Index of the Project

**Part I: Project memory** ........................................................................................................................................... 2

**Chapter 1: Introduction** ........................................................................................................................................ 3

1.1 Study of existing technologies related to the project ...................................................................................... 3
1.2 Motivation for the project ............................................................................................................................... 5
1.3 Objectives .................................................................................................................................................... 6
1.4 Explained solution ....................................................................................................................................... 7
1.5 Resources/tools used ................................................................................................................................... 24

**Chapter 2: Results and experiments** .................................................................................................................. 25

**Chapter 3: Conclusions** ....................................................................................................................................... 30

**Chapter 4: Future improvements** ..................................................................................................................... 32

**Bibliography** 34

**Part II: Code** 35

**Code** 36
Figure’s index:

Figure 1: Four-finger model made by Professor T. Yamada ......................... 7
Figure 2: Two-finger model used in the project.......................................... 8
Figure 3: Two finger model with the user’s glove ........................................ 9
Figure 4: Closer look to the structure of one finger. ..................................... 9
Figure 5: Represents the different joints of finger1 (blue) and finger2 (green) ... 10
Figure 6: Sketch of one of the fingers in the XYZ plane. .............................. 11
Figure 7: Sketch of one finger in the Y’Z plane........................................... 12
Figure 8: Picture of the controller RSC-U485............................................. 12
Figure 9: Power supply for our model......................................................... 13
Figure 10: Sketch of the process. ................................................................. 13
Figure 11: Change of coordinates ............................................................... 15
Figure 12: tryWallOne has activated two walls ......................................... 17
Figure 13: tryWallTwo has activated the four walls around the finger.......... 17
Figure 15: How to determine the exit of the sphere..................................... 20
Figure 16: How to exit the cube ................................................................. 22
Figure 17: Transmitted feeling ................................................................. 23
Figure 18: Parameters of the fingers while using transFeel ......................... 26
Figure 19: The security measure is not respected ....................................... 27
Figure 20: The security measure is respected .............................................. 28
Figure 21: The fingertips do not enter the sphere ...................................... 28
Figure 22: Results due to activating the sphere......................................... 29
Table’s index

Table 1: How the parameters are shown in the result figures .......................... 25
PART 1: PROYECT MEMORY
CHAPTER 1: INTRODUCTION

1.1 STUDY OF EXISTING TECHNOLOGIES RELATED TO THE PROJECT

Haptics is the study that focuses on the interaction between the computer and the user through tactile feedback [1]. The haptic interface is what allows this interaction. It has different responses depending on the situation and the programming it has. The haptic interface can be a stick, a pen, a joystick or a grasping device for example. This element allows the communication to be from the computer to the user and the other way as well. [2] The user can interact with virtual objects through the haptic interface. It is programmed in order to send a signal to the user depending on the information that the computer is given.

[1] With this technology, a user could hold a virtual tennis ball and get the feeling as if it was an actual object, since the computer sends information through the haptic interface. The computer will mimic the feeling this object would have if the user was holding an actual tennis ball.

The sector that focuses on the investigation of haptics is very wide and has many different and useful applications. It is also a sector that develops continuously and improves rapidly. It aims towards an interface that will be able to transmit the feeling of touch to the user with great accuracy.

To show this improvement, one can just pay attention to the experiment of the students from Salisbury that will be described below [3]. In this case, they used a
virtual stick as the element in contact with the user. However, this has improved and right now the most popular models are those that allow the user to grip like opposable thumbs. The Haptic Gripper interface [4] has been a breakthrough in this sector, since grasping seems like a more natural movement and way of interacting for us humans.

The different applications of this science are mostly focused on two sectors: games and education. In the article from the University of Stanford [3] we find examples of both. An example of the games sector is a virtual ping-pong game called “Haptic Battle Pong” that was created by a couple of students in Salisbury’s experimental haptics. Another example [2] could be the games in which the aim is to win a race. In this type of games, the haptic interface will have a different behavior depending on if the user hits a wall, another car or if the car is driving through a road that has many bumps.

From this article [3] we can also see the importance this element is having in education, mostly applied to medicine. The students will be able to practice incisions since the program tries to reproduce the human body and organs. This is a very hard task since the human parts react differently depending for example if it is a healthy organ or not. In addition to this, this element can bleed after the user makes an incision and it will stop if the user sews it correctly. Summarizing, this part has lots of variables and different aspects that must be taken into account when designing the haptic interface. Its different reactions have to be taken into account as well, which is what makes it hard to make an accurate design.

Another example related to medicine has been developed by the University of Uppsala, Sweden [5]. The user has the ability to study the organ of its patient and this will help the doctors diagnose diseases like cancer for example. This is currently being used to study and detect lung cancer at its early stages.
A different example regarding education is given by the article “Virtual frog dissection for anatomical learning” [6]. In this case the students could not dissect the frog anymore due to the animal protection policy and the price of the resources needed which became more expensive. Due to this, a dissecting practice was simulated. This helped the students have a better understanding of the appearance of the exercise. Meanwhile, the students that use traditional means knew better the aspects related to function.

1.2 MOTIVATION FOR THE PROJECT

Due to what has been explained in the previous section it is clear that the interaction between the user and the robot, in this case a computer, has gained importance in the past years. In addition to this, the number of robots used in different applications has increased and will continue to do so.

The ability to feel whether a robot hits an obstacle or what kind of consistence a surface has can be valuable in many areas. Some examples could be: hospitals during surgeries or increasing the enjoyment of a game, but as discussed earlier this field has more applications.

One can also see that from this interaction many processes can be improved and resources will be saved. Just by looking at the education sector this seems clear. If medicine students can access this technology, they could have more realistic labs before they have to do an actual cut on a living patient. This will make the learning process faster since they could practice many times the same procedure. Also, mistakes could be made to see the different reactions to a bad action which will also complement the education since they could see the repercussions.
In line with this, money will be saved since they will not need as many corpses or plastic models. This is always a good motive from the business point of view.

Another possible sector would be design. One could feel the actual object while it is only a prototype and the buyer could do this as well. This implies saving money and time as well. On the one hand, the designer could modify the element as many times as he or she wants before building the prototype. On the other hand, the buyer could have the actual object and try it before it is actually built so any changes could be made faster without having to rebuild the prototype for example. As said earlier time and money are saved again.

This report describes how different virtual shapes can be felt through a two-finger model haptic interface which can also reflect the forces a remotely controlled finger experiences.

1.3 Objectives

The main objective of this Project is to be able to create artificial or virtual objects and be able to detect them with the help of two robot fingers. This means that the user will experience a reaction and actually notice when the fingertips of our model enter the space determined as object. If this happens, the torque will be turned on in the joints needed in order to exit the object.

This is complemented by an extra feature which will also give an idea of the material that the object is made of. The user could be able to tell if the object is hard or if it is medium-soft. When the object is of the second kind, the user will experience an increasing force while he or she tries to get to the centre of the
object. This will set the difference between an object that you can squeeze for example a pillow or an object that you cannot squeeze like a hard wood desk.

1.4 **EXPLAINED SOLUTION**

The Project presented here is based on a four-finger model made by Professor Takayoshi Yamada who currently works at the University of Lund (Sweden). His model can be seen in Figure 1.

The project was made during the course FRT090 Project in Automatic Control in the University of Lund. It was done with Zsolt-Patrick Demeter and John Kroon and had two supervisors Professor Takayoshi Yamada and Anders Robertsson.

![Four-finger model made by Professor T. Yamada](image)

*Figure 1: Four-finger model made by Professor T. Yamada*

Due to Professor Yamada’s extensive programming, the model that he made has two different modes. In the first one the user can move the fingers freely and they will stay in the last position before the user stops putting any force. The other consists of telling the finger a position where they should go. It worked with a master-slave relationship.
The Project that will be described here is based on a two-finger model that is shown in Figure 2. This model was built by the Automatic Control department at Lunds Tekniska Högskola (LTH) in Sweden.

![Figure 2: Two-finger model used in the project.](image)

As stated before, the goal of this project is to simulate a virtual object which can be felt with the help of our model. This means that the user will experience a reaction and actually notice when the fingertips enter the space determined as object. If this happens, the torque will be turned on in the joints needed in order to exit the object. The maximum torque applied will vary depending on the consistency of the artificial object designed due to programming specifications to make the process more realistic.

This will be done with the use of Forward Kinematics and Inverse Kinematics. The Forward Kinematics will allow the user to determine the position of the fingertip since the angles between the links will be known. The Inverse Kinematics will calculate the angles between the links that are needed in order to access a position with the fingertip.
The fingertips of our model, which was shown in Figure 2, have magnets attached. These magnets allow the user to control the haptic interface better with a glove that has magnets attached to two fingers, which can be seen in Figure 3.

Each finger is articulated and has three different parts, three independent servomotors RCS301CR, which will be called links from this moment on. Each link is connected to the next one through a joint, which is the part that is called knuckle in a human finger. A closer look to the fingers can be made in Figure 4.
Chapter 1: Introduction

The way in which we will refer to the different joints from this point on, can be seen in Figure 5. The distances from J1 to J2 and J2 to J3 are the same 47.4 mm, whereas the distance from J3 to the tip is 23 mm. Notice that both fingers are the same which means that J4 is the same as J1, J5 as J2 and J6 as J3.

![Figure 5: Represents the different joints of finger1 (blue) and finger2 (green)](image)

The joints are really revolute joints and, because of this reason, they have only one degree of freedom. The element that we can vary according to the Denavit-Hartenberg Convention is the joint angle, which will be shown as α from this point on. This angle is the one between a link and the following link. Since a revolute joint is represented by “R”, each of our fingers is considered a “RRR” robot. This means that the other Denavit-Hartenberg parameters: link length, link twist and link offset are constant[8].

In our case, the finger joint J1 allows the finger to move sideways with α1 while the remaining joints, J2 and J3, allow the finger to bend with the angles α2 and α3 respectively.

As stated earlier, both fingers have the same configuration which can be seen in Figure 6. As one can see, the coordinates (x,y,z) are the ones that correspond to the fingertip.
These coordinates are the ones that we are going to work with in order to determine if the fingertip is inside the object or not, since they will be the ones given to us through Forward Kinematics.

One can also see $\alpha_1$ in Figure 6, which is the angle from the vertical plane to the finger after moving around the lower joint $J_1$. So, if the finger is straight up $\alpha_1$ would be zero degrees and if it is lying on the surface then it would be 90 degrees.

![Figure 6: Sketch of one of the fingers in the XYZ plane.](image)

This angle $\alpha_1$ has a different direction from $\alpha_2$ and $\alpha_3$, which take into account how much the finger is bending. It is important to explain that $\alpha_3$ is a relative angle since they all are joint angles, which measure the angle between one link and the next one as explained before.
This means that $\alpha_3$ measures the angle between the second and third link of the finger, so the absolute angle of the third link is $(\alpha_2+\alpha_3)$. All this is shown in Figure 7.

\[\begin{align*}
\text{Figure 7: Sketch of one finger in the } Y'Z\text{ plane}
\end{align*}\]

All the programming is made on the computer which is then connected to RSC-U485 which can be seen in Figure 8, through USB. The communication has a byte rate of 115200 through RS485. The RSC-U485 communicates with the servo motors RCS301CR through half duplex communication. This means that they cannot transfer information at the same time.

\[\begin{align*}
\text{Figure 8: Picture of the controller RSC-U485}
\end{align*}\]
The RSC-U485 can set the reference angle where the servo should go as well as the time lapsed for the movement and the maximum value of the torque for example. The same information can also be read from the servos and seen in the computer.

However, even though the RSC-U485 can convert the control signals, it does not control the power supply for the servo motors and therefore cannot control the current given directly. The power is supplied by the element shown in Figure 9.

![Image](image.png)

*Figure 9: Power supply for our model.*

The internal process that relates the computer with the actual finger can be seen in Figure 10. The arrows that link one part to the next have a different color to reflect that the communication is half duplex which, as said before, means that the communication between them can be done in both directions but not simultaneously. The power supply will be between the RSC-U485 and the 3 servomotors.

![Diagram](diagram.png)

*Figure 10: Sketch of the process.*
To fulfill the objective of this project, Forward Kinematics and Inverse Kinematics have been used. At first the programming was done for one finger and then changed to make it more generic in order to be able to work with both fingers which imply six different joints. The Velocity Kinematics could have also been used in order to implement the gravity compensation model and detect the forces applied to the finger.

Using the Forward Kinematics we are able to know the position of the fingertip because we know the angles between links ($\alpha_1$, $\alpha_2$ and $\alpha_3$). Since each joint only has one degree of freedom in a tangential direction and the length of the links are known, we only need the angles to determine the final position. The equations for this are obtained from Figure 6 and Figure 7, and are the following:

\[
\begin{align*}
  z &= L_2 \cdot \sin(\alpha_2) + L_3 \cdot \sin(\alpha_2 + \alpha_3) \\
  x &= y' \cdot \sin(\alpha_1) \\
  y &= y' \cdot \cos(\alpha_1) \\
  y' &= L_3 + L_2 \cdot \cos(\alpha_2) + L_3 \cdot \cos(\alpha_2 + \alpha_3)
\end{align*}
\]

In the programming this was done through the function “int calculateEndPos(float alpha[], float cords[]), which can be found in Code.

On the other side, by using the Inverse Kinematics the user chooses the position where the fingertip is supposed to be. Then the angles $\alpha_1$, $\alpha_2$ and $\alpha_3$ for this position are determined. The torque will be applied until these angles are obtained between the links. The equations are obtained as well from Figure 6 and Figure 7 and they are the following.

\[
\alpha_1 = \tan^{-1}(x/y)
\]
\[ y' = \sqrt{x^2 + y^2} \]

\[ d = \sqrt{((y' - L_1)^2 + z^2)} \]

Knowing that

\[ \theta_i = \cos^{-1}(z / d) \text{ for } (d \neq 0) \land (-1 \leq z / d \leq +1) \]

\[ \theta_2 = \cos^{-1}(L_0^2 - L_2^2 - d^2 + 2 * L_2 * d) \]

the remaining angles can be calculated as:

\[ \alpha_2 = \pi / 2 - \theta_1 - \theta_2 \]

\[ \alpha_3 = \pi - \cos^{-1}(d^2 - L_2^2 - L_3^2 + 2 * L_3 * d) \]

In the programming this was done with the function “void calculateAngles(float coord[], short angs[], short raw[])”, which can be found in Code.

Since each of the fingers had its own coordinate system and to simplify the calculations needed when calculating if the fingertip was inside the object, a global coordinate system was created. This can be seen in Figure 11.

![Figure 11: Change of coordinates](image)
The blue axis represents the coordinates for finger one and the green axis corresponds to the second finger. The global axis is the one for finger one, so the changes seen in the box in the middle of Figure 11 are the ones needed in order to have the same coordinates in both fingers. This will have to be taken into account both in the Forward and Inverse Kinematics, as well as in other programs.

As stated earlier, we want to create the feeling of different materials. This will be done by regulating the maximum torque that will be given in each joint. A hard object for example, will have a constant value equal to the maximum torque that the servos can handle. On the other hand, if the object is soft, the torques will increase as the user gets closer to the centre of the object.

We can also set a different torque depending on the joint the torque is going to be applied to. This could help create more realistic models in many implementations like medicine for example. If we thought of a human organ, the force needed to expand it will most likely vary depending on the direction of the applied force.

Before creating more complex programs regarding artificial objects, three different programs called “void tryWallOne()”, “void tryWallTwo()” and “void tryWallThree()” were created.

With the first one, a different value for the torque in each joint was given by the elements of the vector TRQ. If the fingertip of finger 1 exited a defined region, which in this case is [-40, +40], the torques in each of the six joints will be turned on and the fingers will move to a certain position defined by the joints’ angles through the vector moveToW1. Otherwise, the torques in every joint will be turned off and the user could move both fingers freely through the space.
The region represents two large walls that are parallel to the z axis and which can be seen in Figure 12. It is important to clarify that the angles of moveToW1 appear to have a very big value in the Code. The value that is shown in this vector is the angle in tenths of degrees. This means that if moveToW1[0] = 100 for example, the first angle will have a value of 10 degrees.

Regarding tryWallTwo, we have created a region of four walls around finger 1. This means that the finger is confined in a room, which is shown in Figure 13. In this figure, the green shape represents the part where the finger can move freely. So in this region, the torques are turned off.
If the fingertip of finger1 enters any of the walls, which is the same as exiting the green zone, the torques will turn on. The program saves the angles just when the fingertip exits the green zone. The torques will be applied to move the finger to the position that we saved the angles. This means that the user will always be moved to the last permitted position that our fingertip was in.

Finally, in tryWallThree, the limitations that are defined in this case are not according to the position of the fingertip but to the angles in between the links. If a certain angle is reached, then the torques will be turned on to move to a new angle. This program has only been done for the joints in finger one and the maximum torque in each joint is also defined by the value TRQ.

As stated earlier the main objective of the project is to create artificial objects and to be able to feel them. The idea was to attach one’s fingers to the robot fingers due to the magnets described earlier. After trying with simple elements like the walls around the finger which were explained above, the next step was to create an artificial object in the room.

The way to accomplish this is to define an arbitrary shape. As long as the fingertips do not enter the area defined as object, the torques will be turned off so the user can move freely.

However, if the fingertip enters the object, the program will restrict the motion of the fingers. The torques will be turned on in order to exit the object through the shortest path to minimize the time and energy needed. Depending on the maximum value established the feeling is created.
The process has three steps that will be shown in the objects that we created. These are:

- Define the object.
- Detect if the fingertip is inside the area defined as object.
- Decide how the program should react to the information: turn on the torque to exit the object according to the programming or do not do anything.

First a sphere was made. The first step is to define the sphere by setting the parameters for its origin (bx,by,bz) and radius (br). Once this is done, we should determine if the finger is inside the object or not since this will determine if the torque should be on or off respectively. The positions of the fingertips are obtained by using Forward Kinematics since we know the angles between the links.

To detect if the user is in the sphere we compare the radius of the sphere with the distance from the fingertip to the centre of the sphere. If this difference is negative it means that the finger is inside the object and otherwise it means that it is not.

Assuming that the finger is inside the object, the torques will turn on until the fingertip is outside the object. However there are many paths that the fingertip can follow in order to exit the object as one can imagine.

It was decided that the shortest path should be followed since it will require less time and therefore less energy. Due to this, the fingertip will move in a normal direction to the object surface.
How to decide the point to exit the sphere is sketched in Figure 14. In this figure O represents the center of the sphere, P the point where the fingertip is inside the object and Q is the closest exit point.

To exit the sphere, the vector between the centre of the sphere and the fingertip represented as OP is normalized. After this, we can calculate the exit point for the fingertip represented by Q. Q is calculated as the normalized vector multiplied by the sphere’s radius and taking into account the centre of the sphere in order to get the correct coordinates.

So the torques of the different joints will be turned on until the fingertip is at point Q. Since we have the point, Inverse Kinematics is used to calculate the angles between the joints. As we can see, both mathematical models are used.

![Figure 14: How to determine the exit of the sphere.](image)
In the programming done, we assumed it was a soft sphere. This means, as stated earlier, that the torque will increase as we get closer to the centre. To do this, the maximum torque applied in each joint will be equal to a parameter that is called “stiff”. This parameter is directly proportional to the position of the fingertip and inversely proportional to the radius of the sphere.

Another figure was created, in this case a cube. The limits of this element were determined by planes that are parallel to the coordinate system that has been implemented. The dimensions of the cube will be defined by the vector $c$. This implies that to determine if the fingertip is inside the cube, its coordinates have to be in the region determined by all the defined planes.

To check if the fingertip is inside the cube, a comparison between the parameters that define the cube and the position of the fingertip, which is determined by the Forward Kinematics, is made. Like with the sphere, once we determine that it is inside the torque will be turned on to exit the object.

The path that will be followed in order to exit the cube is easier than with the sphere since the planes are parallel to the coordinate system, as stated earlier. The distance between the fingertip and each of the walls is stored in the vector defined as “$d[6]$”. By obtaining the minimum distance of one of the finger’s coordinates to the walls, the shortest path is found.

Once this coordinate is found, it will be switched for the wall’s coordinate and the finger will be told to go to the new position. Notice that in this case, only one of the coordinates of the fingertip will be modified since the cube is defined in planes that are perpendicular to our defined axis.
We can see this in Figure 15. The fingertip is inside the cube in the point (1), so the torque will be turned on. Since the plane that is closer to the viewer is the one that is closer to the point (1) as well, the fingertip will move to this plane. So the fingertip will move from point (1) to point (2) through a straight line that is normal to the plane.

![Figure 15: How to exit the cube](image)

In this case, our cube is defined as a hard object. This means that the maximum torque will be applied when the finger enters the cube.

In addition to this, another program called transFeel was created. This program has been thought to be very useful. The user could for example manipulate a smaller model and a bigger model could mimic the user’s movements in order to do a task. An example of this could be if the user wants to lift a very heavy object in a certain manner, he will do it in a smaller scale and the bigger machine will mimic the user and do the actual task. Since this program could be applied for real life situations, a security measure was implemented.

If the security constraint is respected the user will be able to move finger1 since its torques will be turned off, and finger2 will follow or imitate this movement. To do this, the torques that control the joints in finger 2 will be turned on in order to have the same angles $\alpha_1$, $\alpha_2$ and $\alpha_3$ in finger 2 than the ones that finger 1 has.
However, if the constraint is not respected, the torque will be activated in both fingers in order to keep the last position that respected the security constraint. This will continue until the security constraint is met.

The security measure depends on the maximum current that each of the joints of finger 2 experience. This will let the user know that there is something wrong, like an object in the way for example, so that no harm will be caused. As said before, the user could be in contact with finger 1 far away from the copy-cat finger (finger 2), so the user could in this way be aware of the environment around the second finger to prevent causing any damages. An image showing this program can be seen in Figure 16.

Figure 16: Transmitted feeling
1.5 **RESOURCES/TOOLS USED**

All the programming was done using Eclipse with C language. The fundamental protocol that was used to develop this project was made by the FUTABA Corporation [7].

The protocol used had a few modifications made by Professor T. Yamada and one change was made in one of the functions in order to be able to control the maximum torque as explained earlier.

To see how the different parameters of the programs change due to the movement of the finger or fingers, the aterm from Linux will be used.
Chapter 2: Results and experiments

A test was done to get result with the “void transFeel()”. It is the program in which finger2 follows finger1 except if a safety constraint is not satisfied. How the parameters will be shown in the next figures as shown in Table 1. All the parameters in the following experiments will be shown in the same way.

<table>
<thead>
<tr>
<th></th>
<th>Finger 1</th>
<th>Finger 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>Current</td>
<td>J1</td>
<td>J2</td>
</tr>
</tbody>
</table>

Table 1: How the parameters are shown in the result figures

As you can see in Figure 17, the parameters that show the coordinates of both fingertips is the same with the exception of the “z” coordinate due to the difference of 130 mm that was explained in Figure 11.

By looking at the parameters one can see that the coordinates of the second finger are correct but there is an error. This is due to a delay. The second finger moves to have the same coordinates as the first one. However, the user can notice the time difference due to a big delay.

Also, the torque applied in the first finger in zero except in the fingertip due to the user applying a force. On the contrary, the joints of the second finger experience a torque since they are requested to follow the first finger.
<table>
<thead>
<tr>
<th>Activities</th>
<th>Term</th>
<th>ons 14:54</th>
<th>aterm</th>
</tr>
</thead>
</table>

Compart: dev/ttyUSB0, baudrate=115200

The compartment is opened.

| SID-1: 12.7 deg | 0.06 deg/s | 0 | [nA] | 33 | 0 | 6.2 [V] |
| SID-2: 12.3 deg | 0.06 deg/s | 0 | [nA] | 34 | 0 | 6.6 [V] |
| SID-3: 0.5 deg | 0.06 deg/s | 12 | [nA] | 14 | 0 | 6.2 [V] |
| SID-4: 10.0 deg | 0.12 deg/s | 12 | [nA] | 15 | 0 | 6.2 [V] |
| SID-5: 0.9 deg | 0.06 deg/s | 21 | [nA] | 31 | 0 | 6.2 [V] |

**Position:**
- 0: 0 [nA] | 14 [nA] | 18 [nA] | 0 [nA] | 2 [nA] |

**Figure 17:** Parameters of the fingers while using transFeel.

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
GRADO DE INGENIERÍA ELECTROMECÁNICA

Chapter 2: Results and experiments
In Figure 18 we see that in the first rows, the situation is the same as that shown in Figure 17: finger two follows finger one. However, the security condition is not respected, so the whole process is stopped. As explained before, once the security constraint is not respected, both fingers will stay in the last position until the security condition is reestablished.

This can be seen in the second half of Figure 18 since the torque in each of the joints of finger one is turned. One can also see that the position before the security breach is kept in the following moments.

<table>
<thead>
<tr>
<th>Current</th>
<th>0 μA</th>
<th>0 μA</th>
<th>14 μA</th>
<th>18 μA</th>
<th>0 μA</th>
<th>0 μA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>-1.000209m</td>
<td>0 μA</td>
<td>15.324739m</td>
<td>-0.956218m</td>
<td>113.743158m</td>
<td>-126.203675m</td>
</tr>
<tr>
<td>Current</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>14 μA</td>
<td>15 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-2.625656m</td>
<td>0 μA</td>
<td>115.568479m</td>
<td>-1.240063m</td>
<td>114.263330m</td>
<td>-112.015695m</td>
</tr>
<tr>
<td>Current</td>
<td>0 μA</td>
<td>0 μA</td>
<td>14 μA</td>
<td>19 μA</td>
<td>0 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-4.650767m</td>
<td>0 μA</td>
<td>115.666531m</td>
<td>-1.026868m</td>
<td>114.398423m</td>
<td>-112.005893m</td>
</tr>
<tr>
<td>Current</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>14 μA</td>
<td>14 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-9.643473m</td>
<td>0 μA</td>
<td>116.387545m</td>
<td>-0.606767m</td>
<td>115.119023m</td>
<td>-113.635504m</td>
</tr>
<tr>
<td>Current</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>14 μA</td>
<td>14 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-19.650453m</td>
<td>0 μA</td>
<td>116.197469m</td>
<td>-0.834249m</td>
<td>115.578592m</td>
<td>-118.512472m</td>
</tr>
<tr>
<td>Current</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>14 μA</td>
<td>14 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-18.473223m</td>
<td>0 μA</td>
<td>115.369990m</td>
<td>-4.714767m</td>
<td>116.276253m</td>
<td>-118.857538m</td>
</tr>
<tr>
<td>Current</td>
<td>0 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-20.241325m</td>
<td>0 μA</td>
<td>114.787188m</td>
<td>-7.769309m</td>
<td>116.337140m</td>
<td>-122.118562m</td>
</tr>
<tr>
<td>Current</td>
<td>0 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-9.306608m</td>
<td>0 μA</td>
<td>114.716100m</td>
<td>-5.006322m</td>
<td>115.845484m</td>
<td>-124.433941m</td>
</tr>
<tr>
<td>Current</td>
<td>74 μA</td>
<td>187 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-16.653427m</td>
<td>0 μA</td>
<td>115.185933m</td>
<td>-4.350441m</td>
<td>115.535130m</td>
<td>-124.466110m</td>
</tr>
<tr>
<td>Current</td>
<td>21 μA</td>
<td>17 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-17.285911m</td>
<td>0 μA</td>
<td>115.623124m</td>
<td>-2.833625m</td>
<td>115.968046m</td>
<td>-124.234332m</td>
</tr>
<tr>
<td>Current</td>
<td>70 μA</td>
<td>102 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-17.492644m</td>
<td>0 μA</td>
<td>115.642904m</td>
<td>-2.989225m</td>
<td>115.903290m</td>
<td>-124.343312m</td>
</tr>
<tr>
<td>Current</td>
<td>79 μA</td>
<td>103 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-17.223753m</td>
<td>0 μA</td>
<td>115.675092m</td>
<td>-2.385837m</td>
<td>115.949840m</td>
<td>-124.343353m</td>
</tr>
<tr>
<td>Current</td>
<td>85 μA</td>
<td>122 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-17.281490m</td>
<td>0 μA</td>
<td>115.677683m</td>
<td>-2.277252m</td>
<td>115.941650m</td>
<td>-124.343362m</td>
</tr>
<tr>
<td>Current</td>
<td>75 μA</td>
<td>154 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-17.288159m</td>
<td>0 μA</td>
<td>115.677683m</td>
<td>-2.277252m</td>
<td>115.941650m</td>
<td>-124.343362m</td>
</tr>
<tr>
<td>Current</td>
<td>77 μA</td>
<td>155 μA</td>
<td>15 μA</td>
<td>0 μA</td>
<td>15 μA</td>
<td>0 μA</td>
</tr>
<tr>
<td>Position</td>
<td>-17.298014m</td>
<td>0 μA</td>
<td>115.686350m</td>
<td>-2.294549m</td>
<td>115.954747m</td>
<td>-124.343354m</td>
</tr>
</tbody>
</table>

**Figure 18:** The security measure is not respected.

In Figure 19 we see the opposite case, the security condition is reestablished. In the first part, the torque is active in all the joints and the position does not change. The opposite happens in the second part, where the security condition is respected, so the torques for the first finger are turned off and the second finger follows the position of the first one again.
The following results correspond to a sphere that is created between both fingers. As said before, the finger will be able to move freely as long as it does not enter the defined sphere. This situation can be seen in Figure 20, where the torques which are shown are due to the user’s force on the finger while he or she moves it.

Figure 19: The security measure is respected.

Figure 20: The fingertips do not enter the sphere.
The opposite situation can be seen in Figure 21. In this case we can find that the torques will be activated for finger two from the red mark on since the fingertip of finger two has entered the sphere. As one can also see, this does not imply that the torques of finger one are activated since their torques are independent. The torques of the first finger will be activated after its fingertip enters the sphere which occurs where the blue line is placed in Figure 21.

<table>
<thead>
<tr>
<th>Current: 0 mN</th>
<th>0 mN</th>
<th>16 mN</th>
<th>14 mN</th>
<th>0 mN</th>
<th>2 mN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-11.494472m</td>
<td>114.264434m</td>
<td>-18.56856m</td>
<td>-24.412538m</td>
<td>106.638022m</td>
<td>-101.123283m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
<tr>
<td>Current: 0 mN</td>
<td>0 mN</td>
<td>16 mN</td>
<td>14 mN</td>
<td>0 mN</td>
<td>2 mN</td>
</tr>
<tr>
<td>Positions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21.853669m</td>
<td>114.264434m</td>
<td>-24.328796m</td>
<td>-24.229690m</td>
<td>106.638022m</td>
<td>-104.495172m</td>
</tr>
</tbody>
</table>

Figure 21: Results due to activating the sphere.
CHAPTER 3: CONCLUSIONS

After studying the haptic science thoroughly, it is clear that many different applications can be found in a wide variety of sectors and that this technology is continuously improving to increase its uses and broaden the possibilities it offers.

Considering what methods and control were available to use when regulating our fingers, the possibilities were limited a great deal. Many times it would be preferable to control the supplied current and direction for the different joints.

Different programs have been coded. The ones that create a virtual shape in a region are the ones that cover the objectives that were set for this program. However, due to the wide applications for the haptic interfaces, another useful programs like “void tryTransFeel()” was programmed. It could be used for example if a big machine is expected to behave in a certain way. The user could always use a smaller model to manipulate and the big machine will mimic the movements. As explained earlier this program has also a security measure which will be needed in this case as well.

However, some problems were experienced through the development of the project. The calculations of the angles and positions have some limitations due to the usage of sine and cosine operations. Because, as it is known, \( \sin(x) = \sin(\pi - x) \) and \( \cos(x) = \cos(-x) \).

These limitations make the finger behave oddly when passing the 90 degrees limit and switching between positive and negative coordinate values in some situations.
It is mainly a problem for the lowest link since some improvements have been done with elbow up and elbow down modes for the two upper links.

For the different shapes and applications made, the fingers behave pretty well. There are some issues with twitchiness if a too small time is set for the movement. It makes the finger move very quickly and give an excessive torque when exiting the object.

Because it takes some time to update the position and calculate if a torque should be applied or not, this can allow the finger to move back into the object. When it’s detected that the finger is inside the object it pushes it out very quickly and the twitchy movement starts.

When trying to implement the transmitted feeling it was noticed that the “void tryTransFeel()” method suffers from a big delay, which has made it hard to make a good implementation. The controlling finger will be notified about an object too late and will therefore already have entered the object when the regulation starts. This problem does not occur when the finger is moved slowly and only reveals itself when exposed to fast movements.
Chapter 4: Future improvements

There are a lot of improvements that could be done, mostly to make the process more generic, that were not implemented due to lack of time. Some examples will be explained below.

First of all, to make the program more generic it would be interesting to design a function that could get any shape given by the computer. Then decide if the torque is on or off depending on if the finger is inside it or not as it has been done with the sphere and the cube.

In addition to this, a simulation on how the object is behaving would be very interesting. You could see the cube or sphere and how your fingers are trying to enter the object. This visual tool would be a great improvement of the user’s experience.

In order to solve the problem with the sine and the cosine explained above, it would be interesting to take into account the direction in which the finger was moving so that it would say 120 for example instead of 60 which is what the program thinks the angle is now.

This will also mean improving the calculations for the angles and position when reaching some critical coordinate and angular values.

Right now the finger can be freely moved by the user if the fingertip is not inside the defined object. However the effect of the gravity will make it fall. It would be
very interesting to create a gravity compensation model in order for the finger to stay still in any position, after the user applies the extra force.

In addition, the twitching problem should be fixed as well as the delay to make the movement smoother and increase the precision of the processes. Both following with “transFeel” and exiting an object could benefit from this.

Finally if the applied torque depended on the current output in the transmitted feeling method, the consistency of the object could be felt which cannot be done currently.
BIBLIOGRAPHY

PART II: CODE
CODE

/*
 * wall_s23.c
 *
 * Created on: 25/04/2014
 * Author: mat10jk1
 */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#ifdef _WIN32
#include <Windows.h>
#else
#include <unistd.h>
#endif
#include "rs_servo/rs_servo.h"
#include "rs232/rs232.h"
#include "kbhit/kbhit.h"

// For multiple motors
#define NF 2  // The number of fingers
#define NJ 3  // The number of joints for each finger
#define NUM_MOTOR NF*NJ  // The number of servo motors
#define L1 47  //Link length 1
#define L2 47  //Link length 2
#define L3 23  //Link length 3
#define dPi 4.0 * atan(1.0)

int loop;  //Variable for loops
int ret;
int hComm;  // Handle of the comport
float currCoords[6];  // vector for calculated position coordinates
short cAngle[6];  // Vector for the calculated angles
short rawAngle[NUM_MOTOR];  // Vector for the rawAngles
float angle[NUM_MOTOR];
short moveToW1[NUM_MOTOR];  // Calculated angles where to
short moveToW2[NUM_MOTOR];  // move the fingers
float dPi = 1.0;  // PI
short SERVO_TORQUE[NUM_MOTOR];  // Vector for reading which servo is on/off
int TRQ[NUM_MOTOR];  // Limit for torque in each servo
float newCoords[NUM_MOTOR];  // Saves the point the finger should move to
int nsid = NUM_MOTOR;  // The number of servo motors

unsigned char sid[NUM_MOTOR] = { 1, 2, 3, 4, 5, 6 }; // Array of servo IDs
short sPos[NUM_MOTOR]; // Array of servo positions
StructSrvInfo info[NUM_MOTOR]; // Array of servo status

//---Sphere Constants---/
float bx = 0;
float by = 70;
float bz = -65;  // defining where the sphere is located
float br = 70;  // and it's size
float shapeV = 0;
int stiffV = 0;

//----------------------/

//---Cube Constants---/
//(x,x,y,y,z,z) :)
float c[6] = { -40, 40, 50, 80, -90, -40 };

int minDist = 0;
int minWall = 0;
//----------------------/

//---Copy Variables---/
short copyAngle[6];
//----------------------/

void tryWallOne();
void tryWallTwo();
void tryWallThree();
void trySphere();
void tryCube();
void tryTransFeel();
/*
 * Method for calculating the end position when the angles are known.
 * alpha are the known angles
 * coords will save the fingertips coordinates
 */
int calculateEndPos(float alpha[], float coords[]) {
    coords[0] = \((L1 + L2 \times \cos(\alpha[1] \times \frac{\pi}{180}) + L3 \times \cos((\alpha[1] + \alpha[2]) \times \frac{\pi}{180})) \times \sin(\alpha[0] \times \frac{\pi}{180})\);
    coords[1] = \((L1 + L2 \times \cos(\alpha[1] \times \frac{\pi}{180}) + L3 \times \cos((\alpha[1] + \alpha[2]) \times \frac{\pi}{180})) \times \cos(\alpha[0] \times \frac{\pi}{180})\);
    coords[2] = \((L2 \times \sin(\alpha[1] \times \frac{\pi}{180}) + L3 \times \sin((\alpha[1] + \alpha[2]) \times \frac{\pi}{180}))\);
    coords[3] = \(-L1 - L2 \times \cos(\alpha[4] \times \frac{\pi}{180}) - L3 \times \cos((\alpha[4] + \alpha[5]) \times \frac{\pi}{180}) \times \sin(\alpha[3] \times \frac{\pi}{180})\);
    coords[4] = \((L1 + L2 \times \cos(\alpha[4] \times \frac{\pi}{180}) + L3 \times \cos((\alpha[4] + \alpha[5]) \times \frac{\pi}{180})) \times \cos(\alpha[3] \times \frac{\pi}{180})\);
    coords[5] = -130 - \((L2 \times \sin(\alpha[4] \times \frac{\pi}{180}) + L3 \times \sin((\alpha[4] + \alpha[5]) \times \frac{\pi}{180}))\);
}

/*@Method for calculating the inverse kinematics.
 * coords refer to fingertips coordinates
 * angs saves the calculated angles
 * raw the current angles from the motors to determine if elbow up or down
 */
void calculateAngles(float coords[], short angs[], short raw[]) {
    int elbow1 = 1;
    int elbow2 = 1;
    if (raw[2] < 0) {
        elbow1 = -1;
    }
    if (raw[5] < 0) {
        elbow2 = -1;
    }
}
```c
float d1 = sqrt(
    (sqrt(coords[0] * coords[0] + coords[1] * coords[1]) - L1)
    * (sqrt(coords[0] * coords[0] + coords[1] * coords[1]) - L1)
float gama1 = acos((L2 * L2 + L3 * L3 - d1 * d1) / (2 * L2 * L3));
angs[0] = (atan(coords[0] / coords[1])) * 180 / dPi * 10;
angs[1] = (dPi / 2 - acos((coords[2]) / d1)
    - elbow1 * acos((L2 * L2 + d1 * d1 - L3 * L3) / (2 * L2 * d1)))
    * 180 / dPi * 10;
angs[2] = elbow1 * (dPi - gama1) * 180 / dPi * 10;
float d2 = sqrt(
    + (-130 - coords[5]) * (-130 - coords[5]));
float gama2 = acos((L2 * L2 + L3 * L3 - d2 * d2) / (2 * L2 * L3));
angs[4] = (dPi / 2 - acos((-130 - coords[5]) / d2)
    - elbow2 * acos((L2 * L2 + d2 * d2 - L3 * L3) / (2 * L2 * d2)))
    * 180 / dPi * 10;
angs[5] = elbow2 * (dPi - gama2) * 180 / dPi * 10;
}
/*
* Main method decides which shape to create
*/
int main(void) {
    int cport_nr = 16;       // The converter is connected at /dev/ttyUSB0
    int bdrate = 115200;     // 115200 baudrate
    dPi = 4.0 * atan(1.0);
    hComm = RS232_OpenComport(cport_nr, bdrate);
    if (hComm == -1) {
        printf("Can not open comport\n");
        return (0);
    }
    printf("The comport is opened.\n");
    // Switch off torque on all motors
    ret = RsAllTorqueOnOff(hComm, sid, nsid, SERVO_TORQUE_OFF);
    if (ret < 0) {
        puts("RsAllTorqueOnOff() returned ERROR in RsAllOpenInit()");
    }
    return (0);
}
```

// CommClose( hComm );
RS232_CloseComport(cport_nr);
return 0;
}

// The current status of the servos is obtained.
RsAllGetInfo(hComm, sid, nsid, info);
// The status is displayed.
RsAllDispInfo(info, nsid);

while (1) {

// Get the positions of finger.
RsAllGetAngle(hComm, sid, nsid, rawAngle);

for (loop = 0; loop < NUM_MOTOR; loop++) {
    angle[loop] = rawAngle[loop] * 0.1;
}
// calculate the position of fingertip and angles
calculateEndPos(angle, currCoords);
calculateAngles(currCoords, cAngle, rawAngle);

//tryWallOne();
//tryWallTwo();
//tryWallThree();
//trySphere();
tryCube();
//tryTransFeel();

RsAllGetInfo(hComm, sid, nsid, info);

//---Outputs for different measurements---//

//printf("%d %d %d %d %d %d \n\r", rawAngle[0],
rawAngle[3], rawAngle[5], cAngle[0], cAngle[3], cAngle[5]);
printf("
Current: %d mA %d mA %d mA
\n\r",
info[0].mA, info[1].mA, info[2].mA, info[3].mA,
info[4].mA,
info[5].mA);
printf("Position: %fmm %fmm %fmm %fmm %fmm %fmm
\n\r", currCoords[0], currCoords[1], currCoords[2],currCoords[3], currCoords[4], currCoords[5]);
//printf("%d %f %d %d \n\r", stiffV, shapeV, TRQ[0], TRQ[3]);
// printf("%f %f %f \n\n", currCoords[0], currCoords[1], currCoords[2]);

    // If any key is pushed, this program is terminated.
    if (kbhit()) {
        getch();
        break;
    }
}
RsAllTorqueOnOff(hComm, sid, nsid, SERVO_TORQUE_OFF);

return 0;

} /*
 * Method for calculating if we are inside the shape with the fingertip.
 * cords refer to the coords of the tip
 * nrf refer to the finger 0 for first finger or 1 for second one
 */
float getSphere(int nrf) {
    shapeV = (currCoords[3 * nrf + 0] - bx) * (currCoords[3 * nrf + 0] - bx)
            - br * br;
    return shapeV;
}

float getCube(int nrf) {
    if (currCoords[3 * nrf + 0] > c[0] && currCoords[3 * nrf + 0] < c[1]
        int d[6];
        d[0] = currCoords[3 * nrf + 0] - c[0];
        d[3] = -currCoords[3 * nrf + 0] + c[1];
    }
minWall = 0;
for (loop = 1; loop < 6; loop++) {
    if (d[minWall] > d[loop]) {
        minWall = loop;
    }
}
minDist = c[minWall];
return 1;
}
return -1;

/*
* Calculates where the closest exit points in the sphere is
* coords is fingertips coordinates
* nrf is the finger to be calculated 0 for first or 1 for second
*/
void newPoint(float coords[], int nrf) {
    float vx = coords[3 * nrf + 0] - bx;
    float vy = coords[3 * nrf + 1] - by;
    float vz = coords[3 * nrf + 2] - bz;
    float modulo = sqrt(vx * vx + vy * vy + vz * vz);
    newCoords[3 * nrf + 0] = vx / modulo * br + bx;
    newCoords[3 * nrf + 1] = vy / modulo * br + by;
    newCoords[3 * nrf + 2] = vz / modulo * br + bz;
}

/*
* Method for creating an artificial sphere
*/
void trySphere() {
    //checks if first fingertip for first finger is inside sphere
    if (getSphere(0) < 0) {
        //calculates where to exit sphere
        newPoint(currCoords, 0);
        //calculates torque to be applied
        stiffV = ((shapeV * shapeV) / (br * br * br * br)) * 0x64;
        for (loop = 0; loop < 3; loop++) {
            TRQ[loop] = stiffV;
        }
    }
// turns torque on for all motors, finger 1
for (loop = 0; loop < 3; loop++) {
    SERVO_TORQUE[loop] = SERVO_TORQUE_ON;
}

} else {

    //Finger outside the sphere turn all torque off
    for (loop = 0; loop < 3; loop++) {
        SERVO_TORQUE[loop] = SERVO_TORQUE_OFF;
    }

}  // checks if first fingertip for second finger is inside sphere
if (getSphere(1) < 0) {

    // calculates where to exit sphere
    newPoint(currCoords, 1);

    // calculates torque to be applied
    stiffV = ((shapeV * shapeV) / (br * br * br * br)) * 0x64;
    for (loop = 3; loop < 6; loop++) {
        TRQ[loop] = stiffV;
    }

    // turns torque on for all motors, finger 2
    for (loop = 3; loop < 6; loop++) {
        SERVO_TORQUE[loop] = SERVO_TORQUE_ON;
    }

} else {

    // Finger outside the sphere turn all torque off
    for (loop = 3; loop < 6; loop++) {
        SERVO_TORQUE[loop] = SERVO_TORQUE_OFF;
    }

}  // Calculate the angles for the finger joints and tell them to move there
calculateAngles(newCoords, cAngle, rawAngle);
RsAllMoveWithTorque(hComm, sid, nsid, cAngle, 50, SERVO_TORQUE, TRQ);
/ * Method for a cube */
void tryCube() {
    if (getCube(0) >= 0) {
        currCoords[minWall % 3] = c[minWall];
        for (loop = 0; loop < 3; loop++) {
            SERVO_TORQUE[loop] = SERVO_TORQUE_ON;
        }
    } else {
        for (loop = 0; loop < 3; loop++) {
            SERVO_TORQUE[loop] = SERVO_TORQUE_OFF;
        }
    }
    if (getCube(1) >= 0) {
        currCoords[minWall % 3 + 3] = c[minWall];
        for (loop = 3; loop < 6; loop++) {
            SERVO_TORQUE[loop] = SERVO_TORQUE_ON;
        }
    } else {
        for (loop = 3; loop < 6; loop++) {
            SERVO_TORQUE[loop] = SERVO_TORQUE_OFF;
        }
    }
    for (loop = 0; loop < 6; loop++) {
        TRQ[loop] = 0x64;
    }
    calculateAngles(currCoords, cAngle, rawAngle);
    RsAllMoveWithTorque(hComm, sid, nsid, cAngle, 100, SERVO_TORQUE, TRQ);
}
void tryTransFeel() {
    for (loop = 0; loop < 6; loop++) {
        TRQ[loop] = 0x64;
    }
    if (info[3].mA < 100 && info[4].mA < 80 && info[5].mA < 60) {
        for (loop = 0; loop < 3; loop++) {
            SERVO_TORQUE[loop] = SERVO_TORQUE_OFF;
            cAngle[loop + 3] = -cAngle[loop];
        }
        for (loop = 3; loop < 6; loop++) {
            SERVO_TORQUE[loop] = SERVO_TORQUE_ON;
        }
        for (loop = 0; loop < 6; loop++) {
            copyAngle[loop] = cAngle[loop];
        }
        RsAllMoveWithTorque(hComm, sid, nsid, cAngle, 30,
                SERVO_TORQUE, TRQ);
    } else {
        for (loop = 0; loop < 6; loop++) {
            SERVO_TORQUE[loop] = SERVO_TORQUE_ON;
        }
        RsAllMoveWithTorque(hComm, sid, nsid, copyAngle, 30,
                SERVO_TORQUE, TRQ);
    }
}

void tryWallOne() {
    moveToW1[0] = 450;
    moveToW1[1] = 0;
    moveToW1[2] = 0;
    moveToW1[3] = -800;
    moveToW1[4] = 0;
    moveToW1[5] = 0;
    TRQ[0] = 1;
    TRQ[1] = 100;
    TRQ[2] = 5;
    TRQ[3] = 20;
    TRQ[4] = 5;
    TRQ[5] = 0;
    if (currCoords[0] > 40 || currCoords[0] < -40) {
        RsIdTorqueOnOff(hComm, sid[0], SERVO_TORQUE_ON);
    }
}
void tryWallTwo() {
        moveToW2[0] = rawAngle[0];
        moveToW2[1] = rawAngle[1];
        moveToW2[2] = rawAngle[2];
    }
        // RsIdTorqueOnOff(hComm, master_sid[0], SERVO_TORQUE_ON);
        RsIdTorqueOnOff(hComm, sid[1], SERVO_TORQUE_ON);
        RsIdTorqueOnOff(hComm, sid[2], SERVO_TORQUE_ON);
        RsIdMove(hComm, sid[1], moveToW2[1], 60);
        RsIdMove(hComm, sid[2], moveToW2[2], 60);
        // usleep(1*10*1000);
    }
    // RsIdTorqueOnOff(hComm, master_sid[1], SERVO_TORQUE_ON);
    // RsIdTorqueOnOff(hComm, master_sid[2], SERVO_TORQUE_ON);
    SERVO_TORQUE[0] = SERVO_TORQUE_ON;
    SERVO_TORQUE[1] = SERVO_TORQUE_ON;
    SERVO_TORQUE[2] = SERVO_TORQUE_ON;
    SERVO_TORQUE[3] = SERVO_TORQUE_ON;
    SERVO_TORQUE[4] = SERVO_TORQUE_ON;
    SERVO_TORQUE[5] = SERVO_TORQUE_ON;
    RsAllMoveWithTorque(hComm, sid, nsid, moveToW1, 100, SERVO_TORQUE, TRQ);
    // usleep(1*10*1000);
}
} else {
    RsIdTorqueOnOff(hComm, sid[1], SERVO_TORQUE_OFF);
    RsIdTorqueOnOff(hComm, sid[2], SERVO_TORQUE_OFF);
}

void tryWallThree() {

    SERVO_TORQUE[0] = SERVO_TORQUE_ON;
    if (angle[0] > 25) {
        moveToW1[0] = 250;
    } else if (angle[0] < -25) {
        moveToW1[0] = -250;
    } else {
        SERVO_TORQUE[0] = SERVO_TORQUE_OFF;
    }

    SERVO_TORQUE[1] = SERVO_TORQUE_ON;
    if (angle[1] > 25) {
        moveToW1[1] = 250;
    } else if (angle[1] < -25) {
        moveToW1[1] = -250;
    } else {
        SERVO_TORQUE[1] = SERVO_TORQUE_OFF;
    }

    SERVO_TORQUE[2] = SERVO_TORQUE_ON;
    if (angle[2] > 25) {
        moveToW1[2] = 250;
    } else if (angle[2] < -25) {
        moveToW1[2] = -250;
    } else {
        SERVO_TORQUE[2] = SERVO_TORQUE_OFF;
    }

    RsAllMoveWithTorque(hComm, sid, nsid, moveToW1, 5,
    SERVO_TORQUE, TRQ);
}