Rob 1 Assignment 4, Group 1

Martin Moghadam, Martin Franzen
Kalle Grafström, Audrius Palisaitis, Robertas Jacauskas

January 18, 2010

Abstract

The programs are developed in modules so that i.e. LetterFonts C# program can be substituted for another program to write letters with.It should just follow the Pathplanner plugins specifications that parse the text file from the LetterFonts program and translates the Cartesian coordinates to joint coordinates. Which can be simulated and uploaded to the Fanuc LR Mate 200i. The simulation in RobWorksStudio is performed before any upload to the robot is commenced to protect it from harm itself or the surroundings. When the robot receives an upload it performs the milling of the letters in the Styrofoam.

Contents

1 Introduction

This assignment is apart of the Rob1 course. As a starting point this assignment is supposed to give the students experience of developing a small application that make use of the topics learned so far in the course, such as making plugins for RobWork, using the RobWork interface and additionally to that the students should interface the application to a physical robot and using a tool that perform some interaction with the surroundings.

The assignment given to this group of student will make use of a Fanuc LR Mate 200i that is setup to physically carve in styrofoam with a milling tool mounted on the sixth joint of the Fanuc.

The user of the robot should be able to type in a sentence in a GUI with a given font and size. From this GUI a task description that tells the robot the lines it has to follow to mill the letters in the foam, is created. The task description must then be translated into joint coordinates that will be uploaded to the Fanuc robot.

But before the joint coordinates is uploaded to the robot it must be simulated in RobWorkStudio to make sure that it doesn't damage itself, the milling tool or the surroundings. Some thoughts of how the the tool should be moved when it's airborne and when it is docked to the foam and maby how and where to start and stop the milling tool. Another part of the assignment is to specify the work area where the foam is placed.

2 Specifications of the Robot Arm

It's necessary to verify the Fanuc LR Mate 200i specifications before any work on the RobWork plugins can commence, because the robot in the laboratory is a LR Mate 200iB and in RobWork it's a LR Mate200iC model. It is essential that the LR Mate model in RobWork is true to the real robot in the laboratory. The model in RobWork (LR Mate200iC) is depicted in the figure below left and the model in the laboratory (LR Mate200iB) is depicted below right. As can be seen from the figure, the models in some points are different however in other cases they are similar.

Figure 1: Fanuc Models, left: LR Mate 200iC, right: LR Mate 200iB

Items		LR Mate 200iC (RobWork)	LR Mate 200iB (Laboratory)
Axes		6	6
Payload – wrist (kg)		5	5
Reach (mm)		704	700
Repeatability		+-0.02	+-0.04
Interference Radius (mm)		181	180
Motion Range (degrees)	J1	340 (360 option)	320
	J2	200	185
	J3	388	316
	J4	380	380
	J5	240	240
	J6	720	720
Motion speed (degrees/s)	J1	350	180
	J2	350	180
	J3	400	225
	J4	450	400
	J5	450	330
	J6	720	480

Figure 2: Specifications of LR Mate 200iB and LR Mate 200iC

Find the similarity and difference points in the table above from which an analyze can be commenced. As can be seen the main point to analyze is the motion range that is a little bit different between the two models. But physically are the two robots similar with respect to physical length of the joint arms, it's important to get a correct physical representation of the robot in RobWorkStudio. Other points from the table are not so important for use, among this is payload, reach or motion speed, because they are more related to efficiency.

2.1 Dimensions of the two Models

By analyzing the datasheet of the two models it can be concluded that the physical appearance of the two Fanuc robots are similar in dimensions. They have six revolute joints and simimilar joint arm dimensions. Both models are depicted in the figure 1. Then DH-parameters can be determined using the datasheet and physically inspecting the robot in the laboratory to verify the dimensions.
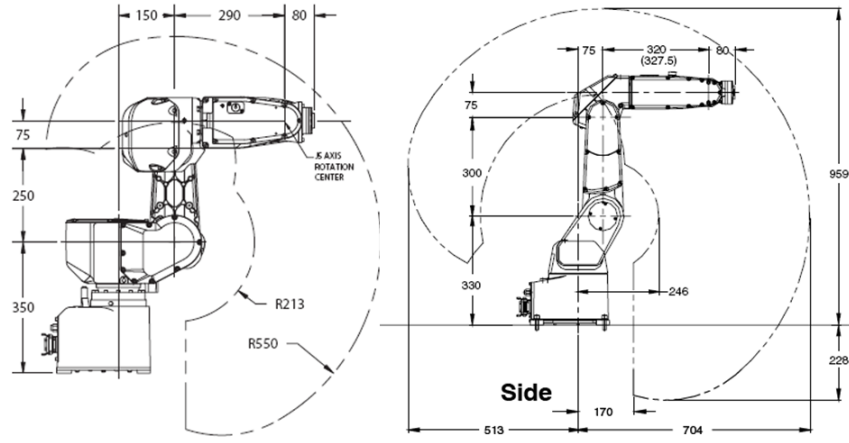

Figure 3: left: LR Mate 200iB (Laboratory), right: LR Mate 200iC (RobWork)

Before determining the DH-parameters, it's important to understand what the meaning of the parameters are, they are defined below:

a_{i-1} − alpha is angle between $Z_i$and Z_{i+1}which is measured about axis X_i
a_{i-1} − here distance in mm from $Z_i$and Z_{i+1}which is measured about axis X_i
d_i − distance from $X_{i-1}$to $X_i$measured along axis Z_i
θ_i − theta is angle between $X_{i-1}$and $X_i$measured along axis Z_i

To determine the DH-parameters we need to know the dimensions of both the models, they are shown in the figure 3. At the left LR Mate 200iB model which we use in the laboratory and on the right LR Mate 200iC model which we use only to simulate. The determined parameters are shown in the figure 4.

Model	LR Mate 200iB (Laboratory)				LR Mate 200iC (RobWork)			
i	α_{i-1}	a_{i-1}	d_i	θ_i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1	0	0	0	θ_1
2	-90	150	0	θ_2	-90	75	0	θ_2
3	180	250	0	θ_3	180	300	0	θ_3
4	-90	75	290	θ_4	-90	75	320	θ_4
5	90	0	0	θ_5	90	0	0	θ_5
6	90	0	0	θ_6	90	0	0	θ_6

Figure 4: DH parameters of LR Mate 200iB and LR Mate 200iC

From figure4 it can be seen that the real model and the model which will be used for simulations is quite similar, however some links are different. However differences are not huge so a decision was made that the model can be used for simulation. The LR Mate 200iB was constructed by using some frames from and the some 3D model objects of the LR Mate 200iC. The model seen in RobWorkStudio are a mix of LR Mate 200iC and LR Mate 200iB, but the dimensions and DH-parameters are determined from the robot in the laboratory.

2.2 Construction of the Cage

After verification of the details, measuring the dimensions of where the robot is fixed were made. The LR Mate 200iB stands in the cage as shown in the figure 5 (here is LR Mate 200iC model). Dimensions of the cage measured and depicted by the RobWorkStudio were created using the XML file. Robot hand stands on the table and around the robot hand is a cage which guarantees the safety of the operator. Position of the LR Mate 200iB in the cage is in the left up corner (if watched from the front view).
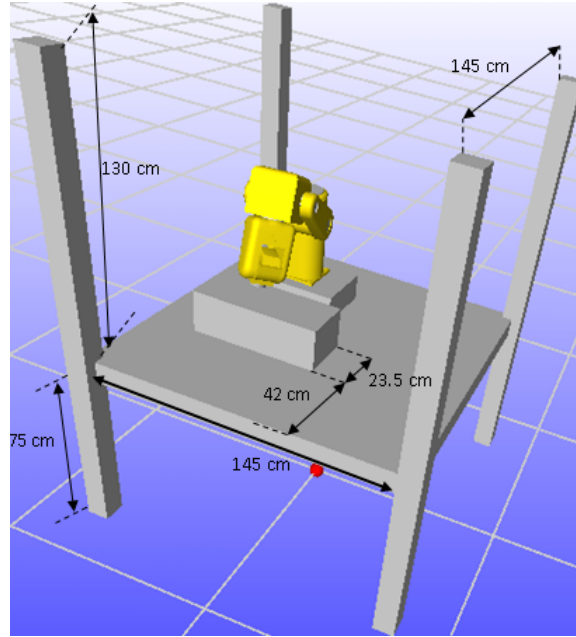
Figure 5: Front view of the LR Mate 200i

Figure 6 shows the robot position from top view and dimensions of the cage
where LR Mate 200i stands. As can be seen the shape of the table is square,
the real table width is a little bit bigger. A smaller is used because it makes
it easier to avoid any damage to the cage. A defined smaller work space gives
more safety.

Lengths were measured by using the 50 cm ruler. Of course a bigger one would
have been better, so there are probably some errors in the measurements $\pm$ 2.5
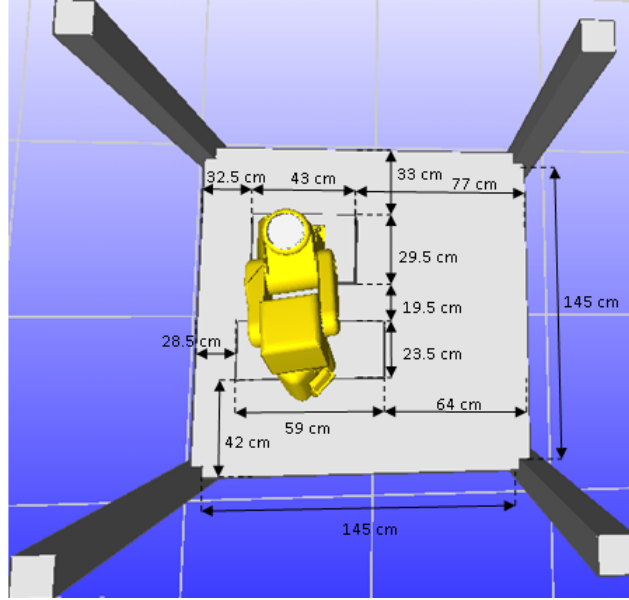cm. It's not guaranteed that all values are precise enough.

Figure 6: Top view of the LR Mate 200i

Regarding this, max theta angles can be determined. They will limit the movements of the joints. If this is not done, there is a risk in damaging the cage or robot.

2.3 Construction of Joint Angles

Joint angles means the interval of turning between min and max. By default values were set as shown in the figure 7, however a change of those do guarantees that the robot hand can't move too much. If these values are allowed and something is wrong in the state sequence there could be a risk of damage to the window, robot or even worse injure people around. So that is why a desicion was made to limit the turning angle of the robot.

```
72      <PosLimit refjoint="Joint1" min="-180" max="180" />
73      <PosLimit refjoint="Joint2" min="-180" max="180" />
74      <PosLimit refjoint="Joint3" min="-180" max="180" />
75      <PosLimit refjoint="Joint4" min="-180" max="180" />
76      <PosLimit refjoint="Joint5" min="-180" max="180" />
77      <PosLimit refjoint="Joint6" min="-180" max="180" />
```

Figure 7: Description of LR Mate 200i in Xml (by default)

Before start an illustration of the robots position in the cage is needed to show

all measured dimensions. With all dimensions and joint positions determined, modelling of the turning angles of joint arms could be done. In the XML file each joint has min and max and is set by θ_{min} and θ_{max} here theta is angle limits of turning.
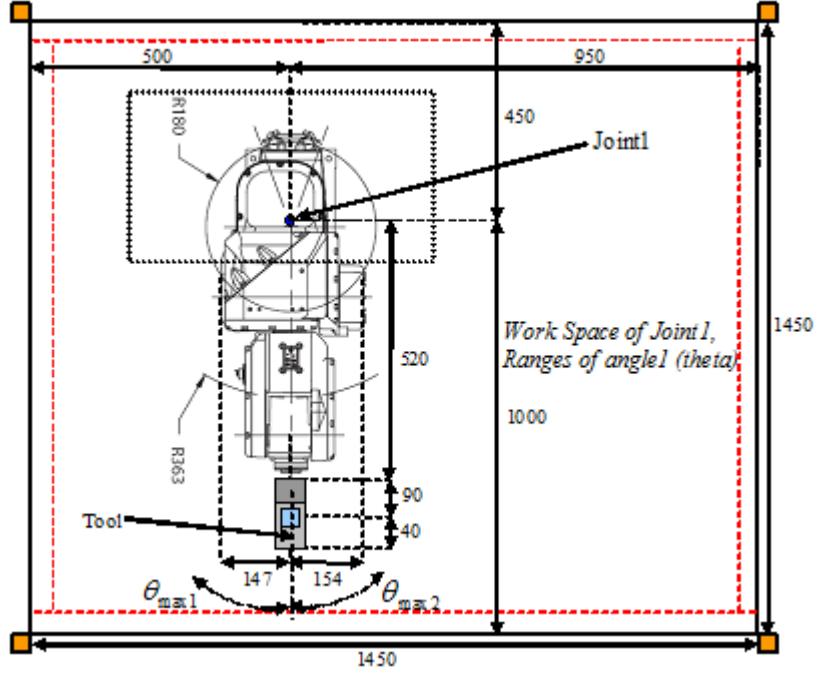

Figure 8: Work space of Joint1 (Top View)

According to figure 8 max movements or angles of joint1 can be calculated:

$\theta_{min} = \theta_{max1}$
$\theta_{max} = \theta_{max2}$

$$\theta_{max1} = arctan\left(\frac{500 - x}{520 + 90 + 40}\right) \tag{1}$$

$$\theta_{max2} = \frac{\pi}{2} + arctan\left(\frac{500 - x}{520 + 90 + 40}\right) \tag{2}$$

In equation 1 value x is wrong. Depending on the shape of the robot x > 154 is used, x = 180 were chosen. Than max movement ranges of the joint 1 could be calculated:

$$\theta_{max1} = arctan\left(\frac{500-180}{520+90+40}\right) = arctan\left(\frac{320}{650}\right) \approx 26deg$$

$$\theta_{max1} = \frac{\pi}{2} + arctan\left(\frac{500-180}{520+90+40}\right) = \frac{\pi}{2} + arctan\left(\frac{320}{650}\right) \approx 116deg$$

We can illustrate angles of theta from another view (view from left). Regarding formula 1 we can also calculate other angles, however we can not do all of them, because joints are dependent each other and we only modeling here statically.
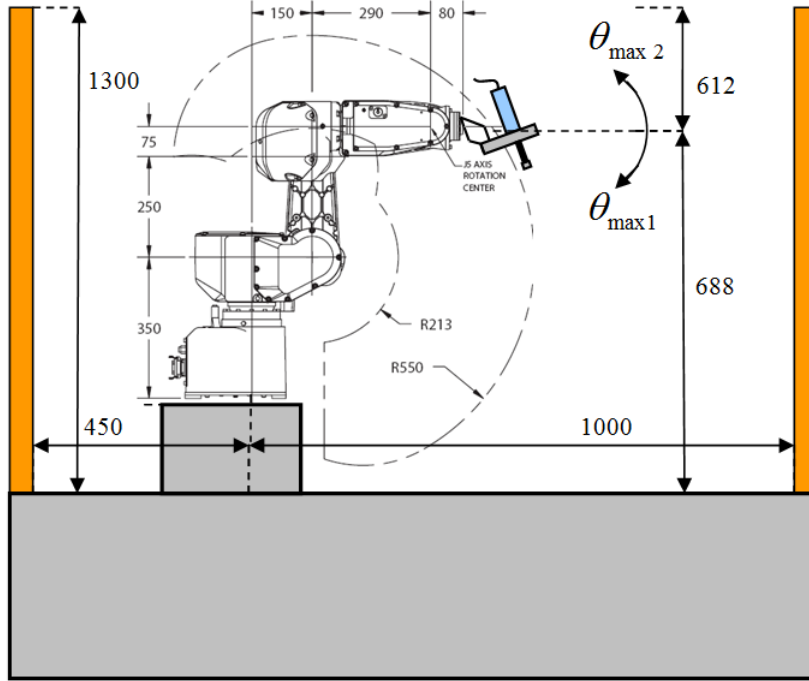

Figure 9: Work space (Left View)

Our chosen calculated angles are shown in the figure 10. As we can see angles are much more less than it can be, because of the safety.

```
70    <PosLimit refjoint="Joint1" min="-26" max="116" />
71    <PosLimit refjoint="Joint2" min="-30" max="90" />
72    <PosLimit refjoint="Joint3" min="-120" max="0" />
73    <PosLimit refjoint="Joint4" min="-45" max="45" />
74    <PosLimit refjoint="Joint5" min="-60" max="60" />
75    <PosLimit refjoint="Joint6" min="-0" max="0" />
```

Figure 10: Description of LR Mate 200i in Xml

2.4 Modeling the Tool Frame

Requirement was to have few parameters which let us easily calibrate the tool frame (drill frame). Before that we need to measure the drill which shown in the figure 11 (not very nice). As we see from the figure 11, there are two unknowns a and b. Values of a and b let define the position of the drill. As we see than we changing b than we can shift to right or left and it depends mostly of the drill construction. Second one called a is height control of the drill and it possibly changed many times. For instance if we broken the drill or changed the new one.
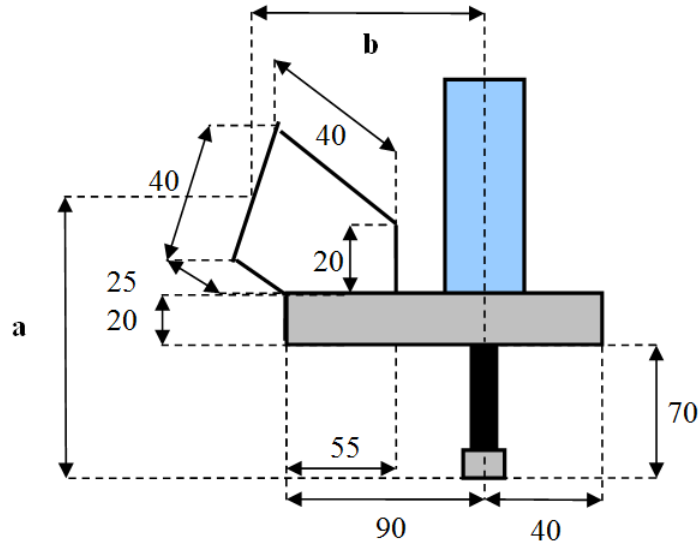

Figure 11: Dimensions of the drill (Tool)

Setting of the tool frame were done throw intermediate tool frame. Intermediate tool frame rotate frame by 45 deg, because of the drill construction. Then we can easily calibrate tool frame according the created intermediate frame. As shown in the figure 12. There are two values a and b (for more details look at the figure 11) value b is width of where drill stand and value a is deep of the drill.

```
68  │ ┌  <Frame name="Intermediate_Tool_Frame" refframe="Joint6">
69  │ │     <RPY> 0 -45 0 </RPY> <Pos> 0 0 -0.08</Pos>
70  │ └  </Frame>
71  │ ┌   <Frame name="ToolFrame" refframe="Intermediate_Tool_Frame">
72  │ │      <RPY> 0 -90 0 </RPY> <Pos> -0.1 0 -0.1</Pos>
73  │ └   </Frame>
74  │ │
```
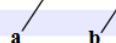a   b

Figure 12: Dimensions of the drill (Tool)

Result of the intermediate and tool frames shown in the figure 13. Configuration of it shown in the XML file extract (figure 12). Regarding this we can easily adopt any desired tool. For instance if we get longer drill or different dimensions of tool than we can easily modify XML file and work.
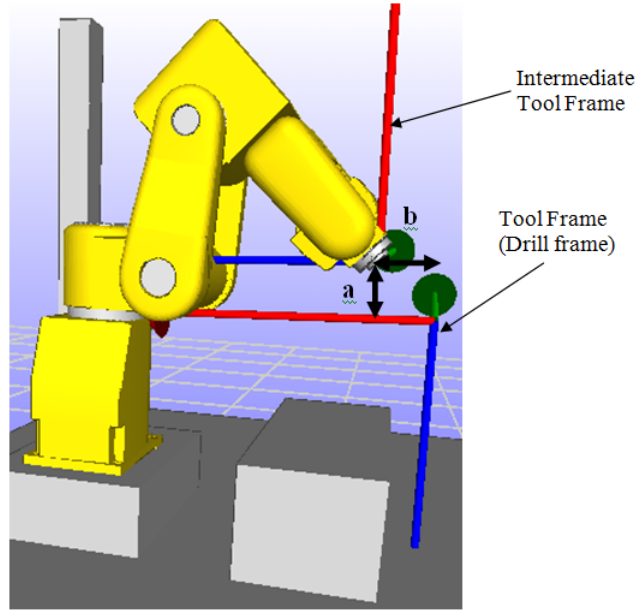

Figure 13: Tool frame (drill frame)

As we can see from the figure 13 tool frame can be easily modified. Tool frame is important for the future parts, like path planner. Path planner takes Cartesian coordinates from the letter model and moves tool frame regarding it. By moving tool frame all join values change and we need to parse them and send to real robot, but this discussion is out of this chapter scope.

2.5   Construction of XML File

Xml file were constructed considering to the few of the things:

1. Base frame setup. Discussion where set the base frame. It chosen to put center down of the table (cage).

2. Dimensions of the cage. All dimensions were measured and shown in the chapter: "Construction of the cage". Regarding this XML file has parameters which depict the cage.

3. Dimensions of the LR Mate 200iB. Regarding of dimensions were constructed all joints in the XML file.

4. Model graphics of the LR Mate 200iC. Regarding this was used 3D model to illustrate simulate robot behavior.

5. Collision setup. There was added reference to collision setup which indicates collisions of the LR Mate 200iC model.

2.6   Construction of Joints

LR Mate 200iB has six axes. Each axis is revolute, dimensions given by manufacturer. Regarding this we can start to create joints in XML. We can do by two ways:

1. Calculate positions of the joints by using template of the Fanuc 200i model. This was given by RobWork, however model is not quite the same.

2. Calculate DH parameters and regarding this construct the XML file with joints

We use first approach because in the assignment there was no appointment how to construct. First approach was used for someone but parameters were not the same, so we need to recalculate all parameters. Before start to construct the joints we need to define where put a base frame. From the base frame all other joints will be constructed. There were discussions where to put the base frame. One opinion was to put base frame at the one of the table corners, another one where robot centre stands and last one down centre of the table. We decided to choose last one, because there was easiest way to start construct table and after that joints. Reason was that the base frame will stand independently to any of the objects around it. Second approach will be more useful, reason of easier understand and adopt new changes of the robot. For instance if the lengths Robot arm changed. We actually need change DH parameters in the XML by entering into places like shown in the figure 14.

```
<DHJoint name="Joint1" alpha="0" a="0" d="0" offset="-90" state="Active" />
<DHJoint name="Joint2" alpha="-90" a="0.15" d="0" offset="-90" state="Active" />
```

Figure 14: Example of configuration by DH parameters

3 Letter Font Application

This section presents letter font model GUI application, the inputs comes from a user, and user can choose a set of settings, the main one is the font, font style, size of letter. This application writes a set of drilling points in to txt file, the txt file also can choose arbitrary a user. The letter font model is implemented under vision (image processing), so all theory from image processing could be applied there.

The purpose of this application is to return letters drilling points, and these drilling points would be used to performing drilling (carving) by robot arm Fanuc 200ib (figure 15).
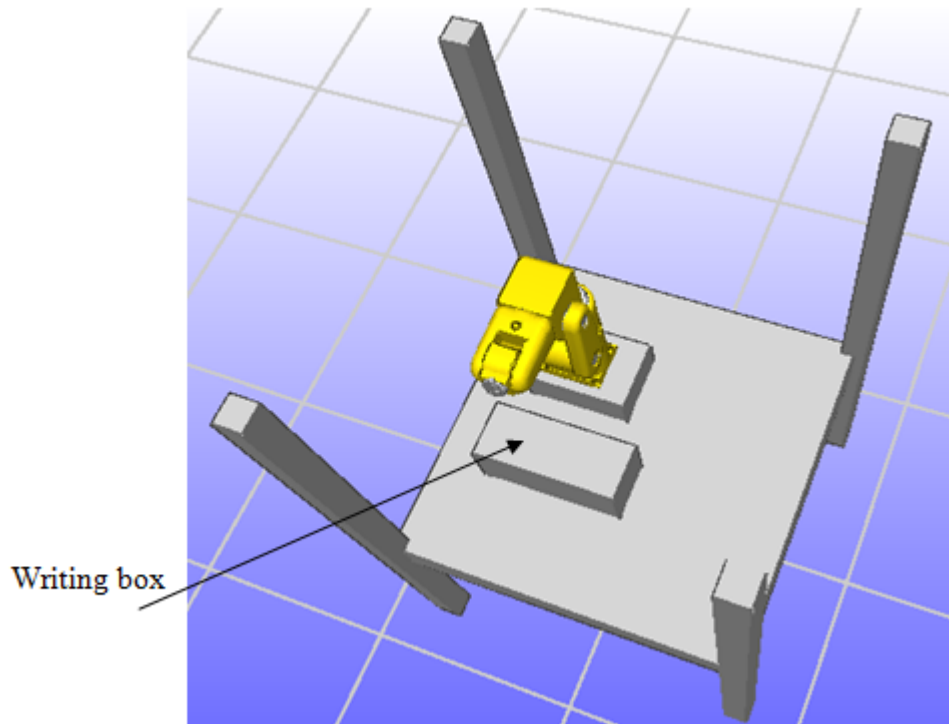

Figure 15: Writing box position

Letter font model is an important step for performing actual (real) letter carving

on foam, another step would be inverse kinematics, which path planner uses for planning movements between frames. The application main window is shown in figure 16, also there is provided a description of application elements.
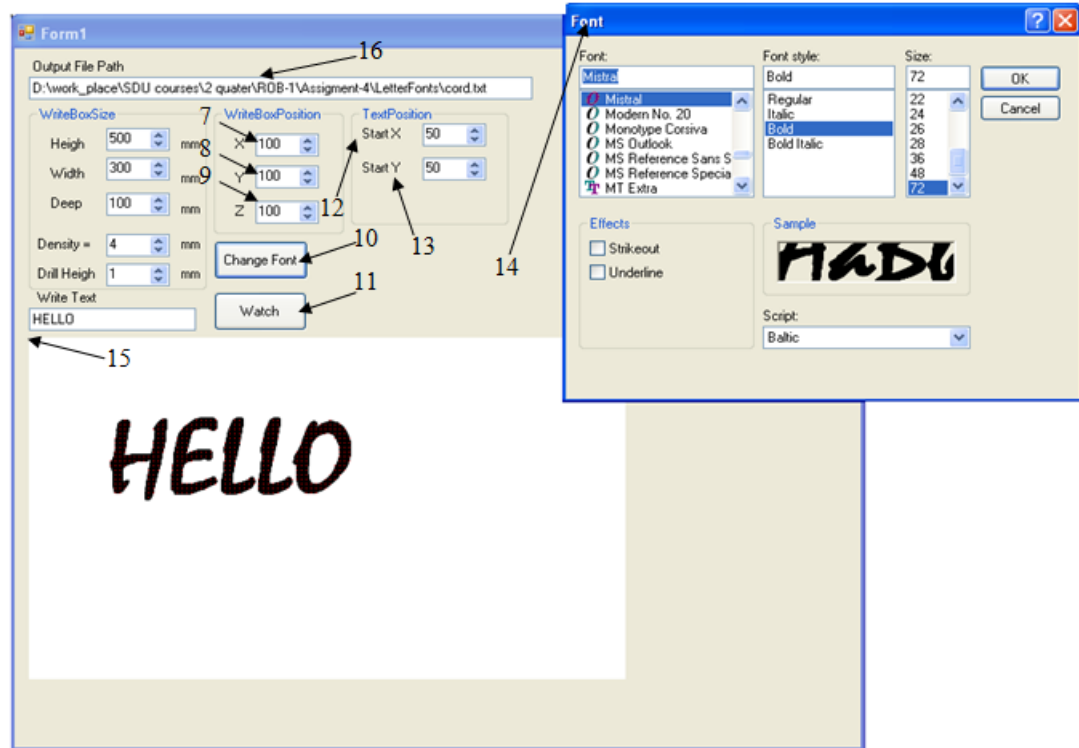

Figure 16: Letter font application

1. Sets writing box height

2. Sets writing box width

3. Sets writing box deep

4. Set the density of drilling points (red points)

5. Sets the drill height, in other words, how deep we want to drill

6. The place of letter or text

7. Set the writing box position according X coordinate (in Cartesian domain)

8. Set the writing box position according Y coordinate (in Cartesian domain)

9. Set the writing box position according Z coordinate (in Cartesian domain)

10. Call the font window

11. Display the text on writing box

12. Set the text position according X (in Cartesian domain) on writing box

13. Set the text position according Y (in Cartesian domain) on writing box

14. The font window, where we can the size, font, font style of text

15. The start position of writing box

16. The path of txt file, where are stored each drilling points

First by application we set the txt file path, the txt file stores each drilling points information, this information we can see also in C# application (figure 17). Second it is necessary to set writing box position, because according this position would be wrote drilling points, and other settings are optional.

Figure 17: Letter font application console

As we can see from figure 17 each letters drilling point are separated, because the image is consider as a set of letters. We could do not separate the letters, but then it would be consider as a set of drilling positions, and we would not be able carving out letters, just drilling. We could react or take some actions then the letters drilling points finish, or skip it.

3.1 Letter labeling application

We deal with image in byte mode and the image is represented like figure 18. As we can see the image is composed of R-red, G-green, B-blue bytes and padding bytes. We are interesting only in RGB bytes, the padding bytes describe how image is stored in memory, and they do not have any influence of displaying image. The programming becomes complex, because in order to jump to next images row we need count offset (which is equal (stride images width)), and in order to access to next pixel we need to jump by three bytes.
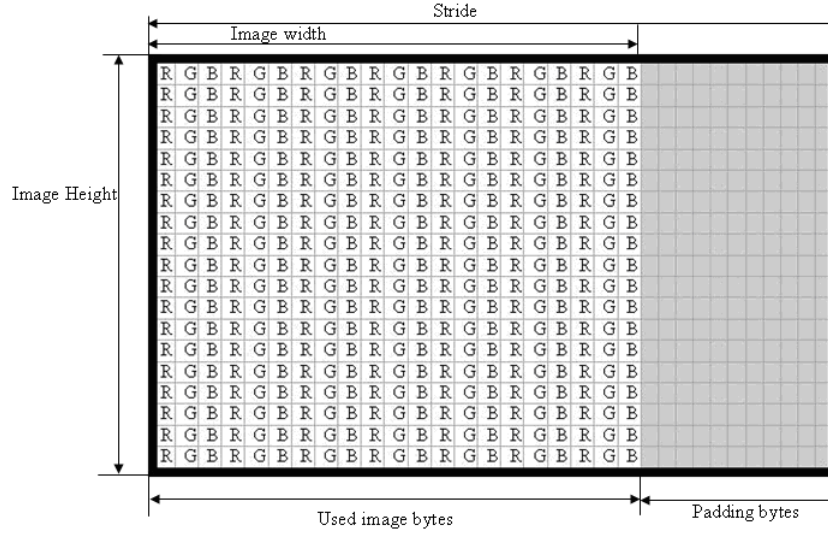
Figure 18: Image in byte mode

We deal with two images, one image is an original, and another image is was a clone of original image. In clone image we access to each pixel bytes, and use such image as an array, where we change values, we do it in such way because it is faster, and easier access to write values, rather than to create a huge array, and write by own functions.

Was used 9 pixels connectivity matrix figure 19, there are a lot of options but we chose such one, it is not too big, by the way and easy manages such shape of matrix (not like triangles or cross shape).

1	2	3
4	5	6
7	8	9

Figure 19: 9 pixels connectivity matrix

Connectivity matrix is used to take values from original image and write the result to a clone image. Another reason of choosing 3x3 connectivity matrix is what we loose only images corners pixels. It was possibility to choose 2x2 matrix, but the time of image processing would significantly increase, and we would also loose also the corner pixels.
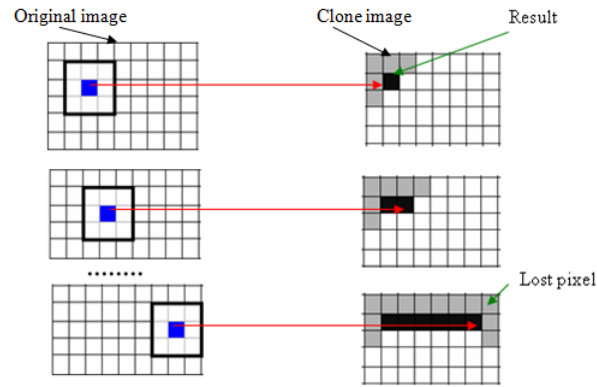
Figure 20: 9 pixels connectivity matrix over image

The letters (objects) labeling algorithm is shown in figure 20. At the beginning the image is converted to binary, it allow us easier to deal with information, we set non-white pixels to black.
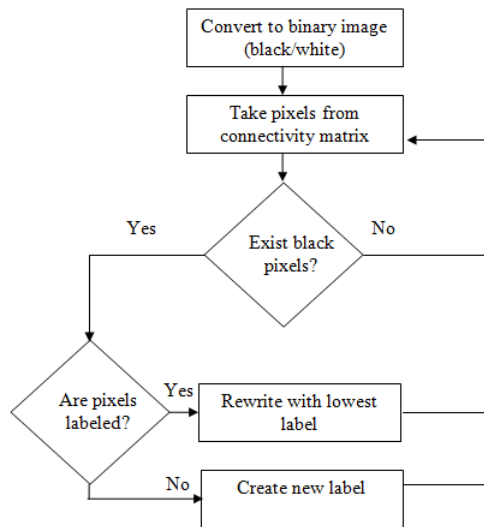

Figure 21: Letters labeling algorithm

When we have a binary image we have only two values: black which represent the letter (or object), and white which we assume as background. In the next step we take pixels from connectivity matrix and check if exists black pixels (which belongs to letter or object), if there are no black pixels we move the connectivity matrix by one pixel in particular direction and repeat the black pixel checking. But if there are black pixels, we perform checking in order to

figure out or we need to create a new label (detected new object) or some of these pixels belongs to before detected letter (or object), and it is enough to rewrite the previous labels. The example of labeling text letters is shown in figure 22. We have a binary image this image is called clone image it is a copy of original. By performing letters labeling algorithm (figure 21) we can detect how much different letters (objects) we have. The important thing is what we should twice repeat letters labeling algorithm, at first time we arbitrary choose from which images corner to start, for instance it is convenient to start from the top/left corner, the second time of repeating algorithm we need to choose an opposite corner it would be bottom/right.
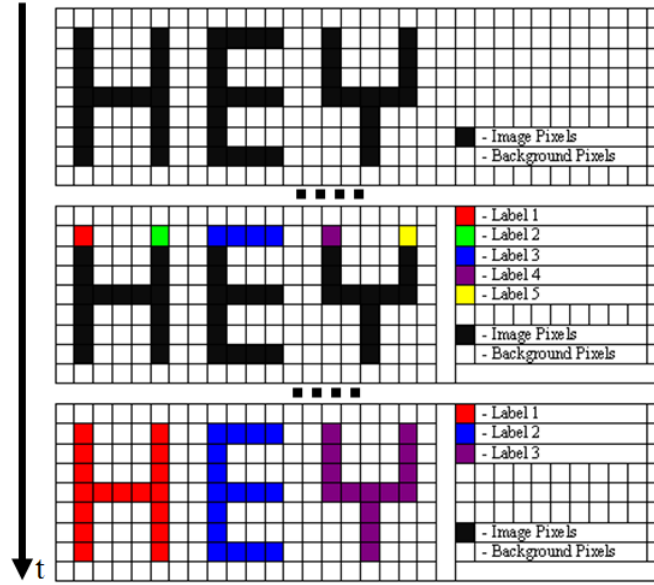

Figure 22: Letters labeling example

If we would not perform algorithm from opposite corner, the algorithm would detect more letters (objects) then there exists, in such letters as U, Y, H, V, W , because the connectivity matrix is too small, but if we choose connectivity matrix quite large, there are risks: detect two letters as one letter, loose more pixels (near image corners).

3.2 Letter Font Application Extension

At the moment the letter font application deals with text as an image, in other words, the letters are consider as image (which we can save, paint, change color). So it is possible to load image directly (this feature is not implemented on final application, because the task was to deal with letters), so by having this

18

application source code and knowing basics of objecting oriented programming, it easy to load any picture here as shown in figure 23.
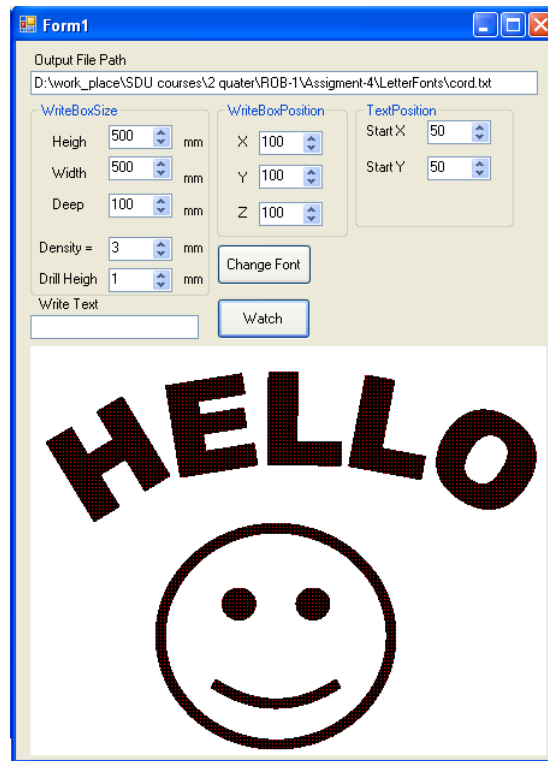

Figure 23: Letter font application

The interesting thing is what all image processing knowledge we could apply here, such as edge detection, Hough transformation etc., and this is mean what this letter font application is universal, we can easy make extensions.

4 Parsing

5 Path Planning

6 Testing the System

7 Conclusion

References

[1] John J. Craig. *Introduction to Robotics Mechanics and Control Third Edition*. Prentice Hall, 2005.

8   Appendix