1. Teoretisk referensram

2.1 Image processing

A picture of a person was first converted to a gray scale picture. Then a binary picture was generated, the value of each pixel is either 1 or 0, with 1 representing a point along the edge and 0 representing a blank point. The robot was designed to draw the edge like a human being.

The computer scanned the whole binary picture to identify the main composition. After the computer identified the first point of an edge segment, it would figure out how to connect all the pixels. Thus, longer lines representing main composition were constructed.

The drawing robot was developed using a 6 degree of freedom robot (x,y,z,yaw,pitch,roll), coordinates of a Cartesian coordinate system attached with w (yaw), p (pitch), r (roll) which are the rotation angles of x, y and z axes.

The position information was derived from the binary picture. The pixels of the portrait were represented by a matrix, after the previous procedure, the pixels of the drawing lines had been recorded according to the sequence. The value of the row and column of the pixels were then mapped to construct the x and y value as indicated

//put the figure

The array needed is saved as a .CSV file to be imported into the Roboguide virtual environment.

 Figur 0-1 Ikonen för programmet NetTool.

2. Genomförande

First in the same folder of the program, put the image we want process there, then create a test.csv file then we enter the origion (by go to the center of the paper then write down the x and y value) and ScaleValue and LiftPenHeight we want, then it ask us to select a file, we change file type to ALL FILES and select an image located within the same folder of the MATLAB program, then the output will go to test.csv

3.1 Initialization

First we clear previous stuff then we use ScaleValue=300; To scale value for x and y, default scale of 1 have x and y value around 0.3 and set LiftPenHeight=10; to set value for z which is the hight we lift pen we also set DistanceWhichLiftPen=0.002*ScaleValue; I don't suggest change this one (even change only change 0.002), because it is for if two points are further than this then we will lift pen (give z value) to the end and start point.

we also set origin = [144 134]; origion of the graph, write as [x y], we can get our own by trying using the robot delta = 0.001; is the distance apart we want record (the smaller the more detailed drawing is) boundary =18; we can change to a different value but this works fine. ZOffset=113.215; is for the height of the robot arm, we need test in real world first then set to here, if enter -100 then all z value will be 100 less. finally for Yaw Roll and Pitch which we need test on our robot first then put Yaw=-176.629; %W value Roll=-2.447; %P value Pitch=112.57; %R value

3.2 Acquire an image from user selection

I = imread(uigetfile); the image need to be in the same folder as this MATLAB program (different folder won't work), after it ask you to select a file, file type change to ALL FILES, then can see images and select. Read an image file: figure; imshow(I); show the image we selected

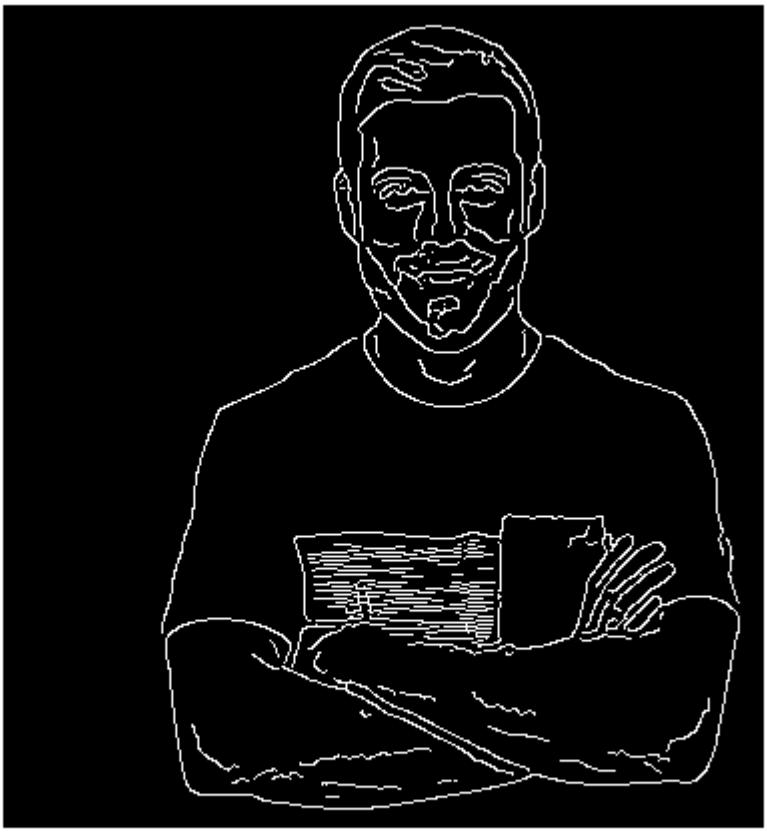
3.3 Edge detection and skeletonization

```
[BW,thre] = edge(rgb2gray(I),'Canny',[0.0813 0.1281]);
```

Basically we convert the image to black and white image by using RGB to Gray method, then finds edges using an approximate version of the Canny edge detection algorithm with the provided variables.

```
BWseg = bwmorph(BW,'skel',Inf);
```

Then bwmorph function applies a specific morphological operation to the binary image BW.



3.4 Detect start and end points and find the end point and sort the points

Detect start and end points:

[B,L] = bwboundaries(BWseg,'noholes');

bwboundaries function traces the exterior boundaries of objects, in our case th boundaries of holes inside these objects we don't want trace.

imshow(label2rgb(L, @jet, [.5 .5 .5]))

Here label2rgb function converts a label image, L into an RGB color image for the purpose of visualizing the labeled regions.

Here jet is colormap as a three-column array with the same number of rows as the colormap for the current figure.

hold on

```
for k = 1:length(B)
   boundary = B{k};
   plot(boundary(:,2), boundary(:,1), 'LineWidth', 3)
end
```

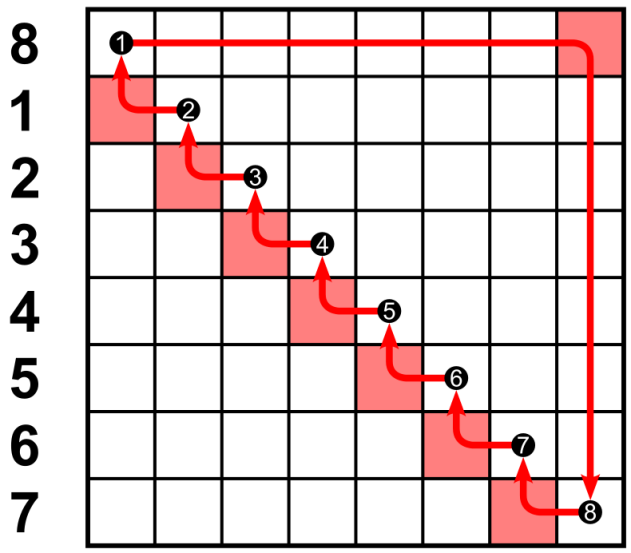
Then we show the user a graph and draw the the path on the graph, looping using for loop.

To find the end point, we final all points both present in circularly shifted elements in boundary array by 1 and -1 positions, because if those points are connected then when circular shifting they will move together regardless of which direction.

```
edgeind = find(all(circshift(boundary,1)==circshift(boundary,-1),2),1);
```

The find function returns the row and column indices of the nonzero entries in the matrix.

The all function tests along the first array dimension of circularly shifted 1 bountry.



//how circular shifting works, image from wiki

Then we sort the points by again circularly shift the elements in array boundary by –edgeind+1 positions, edgeind is edge indices of the row and column of the nonzero entries in the matrix, we use this value because it is used for digraph inputs whose edge weights contain some negative values.

```
boundary = circshift(boundary,-edgeind+1);
```

```
boundary = boundary(1:ceil(end/2),:);
```

The ceil function rounds values to the nearest integer toward positive infinity

The colon is one of the most useful operators in, it can create vectors, subscript arrays, and specify for iterations.

Here 1:ceil(end/2) creates a unit-spaced vector with elements [1,1+1,...,ceil(end/2)].

The ceil function rounds values to the nearest integer toward positive infinity.

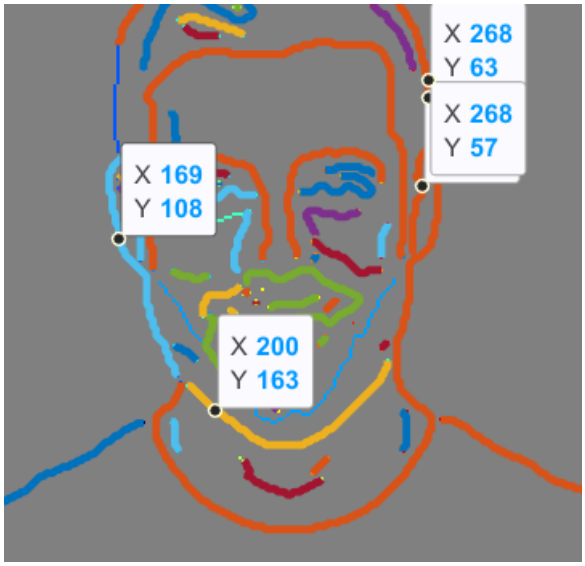
Here boundary(1:ceil(end/2),:) are indexing expressions for a matrix boundary that contain a colon, it acts as shorthand to include all subscripts in boundary dimension.

3.5 Create path for each edge

```
for k = 1:length(B)
```

```
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'LineWidth', 3)
end
```

we used the for loop to draw each boundry from begining to end



3.6 Convert to 3D coordinate

To conver to 3D coordinates, we use the for loop to set the coordinates inside B2 matrix:

```
for i = 1:length(B)
    b = B{i};
    bx = -b(:,2)*delta*ScaleValue;
    by = b(:,1)*delta*ScaleValue;
    bz = zeros(length(bx),1);
    B2{i} = [bx by bz];    %B2 is a matrix, i loop over and set each bx,by,bz of the matrix
    %plot3(bx,by,bz); hold on;
end
```

Here b is a matrix we had earlier and b(:,1) is the all elements of first column of matrix b.

3.7 Find the center point of the graph, then shift to 0,0 then shift to our center point

```
XYZmatrix = cell2mat(B2);
```

if we don't convert then we need write using writecell and the output is a mess, therefore we convert it to matrix then export, the result is much cleaner

```
XYZmatrixRowSize=size(XYZmatrix,1);
```

we get matrix's first dimension (row) size used later for for loop

```
SumOfEachColumn=sum(XYZmatrix,1);
```

S = sum(A,dim) returns the sum along dimension dim. For example, if A is a matrix, then sum(A,1) is a column vector containing the sum of each column

```
OldCenterPointX=SumOfEachColumn(1)/size(XYZmatrix,1);
```

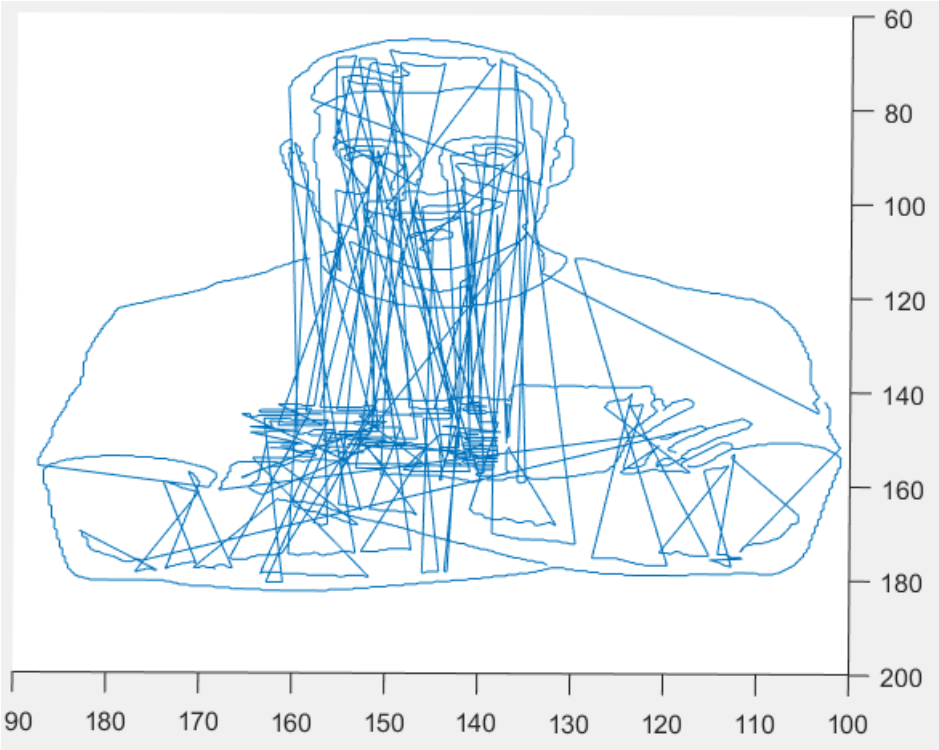
```
OldCenterPointY=SumOfEachColumn(2)/size(XYZmatrix,1);
```

`size(XYZmatrix,1)` is number of rows of XYZ matrix, first column is sum of x values, second column is sum of y values

```
for iii = 1:XYZmatrixRowSize
    XYZmatrix(iii,1)=XYZmatrix(iii,1)-OldCenterPointX+origin(1);
    XYZmatrix(iii,2)=XYZmatrix(iii,2)-OldCenterPointY+origin(2);
end
```

```
plot3(XYZmatrix(:,1),XYZmatrix(:,2),XYZmatrix(:,3));
```

we loop over the matrix to change the points to the shifted position with new origion



3.8 Edit z axis if we start draw new line (like human drawing not connected lines)

previousX=0; previousY=0;

after we store bx and by, we check it with new bx and by value, first time set to 0

for ii = 1:XYZmatrixRowSize

ThisX =XYZmatrix(ii,1);

extract the first folumn (which is x) of every XYZmatrix using for loop

ThisY=XYZmatrix(ii,2);

extract the second folumn (which is y) of every XYZmatrix using for loop

if abs(ThisX-previousX)>DistanceWhichLiftPen || abs(ThisY-previousY)>DistanceWhichLiftPen

if the move distence is too much between two points then lift pen to move robot arm there

XYZmatrix(ii,3)=ZOffset+LiftPenHeight;

we change the z value here for how high the pen lift

else

```
    XYZmatrix(ii,3)=ZOffset+0;
  end
```

no lifting means drawing.

```
   previousX=ThisX;
   previousY=ThisY;
end
```

after we store bx and by, we set it as new previous value, which will be compared when run for loop again

3.9 For yaw (w) roll (P) and pitch (R) and output to csv file

```
WPRMatrix(1:XYZmatrixRowSize,1) = Yaw;
```

make a number of row same as XYZmatrix and first column equals yaw.

```
WPRMatrix(1:XYZmatrixRowSize,2) = Roll;
```

second column equals pitch.

```
WPRMatrix(1:XYZmatrixRowSize,3) = Pitch;
```

third column equals Pitch.

```
XYZWPRMatrix = [XYZmatrix WPRMatrix];
```

combine x,y,z and yaw pitch and roll to one sum matrix

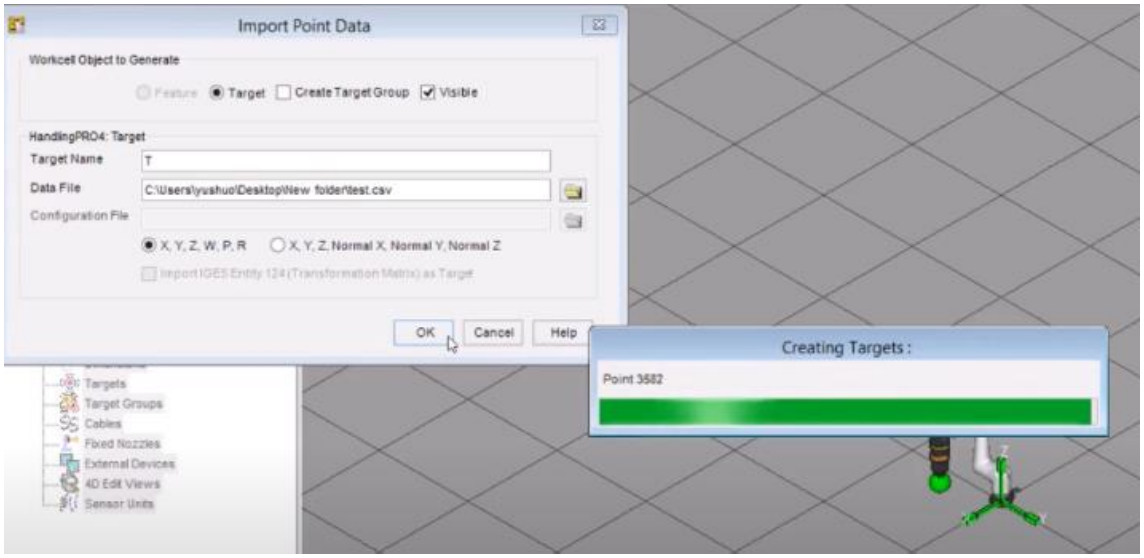
finally utput to csv file:

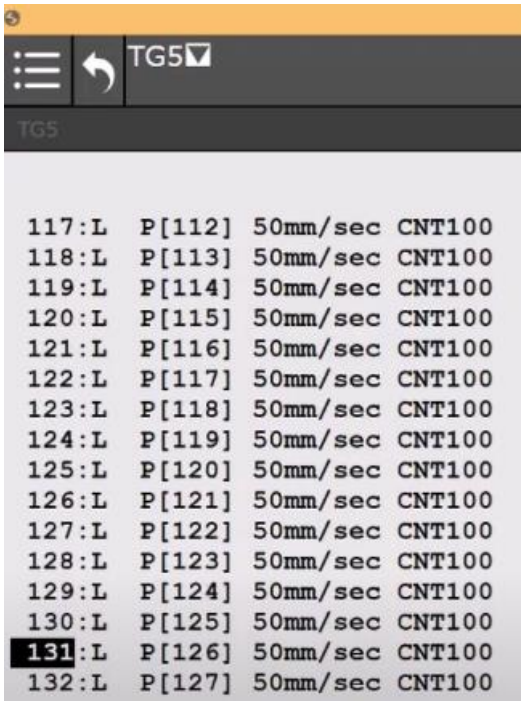
```
writematrix(XYZWPRMatrix,'test.csv');
```

	A	B	C	D	E	F
1	112.1957	190.9094	123.215	-176.629	-2.447	112.57
2	112.4957	190.6094	113.215	-176.629	-2.447	112.57
3	112.4957	190.3094	113.215	-176.629	-2.447	112.57
4	112.4957	190.0094	113.215	-176.629	-2.447	112.57
5	112.4957	189.7094	113.215	-176.629	-2.447	112.57
6	112.4957	189.4094	113.215	-176.629	-2.447	112.57
7	112.7957	189.1094	113.215	-176.629	-2.447	112.57
8	112.7957	188.8094	113.215	-176.629	-2.447	112.57
9	112.7957	188.5094	113.215	-176.629	-2.447	112.57
10	112.7957	188.2094	113.215	-176.629	-2.447	112.57
11	112.7957	187.9094	113.215	-176.629	-2.447	112.57

3.10 Export to Roboguide

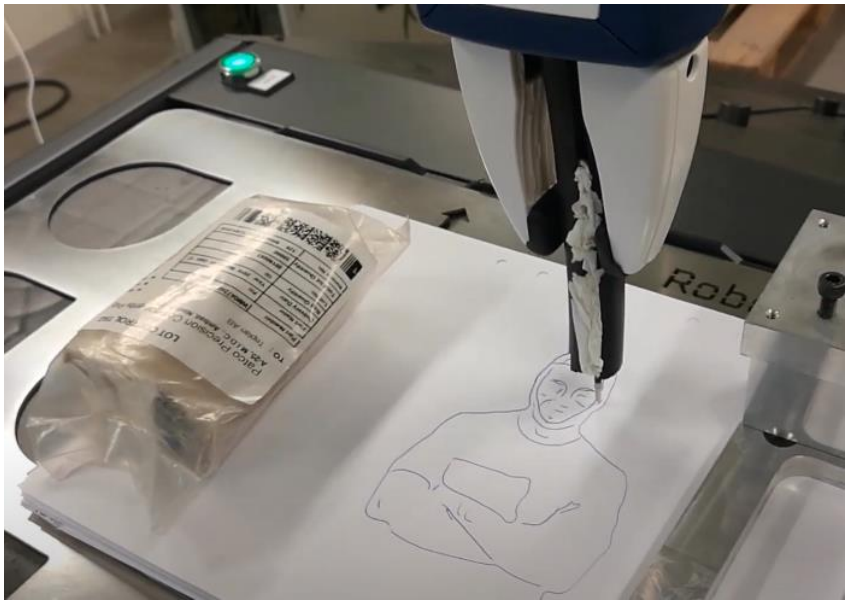
The teaching points generated and saved in the previous step needed to be input as Targets of the work cell and were then introduced into a target group as members to generate TP programs then exported to USB finally to the robot arm.





3.11 Robot arm rita

After tranfer the generated program, the robot arm will draw if we start the program.



3. Analys och slutsatser

References

Kikawada, T. (2022). Portrait Drawing using Computer Vision and Robot Manipulator (https://www.mathworks.com/matlabcentral/fileexchange/67926-portrait-drawing-using-computer-vision-and-robot-manipulator), MATLAB Central File Exchange. Retrieved June 20, 2022

Li, M. (2017). Using Targets in Roboguide to Visualize Path Planning of a FANUC Robot Retrieved from https://sites.asee.org/edgd/wp-content/uploads/sites/22/2017/12/Part25-Li.pdf

FANUC. (2016). ROBOGUIDE - FANUC Simulation Software. Retrieved from http://robot.fanucamerica.com/products/vision-software/ROBOGUIDE-simulation-software.aspx

Pan, Z., Polden, J., Larkin, N., Van Duin, S., & Norrish, J. (2012). Recent progress on programming methods for industrial robots. Robotics and Computer-Integrated Manufacturing, 28(2), 87-94.

Sheng, W., Xi, N., Song, M., Chen, Y., & MacNeille, P. (2000). Automated CAD-guided robot path planning for spray painting of compound surfaces. Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on (Vol. 3, pp. 1918-1923). IEEE.