Universitat De Girona

Computer-aided Surgery and Medical Robotics

Abdomen Opening Prototype using Staubli TX-60 Robot

Authors:
Basel Alyafi
Zafar Toshpulatov

Supervisor:
Xavier Cufí

January 20, 2019

1 Introduction and problem definition

Speed and accuracy are two of the main benefits of using robotic systems. One application in assisting surgeons is to cut the skin of the patient. In this project, the goal is to open the skin, usually the tummy, using an industrial robot. Skin cutting is a challenging part of computer-aided surgery and an important task in medical robotics where accuracy is required in its highest rates. In order to develop a prototype for such an application, the STAUBLI TX-60 6 DoF arm is used. The procedure of skin cutting has been performed with the help of a surgery knife (scalpel) in the form of a circle which is described in Figure 1. For safety issues, it cannot be tested directly on a human body, so, we conducted our experiments on a soft material (e.g., cushioning).

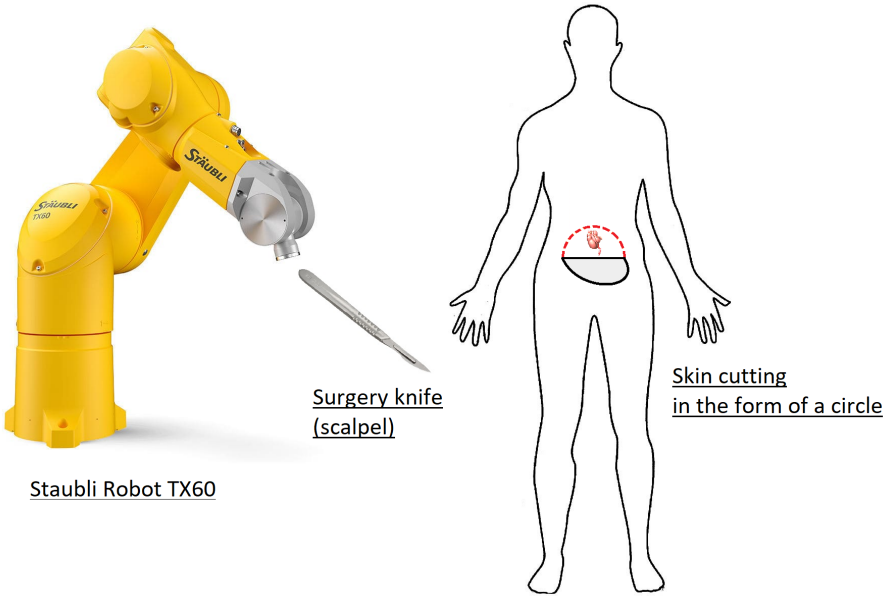

Figure 1: Skin cutting

2 Preparation

For this project, we chose Staubli TX-60 robot. This is a 6 Degrees of Freedom industrial robot which is mounted on horizontal base on the floor. Before starting the project, some installations were performed on the end-effector and the work-table. The project is divided into two main stages. For the first stage, we selected the trocar-like metalic as tool and mounted it on the end-effector using scotch and strips. For the second step, a scalpel (surgical knife) is selected as tool and mounted with strips. Thanks to Professor Xavier Cufi for providing scalpel. For safety issues, cushioning is used as a patient representation. We decided to fix the cushioning on vertical (standing) position using strips which is shown in Figure 2. Moreover, we

tried to get familiarized with Staubli Robot by training on MCP, moving the arms, reading and saving points. Staubli user manuals were of great use.

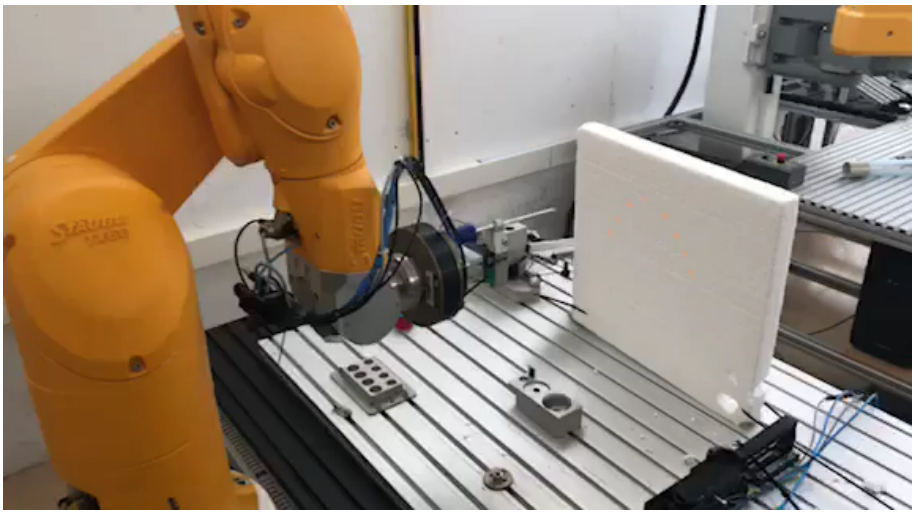

Figure 2: Supporting point and fixed cushioning in real practice

3 Design and implementation

In this section, we discuss the implementation of the robotic system used for opening the tummy in two steps. The first step is to use the trocar-like tool to read the landmarks coordinates only (end effector does not matter in this step). Then, we used, in the second step, the scalpel considering the end effector position and direction along the trajectory.

3.1 Step 1

This step was important to build a basis for the final step. It actually worked perfectly to initialize the job manually, then to automate the movement. The main stages here were to draw landmarks manually, fix the cushioning in the good place, read landmarks coordinates, and use the points for coding the final software. Finally, the application was transferred back to the robot control unit.

3.1.1 Draw the landmark using the marker

We started the first step with a simple approach drawing the landmarks over the cushioning manually using a marker. The landmarks were mainly 8 points (from point2 to point9) each in the form of an asterisk, this is shown in Figure 3 and 9. After that, we focused on working on Staubli TX-60.

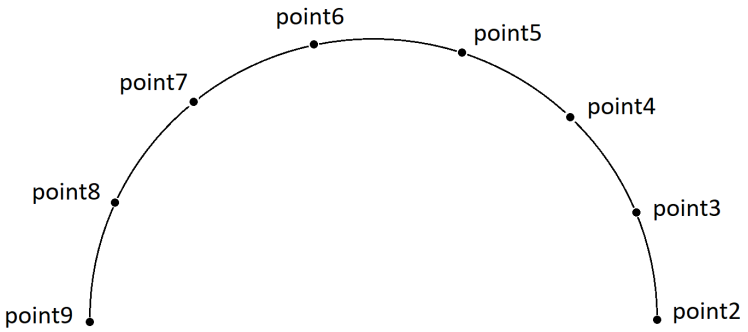

Figure 3: The landmark points in the form of circle

3.1.2 Taking points using Staubli TX-60

The next step is to take the position of trajectory points marked. The new application has been created through MCP controller that named `baszaf`. We created 10 new points in our application using F5 button. Figure 4 shows the initial point (point1) that was manually defined. Another assisting point was aimed at allowing the robot arm to be in a good position to start the trajectory without colliding with the cushioning. We saved this point as point10 and named it as the supporting point which is shown in Figure 5.

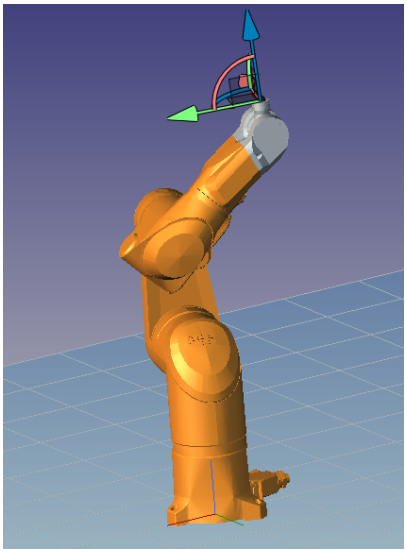

Figure 4: Initial point (point1)

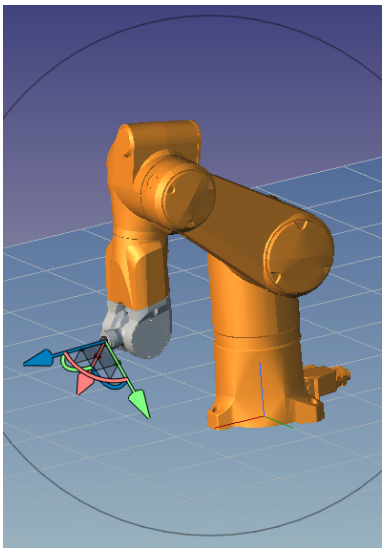

Figure 5: Supporting point (point10)

Other landmarks were defined in a similar manner, moving the arm to asterisk then reading. That was repeated from point2 to point9, see Figure 3. All position of points have been manually selected moving the arm with the help of 3 modes of movement ("Joint", "Frame" and "Tool"). Then we saved all the points and closed our application for editing in the next step.

3.1.3 Editing the application on PC

To move the robot along the path, we used Staubli Robotics Suite program for editing the application on PC using Val3 ver 6.8. Instructions avaialble in user manual were used to create cells and applications. The following options were selected as follows : Staubli TX60, version of the controller 6.8, mount, floor, Horizontal Power Connector. Then, we transferred our application (baszaf) from the Control Unit (CS8) MCP of the robot to the computer using transfer manager and setting IP address 84.88.129.201.

Thereafter, we edited the START and STOP functions considering our ten points. The following routine defines the movement that from initial point1 to supporting point10, then circle movement from point2 to point9. At the end, it will move back to supporting point and then finally again to the initial point. The routine STOP moves to the last point1 and it will stop the movement. We set the speed of movement to 15 percent. But in order to cut slowly, the cutting movement was set to 1 percent and from supporting point to point2 to 5 percent. To check syntax, we ran the code and there were no errors in it.

Listing 1: Start()

```
1  begin
2    userPage()
3    cls()
4    // Starting the trajectory
5    putln("program starter")
6    enablePower()
7
8    // speed is 15 percent
9    mNomSpeed.vel=15
10
11   // the trajectory has nine points in addition to two initial points
12   // move to the initial point
13   //open(flange)
14
15   movej(point1[0],flange,mNomSpeed)
16   putln("initial point reached")
17
18   // move to second supporting point
19   movej(point10[0],flange,mNomSpeed)
20   putln("supporting point reached")
21
22   // speed is 5 percent
23   mNomSpeed.vel=5
24
25   // moving to point2
26   movej(point2[0],flange,mNomSpeed)
27   putln("point2 reached")
```

```
28
29      // speed is 1 percent
30      mNomSpeed.vel=1
31
32      // moving to point3
33      movej(point3[0],flange,mNomSpeed)
34      putln("point3 reached")
35
36      // moving to point4
37      movej(point4[0],flange,mNomSpeed)
38      putln("point4 reached")
39
40      // moving to point5
41      movej(point5[0],flange,mNomSpeed)
42      putln("point5 reached")
43
44      // moving to point6
45      movej(point6[0],flange,mNomSpeed)
46      putln("point6 reached")
47
48      // moving to point7
49      movel(point7[0],flange,mNomSpeed)
50      putln("point7 reached")
51
52      // moving to point8
53      movel(point8[0],flange,mNomSpeed)
54      putln("point8 reached")
55
56      // moving to point9
57      movel(point9[0],flange,mNomSpeed)
58      putln("point9 reached")
59
60      // speed is 5 percent
61      mNomSpeed.vel=5
62
63      // moving back to supporting point
64      movej(point10[0],flange,mNomSpeed)
65      putln("supporting point reached")
66
67      // speed is 15 percent
68      mNomSpeed.vel=15
69
70      // moving back to initial point
71      movej(point1[0],flange,mNomSpeed)
72      putln("initial point  reached")
73      close(flange)
```

```
74 end
```

Listing 2: Stop()

```
1 begin
2   movej(point1, flange, mNomSpeed)
3   //open(flange)
4   putln("point1 achieved. Program finished")
5   popUpMsg("Pending movement commands have been canceled")
6   resetMotion()
7 end
```

3.1.4 Simulation

We also simulated our application using the Emulator and 3D View in order to see how the robot is moving according to the code that we did , see Figure 6. Figure 7 shows the trajectory of the movement in 3D view.

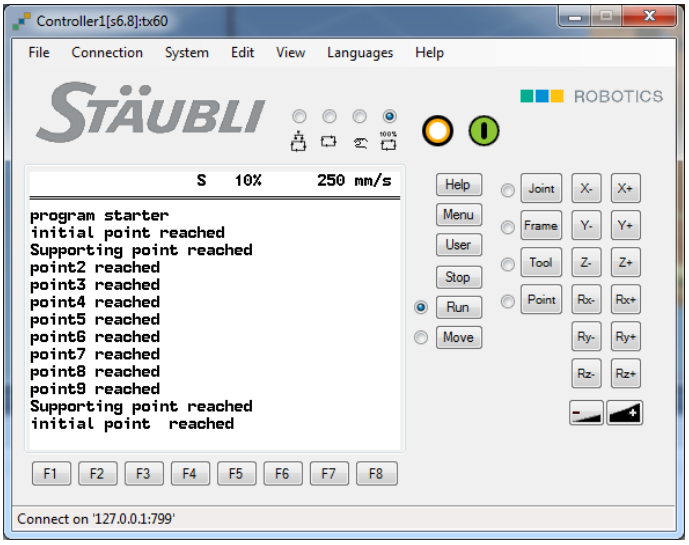

Figure 6: Successfully simulated application using MCP Emulator

3.1.5 Transfer the code back to the MCP

After a successful simulation, the modified application has been transferred back to the MCP for a realistic test on the robot. The robot performed the exact movement we designed and simulated. The capture of step 1 is available at this link: `https://drive.google.com/open?id=1H5YFA4uTVj2S7tvtPCNHMsWgdyfMn-ng`

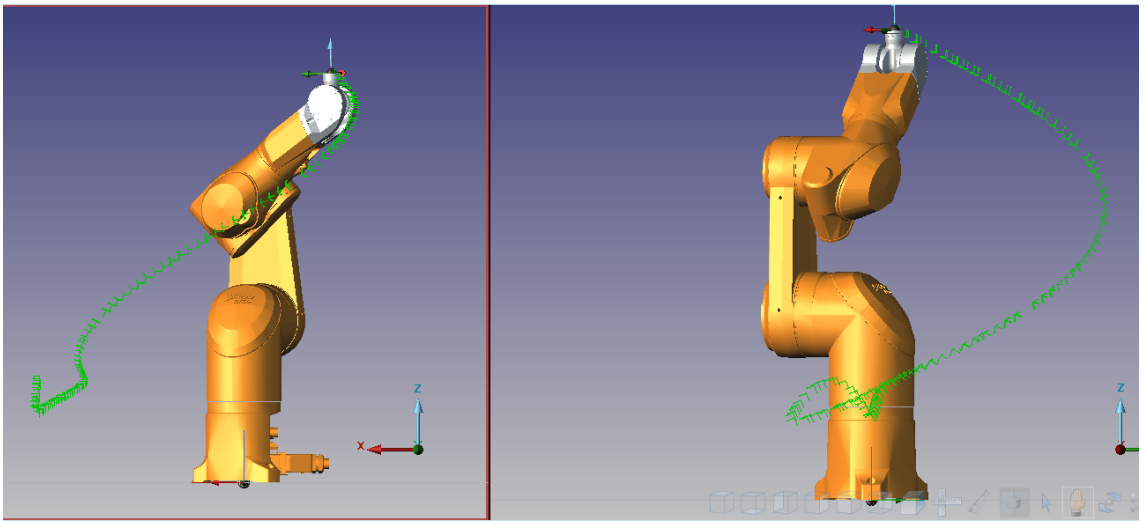

Figure 7: The last point from different perspectives, the green trace represents the locations the robots has visited

3.2 Step 2: mount the scalpel and cut

The above steps were tailored to try an experiment with a simple approach. In this step, we mounted the scalpel on the tool using strips to cut the red highlighted path on the cushioning which is shown in Figure 9 and the landmark is highlighted with green. For cutting, the end-effector orientation was necessary to be considered. Therefore, we reread the points 2-9 again considering the end-effector orientation. Figure 8 shows how the scalpel changes its orientation along the cutting path. We saved the new points and transferred them again to the computer. The execution was successful on Staubli Robotics Suite program as previous. A video describing the complete scenario from the first until the last point is available at: `https://drive.google.com/open?id=11dwxK0ezcvvKh8ieaO23fzoALymLFU2I`

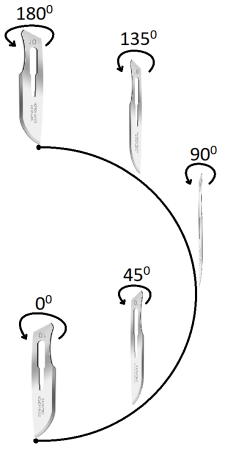

Figure 8: Rotation of scalpel for avoiding singularity

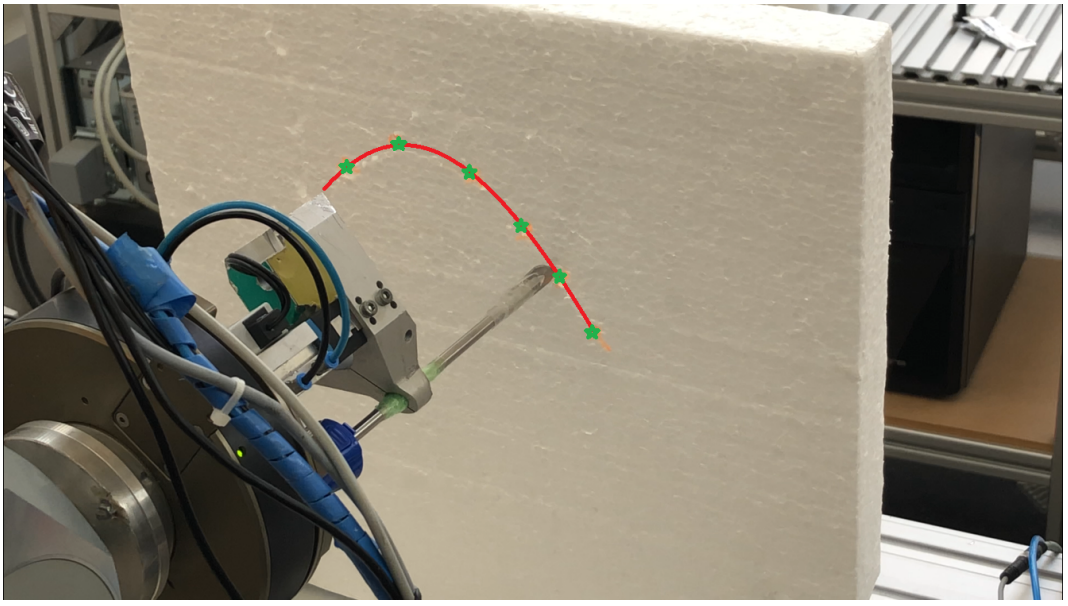

Figure 9: The mounted scalpel on the tool

4 Problems encountered

When working on this project, we came across a few obstacles which we eventually overcame.

- We need to mention that when we first installed our cushioning on horizontal position, we faced problems with limited workspace of robot and with other fixed tools on the work-table. Then, we decided to fix the cushioning in a vertical (standing) position using strips, see in Figure 2.
- We tried to use `movec` but we faced a problem with the direction of the circle. The robot was always moving vertically which did not help in our case.
- When we implemented the code and transferred to the MCP in the first step, the robot sometimes did not take the easy way between two close points (i.e between supporting point and point2 or point5 and point6). But it took the long way (clockwise instead of counterclockwise). The issue was fixed by the supervising Professor Xavier Cufi. The reference frame of the robot was stuck in our working space and we changed this to another direction.

5 Conclusion

Overall, that was a beneficial experiment on how to use an industrial robotic arm for a surgical purpose. In reality, surgical robots are much more precise and flexible than industrial ones, but as said before, the idea is to come up with an application and try to implement using available tools.