Resource	Units	Unit price
Ubuntu	1	0€
ROS	1	0€
ROS algorithms	1	0€
Latex	1	0€
Matlab	1	500€
Total	1	500€

Table 10: Software resources costs

As the laboratories are part of the Facultat de Matemàtiques i Estadística, the network resources are shared with all the users from the faculty what makes difficult to make a real estimation of the general expenses. However, it is estimated that the cost of general expenses is negligible compared with the total expenses.

In conclusion, the total cost of the project is around 13.571,26€

6 Conclusions and future work

The work presents a study of visual and haptic human-robot interaction during the robot-assisted dressing task. The work consisted of several stages that included the analysis of the state of the art, developments, software and hardware integration, and finally, the experimentation with users.

The main achievements include force and torque models of the dressing task, and the analysis of the reliability of user tracking during dressing. The work sets the base for the final scenario, of performing a complete dressing task with lay users and detecting dressing errors that may occur, such as missed sleeve or garment snagging. The results obtained confirm the idea that it is possible to propose a model based on force and torque measurements. Additionally, simple posture recognition has been developed as well as a hand-following algorithm that are used as a proof of concept and will further be developed as part of the future work.

The presented work was performed as a part of the I-DRESS project, and it makes an important contribution to its analysis of multimodal human-robot interaction.

As the project is not finished yet, there is still lot of work to do. As future work, more tests with different users in different postures are needed to be able to propose a more complete model. Also, the framework can be adapted to other scenarios such as a surgical gown dressing.

The possibility of using two WAM robots to perform the dressing task was proposed but the software to synchronize both robots in real time is still under the development phase.

Finally, working on this project has been a great opportunity to learn about robotics. I've also improved my programming skills in C++ and ROS, and I learned how to work with robot hardware. I improved my programming skills in Matlab and I've learned how to write scientific documents in Latex. Work at IRI has been useful not only to write this thesis but also to gain work experience and knowledge that will allow me to continue with my studies at a Master level, but also be a competitive candidate for jobs in industry..

Appendix A

Table 11: ATI force sensor specifications.

Single-Axis Overload	
Fxy	$\pm 810N$
Fz	$\pm 2400N$
Txy	$\pm 19Nm$
Tz	$\pm 20Nm$
Stiffness (Calculated)	
X-axis and Y-axis forces (Kx, Ky)	$1.1x10^7 N/m$
Z-axis force (Kz)	$2.0x10^7 N/m$
X-axis and Y-axis torque (Ktx, Kty)	$2.8x10^3 Nm/rad$
Z-axis torque (Ktz)	$4.0x10^3 Nm/rad$
Resonant Frequency	
Fx, Fy, Tz	$3200Hz$
Fz, Tx, Ty	$4900Hz$
Physical Specifications	
Weight	$0.0499Kg$
Diameter	$40mm$
Height	$12.2mm$

References

[1] Kinect SDK tutorial. http://homes.cs.washington.edu/~edzhang/tutorials/kinect2/kinect3.html, June 2017.

[2] I-DRESS Project. https://i-dress-project.eu/presentation, June 2017.

[3] ROS. http://wiki.ros.org/, February 2017.

[4] Y. Gao; H.J. Chang; Y. Demiris. Iterative Path Optimisation for Personalised Dressing Assistance using Vision and Force Information. In *International Conference on Intelligent Robots and Systems*, pages 4398–4403, 2016.

[5] G. Chance; A. Jevtić; P. Caleb-Solly; S. Dogramadzi. A Quantitative Analysis of Dressing Dynamics for Robotic Dressing Assistance. In *Frontiers in Robotics and AI*, pages 1–14, 2017.

[6] G. Chance; A. Camilleri; B. Winstone; P. Caleb-Solly; S. Dogramadzi. An Assistive Robot to Support Dressing – Strategies for Planning and Error Handling. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 774–780, 2016.

[7] A. Clegg; J. Tan; G. Turk; K. Liu. Animating Human Dressing. In *Siggraph*, pages 1–9, 2015.

[8] Official ROS tutorial. http://wiki.ros.org/ROS/Tutorials, February 2017.

[9] IRI ROS tutorial. http://wiki.iri.upc.edu/index.php/LabRobotica_Tutorial_Hello_World, February 2017.

[10] cplusplus tutorial. http://www.cplusplus.com/doc/tutorial/, February 2017.

[11] OpenNI. http://openni.ru/index.html, June 2017.

[12] Barret Technology. http://www.barrett.com/products-arm-specifications.htm, March 2017.

[13] ROS tf package. http://wiki.ros.org/tf, March 2017.