Machine vision camera calibration and robot communication

Per Stark
Summary

This thesis is a part of a larger project included in the European project, AFFIX. The reason for the project is to try to develop a new method to assemble an aircraft engine part so that the weight and manufacturing costs are reduced. The proposal is to weld sheet metal parts instead of using cast parts. A machine vision system is suggested to be used in order to detect the joints for the weld assembly operation of the sheet metal.

The final system aims to locate a hidden curve on an object. The coordinates for the curve are calculated by the machine vision system and sent to a robot. The robot should create and follow a path by using the coordinates. The accuracy for locating the curve to perform an approved weld joint must be within +/- 0.5 mm.

This report investigates the accuracy of the camera calibration and the positioning of the robot. It also brushes the importance of good lightning when obtaining images for a vision system and the development for a robot program that receives these coordinates and transform them into robot movements are included.

The camera calibration is done in a toolbox for MatLab and it extracts the intrinsic camera parameters such as the distance between the centre of the lens and the optical detector in the camera: \( f \), lens distortion parameters and principle point. It also returns the location of the camera and orientation at each obtained image during the calibration, the extrinsic parameters. The intrinsic parameters are used when translating between image coordinates and camera coordinates and the extrinsic parameters are used when translating between camera coordinates and world coordinates.

The results of this project are a transformation matrix that translates the robots position into the cameras position. It also contains a robot program that can receive a large number of coordinates, store them and create a path to move along for the weld application.
# Table of Contents

Summary..............................................................................................................................................i  
1 Introduction.......................................................................................................................................1  
2 Method / Approach .........................................................................................................................3  
  2.1 Detail Description....................................................................................................................3  
3 Robot Repeatability Test..............................................................................................................4  
4 Camera Calibration .........................................................................................................................6  
  4.1 Camera Calibration Procedure.............................................................................................10  
  4.2 Camera Calibration Results Narrow View Angle Lens system........................................12  
  4.3 Camera Calibration Results Wide View Angle Lens system.............................................14  
5 Illumination......................................................................................................................................15  
6 Robot Communication.....................................................................................................................18  
7 Results...........................................................................................................................................20  
8 Conclusions....................................................................................................................................24  
  8.1 Recommendations for Further Work....................................................................................25  
9 References.......................................................................................................................................26  

## Appendices

A. Flow Chart LabVIEW Program  
B. Flow Chart Robot Program  
C. Robot Program, RAPID
1 Introduction

The purpose for this project is to find a solution for the welding of the aircraft component shown in Figure 1.1. This project is a part in a larger European project called AFFIX, a collaboration between universities and companies in Europe. One part of AFFIX aims to reduce weight and manufacturing costs for the aerospace component shown in Figure 1.2. This component is a part in the combustion chamber within an aircraft engine.

![Welded part in Affix project](image)

Today both the inner ring and the outer ring, of the combustion chamber, are cast and then joined together by their veins. The cast parts are today very expensive and heavy and this has to be improved. The proposal is to manufacture the component by assembly smaller parts, see Figure 1.1, together. These smaller sectors are to be built up by five different titanium parts shown to the right in Figure 1.1. Part 1 is cast and parts 2 to 5 are titanium sheets.
The assembly of these five components are as follows: components 2 to 5, shown in Figure 1.1, are spot welded together and then clamped in a fixture with component 1. During this process some deformations and movements along the three axes could occur. A robotized laser weld application finally seam weld together the 5 parts. The problem here is the red marked joints in Figure 1.1. Those joints are hidden and the welding should be performed through the gable of the component. To find those joints a machine vision system are to be used. This thesis comprises an evaluation of the accuracy of the robot and camera calibration that are to be used in the vision system. Included is also the programming of the robot system and communication as well as a transformation matrix that translates the robot tool coordinate system into the camera coordinate system.
2 Method / Approach

The first thing that was done was to evaluate the robot's repeatability: This was done by mounting a laser pointer on to the robot hand and checks how accurately the robot can hit a reference point after big reorientations. The reason is to assure that the robot hits the same point from time to time when obtaining the images for the machine vision program. The next step was to introduce the camera into the robot's coordinate system and calibrate the camera to receive the camera's intrinsic and extrinsic parameters along with the robot positions. This was by defining the camera calibration toolbox and the robot's work object in the same coordinate system. The reason for this is to simplify the calculations of the transformation matrix that translates the camera pose into the robot pose. The third step was to use the extrinsic camera parameters received from the MatLab camera calibration toolbox to calculate the transformation matrix. And the last step was to develop a communication program for the robot for receiving the coordinates of the object. When the finished machine vision program calculates the coordinates for the weld path the robot program must be able to receive and sign for the coordinates. It also has to be able to generate a path to move along. To evaluate the robot program a test program in LabVIEW is created.

2.1 Detail Description

In this project an ABB robot of type IRB2400 with a S4 control system was used to carry the hand mounted camera. The camera calibrated was a CCD analogue and two different lenses were used to evaluate different results. The frame grabber card was a DT 3120 and the software's LabVIEW and MatLab was used. MatLab was used for the camera calibration with the camera calibration toolbox and LabVIEW was used to create a test program for the robot communication.
3 Robot Repeatability Test

A test was performed to test the robot's repeatability considering the angular precision. The robot used in the test was an ABB IRB-2400 with the S4 control system. The robot's repeatability is stated to be within 0.06 mm but nothing is told about the angular precision [1]. The angular repeatability is important when positioning the camera and for the calculations to transform the robot coordinate system into the camera coordinate system. The test was to use a laser pointer mounted on the robot arm pointing on a wall and see how accurately the pointer could target the same spot from time to time. If the distance between the wall and the robot is known, the error between the reference point and the repeated points can be measured and the angular error calculated and evaluated. The first step was to mount a laser marker on the robot arm, see Figure 3.1.

![Figure 3.1: Laser pointer mounted on the robot](image)

The next step was to create a reference point on the wall. The distance $L$ between the robot base frame and the wall was measured to $L = 30$ meters, see Figure 3.2. After the point was selected, it was stored as a robot pose. After creating the pose, two more poses were created with a large reorientation relative to each other. This was done to try to create as large of an error as possible when returning to the reference point. The program with the three different poses was then repeated a number of times to investigate how close to the reference point the laser marker would come each time. The result of the test was that the error was undetectable with a human eye and because of the long distance $L$ between the robot and the reference point, the angular error can be neglected. The conclusion is based on the fact that the application should work with much shorter distance than $L$ for the camera to the object and therefore the error would be even smaller.
Figure 3.2: Illustration of repeatability test in RobCad
4 Camera Calibration

For the camera calibration a toolbox in MatLab was used [2]. The toolbox works with images obtained by a camera and loaded into the program as files. For the camera calibration 20 different images were obtained from different locations and orientations when the camera was mounted onto the robot's hand. The camera's $f$ (the distance between the centre of the lens and the camera's optical chip) had to be kept intact at all time. After the work object for the robot was defined, Figure 4.1 the images obtained by the robot hand mounted camera were loaded into the MatLab camera calibration toolbox. The coordinate system for the camera were placed along the same sides so that the two coordinate systems were aligned, see Figure 4.2.

The calibration of the camera is described in detail in chapter 4.1. When satisfied with calibrating the images a file with the calibration results can be opened. This file contains the intrinsic and extrinsic camera parameters. The intrinsic camera parameter contains the $f$, principle point, distortion parameters. The extrinsic camera parameters are represented in a 3x3 rotation matrix, $R$, and a 1x3 translation vector, $T$. These can be used to transform from reference coordinates to camera coordinates with the relation [2]:

$$X = RX + T$$  \hspace{1cm} (4.1)

$XX$ is a coordinate vector expressed in the reference coordinate system shown in Figure 4.2 and $XX_c$ is a coordinate vector expressed in the camera reference frame. $R$ is the rotation matrix and $T$ is the translation vector given by the camera calibration toolbox. The rotation matrix is the rotation around the three axes. The translation vector is the translation along the three axes and is given in millimetres. The equation can be summed into a 4x4 matrix that contains both the rotation matrix and translation vector [3]:
Machine vision camera calibration and robot communication

\[
\begin{bmatrix}
\beta_1 & \beta_2 & \beta_3 & T_x \\
\beta_4 & \beta_5 & \beta_6 & T_y \\
\beta_7 & \beta_8 & \beta_1 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
\beta_1 & \beta_2 & \beta_3 & T_x \\
\beta_4 & \beta_5 & \beta_6 & T_y \\
\beta_7 & \beta_8 & \beta_9 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix}^{-1}
\] (4.2),(4.3)

where \( ^{c}T_{r} \) and \( ^{r}T_{c} \) are the transformation matrices containing both the reorienation and the translation. \( \beta_1 \) to \( \beta_9 \) are components in the rotation matrix obtained by multiplying Eq. (4.4) – (4.6) together shown in Eq. (4.7). This shows the relation between \( \beta_i \) and the rotation angles. To get the complete \( ^{c}T_{r} \), Eq. (4.8), the translation, is multiplied. This matrix expresses where the origin of the reference coordinate system is in the cameras coordinate system and to get the camera coordinate system represented in the reference coordinate system the matrix is inverted. The robots pre defined tool, tool0's location in (x, y, z) in the reference coordinate system is given by its program’s log file. The orientation of tool0’s coordinate system expressed in the reference coordinate system is given in angles (degrees) rotated around (z, y, x). Tool0’s rotation matrix is given by first rotating around the z-axis, Eq. (4.4), then the y-axis, Eq. (4.5) and last the x-axis, Eq. (4.6) and its translation is given in mm along the three axis, Eq. (4.8) [3].

\[
\text{Rot}(z, \alpha_1) = \begin{bmatrix}
\cos(\alpha_1) & -\sin(\alpha_1) & 0 & 0 \\
\sin(\alpha_1) & \cos(\alpha_1) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (4.4)

\[
\text{Rot}(y, \alpha_2) = \begin{bmatrix}
\cos(\alpha_2) & 0 & \sin(\alpha_2) & 0 \\
0 & 1 & 0 & 0 \\
-\sin(\alpha_2) & 0 & \cos(\alpha_2) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (4.5)

\[
\text{Rot}(x, \alpha_3) = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(\alpha_3) & -\sin(\alpha_3) & 0 \\
0 & \sin(\alpha_3) & \cos(\alpha_3) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (4.6)
Machine vision camera calibration and robot communication

\[
\text{Rot}_{z,y,x} = \begin{bmatrix}
\cos(\alpha_1) \cos(\alpha_2) & \cos(\alpha_1) \sin(\alpha_2) \sin(\alpha_3) & -\sin(\alpha_1) \cos(\alpha_2) & \cos(\alpha_1) \sin(\alpha_2) \cos(\alpha_3) + \sin(\alpha_1) \sin(\alpha_3) \\
\sin(\alpha_1) \cos(\alpha_2) & \sin(\alpha_1) \sin(\alpha_2) \sin(\alpha_3) + \cos(\alpha_1) \cos(\alpha_3) & \cos(\alpha_1) \sin(\alpha_2) \cos(\alpha_3) - \cos(\alpha_1) \sin(\alpha_3) & 0 \\
0 & \cos(\alpha_2) \sin(\alpha_3) & \cos(\alpha_2) \cos(\alpha_3) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(4.7)

\[
\text{Trans}(x, y, z) = \begin{bmatrix}
1 & 0 & 0 & T_x \\
0 & 1 & 0 & T_y \\
0 & 0 & 1 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(4.8)

Tool0's coordinate system represented in the reference coordinate system can be calculated by equation (4.9) and represented by a 4x4 matrix:

\[
'^r T_l = \text{Trans}(x, y, z) \cdot \text{Rot}(x, \alpha_3) \cdot \text{Rot}(y, \alpha_2) \cdot \text{Rot}(z, \alpha_1)
\]

(4.9)

and

\[
'^r T_r = 'r T_l^{-1}
\]

(4.10)

This can be used to translate the tool0’s coordinates into the reference coordinate system (4.11)

\[
XX_r = '^r T_r \cdot XX_l
\]

(4.11)

Where \(XX_l\) is a coordinate vector expressed in the tool0’s coordinate system and \(XX_r\) is a coordinate vector expressed in the reference coordinate system. Given the tool0’s and the cameras positions in the reference coordinate system a transformation matrix that translates tool0’s coordinates to the cameras coordinates can be calculated, see Figure 4.3 and Eq. (4.12).

\[
'^r T_c = '^r T_r \cdot '^r T_c
\]

(4.12)

Given the transformation matrices \('^r T_r\) and \('^r T_c\) from Eq. (4.3) and (4.10).
Figure 4.3: Tool0's and cameras coordinate system represented in the reference coordinate system
4.1 Camera Calibration Procedure

The first step in the camera calibration procedure was to load the images into the camera calibration toolbox. This was done by obtaining twenty images from different positions and orientations shown in Figure 4.4.

![Figure 4.4: Calibration images](image)

When the images are loaded a corner extraction function is used. This function lets you define the origin of the reference coordinate system and the orientation of the x-axis and the y-axis. It also asks for the number of squares along the two axes and the dX and dY, shown in Figure 4.5. When these constants are defined the toolbox projects an undistorted pattern of red crosses onto the images. The user then has to decide if an initial guess for lens distortion must be done. This is necessary if the locations of the red crosses are inconsistent from the original squared pattern. When satisfied with consistency between the original pattern and the projected pattern the next image is loaded and the procedure is repeated.

![Figure 4.5: Projected pattern in the form of red crosses and the constants dX and dY](image)
When all twenty images are defined the calibration can begin. The function returns the cameras intrinsic and extrinsic parameters as mentioned before. It also returns a chart of the pixel error between the projected pattern and the original pattern, see Figure 4.6.

![Figure 4.6: Pixel error between projected pattern and original pattern in all corners](image)

Each cross in Figure 4.6 represents the pixel error in an image between one corner of the original pattern and projected one. This chart can be used to improve images with large pixel errors by reloading them and repeat the steps with initial guess of lens distortion. If the image still shows large errors it can be excluded from the camera calibration. When satisfied with the chart in Figure 4.6 the calibration is finished and the calibration result file can be opened. This file contains all the data of the camera calibration. This procedure was performed for two different kinds of lenses to evaluate which one that gave the best result.
4.2 Camera Calibration Results Narrow View Angle Lens system

The first camera calibration was with a lens system with a focal length of 6-60 mm and a horizontal view angle of 6-65 degrees. This lens system has the benefits of low lens distortion and the drawback of quite long focal length. Twenty images from different positions and orientations were obtained shown in Figure 4.4. The first ten images were loaded in to MatLab calibration toolbox to calibrate the camera. The intrinsic parameters are presented below:

Distance between the centre of the lens to the optical detector in the camera, given in scaled values pixel width and pixel height:

\[ f = [1235.922 \ 1220.267] \]

Principal point. The location of the optical axes in the image plan given in scaled pixel width and pixel height:

\[ cc = [349.533 \ 334.654] \]

Radial distortion coefficients, coefficients to compensate for the distortion caused by the lens convexity:

\[ \kappa = [-0.371 \ 0.398 \ -0.002 \ 0.004] \]

were \( f \) and \( cc \) are measured in pixels and the distortion is the lens radial distortion defined in [2] and shown below:

If A be a point with the coordinate vector \( XX_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \) in the camera coordinate system.

then \( x_p \) is a pinhole image projection:

\[ x_p = \begin{bmatrix} X_c \\ Z_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \]

Let then \( r^2 = x^2 + y^2 \) and after including lens distortion the new normalized point

\[ x_d = \begin{bmatrix} x_{d_1} \\ x_{d_2} \end{bmatrix} = (1 + \kappa_1 \cdot r^2 + \kappa_2 \cdot r^4 + \kappa_3 \cdot r^6) \cdot x_p + dx \]

where \( dx \) is the tangential distortion vector not discussed in the report. After the lens distortion is applied the pixel coordinates \( x_p \) and \( y_p \) are:

\[ x_p = f \cdot \left( x_{d_1} + \alpha_{c} \cdot x_{d_2} \right) + cc_1 \]

\[ y_p = f_2 \cdot x_{d_2} + cc_2 \]

where \( \alpha_{c} \) is the skew coefficient not discussed in this report.
The extrinsic parameters are recalculated so that they represent the camera in the reference coordinate system according to Eq. (4.3). Tool0’s location and orientation are recalculated according Eq. (4.9) and the transformation matrix \( T_c \), from tool0 to camera can be calculated according to Eq. (4.12). The matrix contains both the translation in millimetres and the reorientation expressed in cosine and sine components. The \( T_c \) are calculated for the ten first robot positions and camera locations to acquire an average transformation matrix shown below:

\[
t_{Tc\_\text{average}} = \frac{\left( T_{c_1} + T_{c_2} + T_{c_3} + T_{c_4} + T_{c_5} + T_{c_6} + T_{c_7} + T_{c_8} + T_{c_9} + T_{c_{10}} \right)}{10}
\]

\[
t_{Tc\_\text{average}} = \begin{bmatrix}
-0.0089 & 0.9999 & -0.0045 & 88.2172 \\
-0.9992 & -0.0087 & 0.0385 & 1.1613 \\
0.0385 & 0.0049 & 0.9992 & 235.2393 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

To evaluate the results and check the accuracy of the transformation matrix ten more images were used. This was done by comparing the calculated camera position, obtained by multiplying the known robot position with \( T_{c\_\text{average}} \), with the calibrated camera position, obtained by the MatLab camera calibration toolbox. The cameras position was recalculated to be represented in the reference coordinate system, Eq. (4.3), and the calculated camera position was obtained by multiplying the known robot position with the \( T_{c\_\text{average}} \). The absolute error for each comparison is used to calculate the average and maximum error between calculated and the calibrated location. This was done for the ten last images and the results are shown below: The rotation is expressed in cosine and sine components and the translation in millimetres.

\[
\text{average\_error} = \begin{bmatrix}
0.0011 & 0.0002 & 0.0025 & 1.2152 \\
0.0003 & 0.001 & 0.0023 & 1.0659 \\
0.0025 & 0.0022 & 0.0006 & 0.6413 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{max\_error} = \begin{bmatrix}
0.0018 & 0.0005 & 0.0052 & 2.6332 \\
0.0006 & 0.0018 & 0.0039 & 1.9181 \\
0.0043 & 0.0049 & 0.0011 & 1.2989 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
4.3 Camera Calibration Results Wide View Angle Lens system

The second camera calibration was done with a lens system with shorter focal length, 3.5-8 mm and a horizontal view angle of 44-95 degrees. This lens has the benefits to be closer to the object but the drawback of larger lens distortion. Twenty new images were obtained in similar locations as previous camera calibration and loaded in to the calibration toolbox. The same calculations were performed as for previous lens and the intrinsic parameters are shown below:

Distance between the centre of the lens to the optical detector in the camera, given in scaled values pixel width and pixel height:
\[ f = \begin{bmatrix} 1007.120 \\ 994.243 \end{bmatrix} \]

Principal point. The location of the optical axes in the image plan given in scaled pixel width and pixel height:
\[ c_{\text{c}} = \begin{bmatrix} 322.752 \\ 304.609 \end{bmatrix} \]

Distortion coefficients, coefficients to compensate for the distortion caused by the lens convexity:
\[ \kappa = \begin{bmatrix} -0.379 \\ 0.188 \\ -0.002 \end{bmatrix} \]

The extrinsic parameters are recalculated as before and a new transformation matrix, \( t_{Tc_{\text{average}}} \) is obtained, see below.

\[
t_{Tc_{\text{average}}} = \begin{bmatrix}
-0.0088 & 0.9999 & -0.0112 & 88.3381 \\
-0.9993 & -0.0084 & 0.0358 & 0.3388 \\
0.0357 & 0.0115 & 0.9993 & 209.5504 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The same procedure to check the accuracy of the transformation matrix was repeated. The results are shown below.

\[
\text{average}_\text{error} = \begin{bmatrix}
0.0010 & 0.0002 & 0.0021 & 0.8904 \\
0.0003 & 0.0011 & 0.0021 & 0.7876 \\
0.0023 & 0.0018 & 0.0005 & 0.5623 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{max}_\text{error} = \begin{bmatrix}
0.0018 & 0.0006 & 0.0034 & 1.6896 \\
0.0006 & 0.0021 & 0.0038 & 1.6053 \\
0.0044 & 0.0028 & 0.0012 & 1.5061 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
5 Illumination

Illumination is an important factor in machine vision applications. Once you obtain an image there are various options to improve contrast using filters. But if the image obtained has not a good enough contrast to start with there are little you can do to find edges and patterns. One way to improve your contrast is to illuminate your object in different ways. Here follow an example using National Instruments Vision Development Module to try to locate different coordinates with help of an edge detection function [4]. The images were obtained using a single CCD camera and a Data Translation frame grabber card model DT 3120. The first image, seen in Figure 5.1, was captured using illumination from the same point of view as the camera (the right side of the image). The second image, seen in Figure 5.2 was captured by using illumination from two sides. The two light sources were one from the cameras point of view and one from the opposite side (the left side of the image).

As seen in the two images the contrast between the two sheets of metal is better in Figure 5.2 then in Figure 5.1. This is very useful when trying to find edges in an image of objects with similar colours. After the images were obtained using the CCD camera they were imported into Nation Instruments Vision Development Module. The images were transferred into greyscale. The contrast and the brightness were adjusted using the built in functions in the Vision Module. An edge detection function were then applied to try to locate coordinates along the red marked contour shown in Figure 5.3
Machine vision camera calibration and robot communication

The edge detection function was applied to both images seen in Figure 5.1 and 5.2 to illustrate the importance to have a good image with good contrast to start with. Seen in Figure 5.4 is the first image captured with bad contrast to start with. The contrast has been improved but not satisfactory.

The green arrows search along them selves for a contrast difference defined by the user, a threshold value between 0 and 255 for greyscale images. When the function finds a big enough difference it sends the coordinates in y-axis and x-axis back in pixels, marked as blue boxes in the image. Twelve searches were done on the image and the function only returned two
coordinates. When the edge detection filter were applied on the image shown in Figure 5.2, with the better contrast the result improved drastic, shown in Figure 5.5.

As seen in Figure 5.5 the edge detection function returns twelve hits, marked with the blue boxes. Each hit returns two values, the x-coordinate and the y-coordinate. The default value is in pixels but the image can be calibrated so that the function returns the distances in for example mm or inches. The reference coordinates system can also be user defined and are by default in the lower left corner of the image.
6 Robot Communication

The coordinates for the curve to be welded are calculated by the vision program. These coordinates are then sent to the robot via the serial port. The robot program has to be able to receive an undefined number of coordinates along the curve and therefore some sort of protocol has to be set up for starting and ending the transmission. The coordinates received are the translation in X, Y and Z direction to a defined origin for the robot. The configuration for the serial port was set up as following:

- Baud rate: 9600
- Parity: None
- Flow control: None
- Number of bits: 8
- Number of stop bits: 1
- Termination char: Linefeed (\n)
- Timeout: 10000 ms

To test the robot programs functionality a test program in LabVIEW was created [6]. This program were able to send an infinite number of coordinates of own choice until a stop word was sent. The program starts with sending the word start then it sends the X-coordinate, Y-coordinate and the Z-coordinate. After each send coordinate LabVIEW waits for the robot program to sign it and then sends the next one. This can be repeated until the word stop is send by pressing the move button. See flow chart in appendix A. As seen if button read is pressed the program reads the X-coordinate filled in on the front panel shown in Figure 7.1, then it opens the serial channel and sends it to the robot, it waits for an OK from the robot and then sends the Y-coordinate. This is repeated until all coordinates are sent. If needed, the user can then define a new set of coordinates and repeat the procedure.

![Figure 6.1: Front panel of LabVIEW test program](image-url)
The robot program, developed with help of the rapid reference manual [5], had to be designed so that it could receive an unknown number of locations because the number of detected locations along the curve in the image is unknown in advance. This was solved by creating three coordinate vectors, one for X one for Y and one for Z locations. These vectors are filled simultaneously for each set of coordinates and are when all locations received available for the robot programs MOVE function. The robot program functionality is explained below, the flow chart is found in appendix B and the robot program in appendix C.

1. The robots serial channel is open for reading and writing
2. The program reads the serial channel and makes a decision. Start reading coordinates or sending the coordinates to the MOVE PROC. If nothing is found at the serial port an error routine is called and the program starts over and reads the serial port again
3. If the string “start” is found at the serial port the program starts to read coordinates sent. Starting with the X-coordinate, converting it into a number and saving it in the coordinate vector’s first position. If the instructions succeed the robot returns the string “OK” to the computer and repeats the same procedure with the Y and Z-coordinate of the location. If the instructions fail the program jumps to an error routine and restarts the program.
4. When received all three coordinates for the location the program increments the coordinate vector for the next set of coordinates and jumps to the function seen in step 2
5. If the string “stop” is found on the serial port in step 2 the program jumps to the move routine. This routine extracts the locations from the coordinate vectors and move’s the robot to the defined location starting with the first. This procedure is repeated for as many times as the coordinated vector has been incremented. From 1 time up till a 1000 which is the maximum number of the coordinates the program can receive. After moving to the last location the program resets the coordinate vectors and jumps to step 2.
7 Results

The accuracy of the system mainly depends on the robot’s absolute accuracy and the resolution and calibration results of the camera. The absolute accuracy of the robot may vary from position to position. A test to measure the angular repeatability of the robot was performed to make sure that the robot’s repositioning ability was accurate enough with respect to the angular positioning of the robot arm. The positional repeatability of the robot was stated to be within 0,06 mm [1]. A test with a laser pointer was set up to check how accurate the robot could target a laser point on a wall at the same position after repeating a loop of other positions. It was discovered that the error in the repeatability was undetectable and therefore can be set to zero, more about the test can be read in chapter 3. The next parameter to evaluate was the accuracy of the camera calibration. This was performed by first calibrating the camera so that a number of positions for the camera were known. The robot position at all locations was saved in a log file and after the calibration a transformation matrix to translate the robot’s tool0 position to the camera position was calculated. Then images not used to calculate the transformation matrix and the robot position for those locations was used to evaluate the accuracy of the system. Two different lenses were used in the calibration, one wide view angel lens system (horizontal view angle of 44-95 degrees) and one narrow view angel lens system (a horizontal view angle of 6-65 degrees) to investigate if low lens distortion gives a better result then being closer to the object. The first calibration was made with the narrow field of view lens system and an $f$ of 6-60mm. The results from the camera calibration can be found in chapter 4 and below is the average transformation matrix, $tTc\_average$, calculated from ten camera and robot locations. The average error, $average\_error$, is the result of a comparison between the calculated camera position using the robot position and the transformation matrix and the calibrated camera position obtained by the camera calibration toolbox in MatLab. The average error was calculated from ten comparisons. The maximum error, $max\_error$, is the largest error among all ten images from the same comparison. Both error matrices are the absolute value of the error.

$$tTc\_average = \begin{bmatrix} -0.0089 & 0.9999 & -0.0045 & 88.2172 \\ -0.9992 & -0.0087 & 0.0385 & 1.1613 \\ 0.0385 & 0.0049 & 0.9992 & 235.2393 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Which gives $\alpha_1\_tTc\_average = -89,38^\circ$ and $T_x\_tTc\_average = 88.22\text{mm}$, $\alpha_2\_tTc\_average = -2,21^\circ$ and $T_y\_tTc\_average = 1,16\text{mm}$, $\alpha_3\_tTc\_average = 0,28^\circ$ and $T_z\_tTc\_average = 235,24\text{mm}$
average_error =
\[
\begin{bmatrix}
0.0011 & 0.0002 & 0.0025 & 1.2152 \\
0.0003 & 0.001 & 0.0023 & 1.0659 \\
0.0025 & 0.0022 & 0.0006 & 0.6413 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[\alpha_1_{\text{average_error}} = 0.017^\circ\]
\[\alpha_2_{\text{average_error}} = -0.14^\circ\]
\[\alpha_3_{\text{average_error}} = 0.13^\circ\]
\[T_x_{\text{average_error}} = 1.22\text{mm}\]

and
\[T_y_{\text{average_error}} = 1.07\text{mm}\]
\[T_z_{\text{average_error}} = 0.64\text{mm}\]

max_error =
\[
\begin{bmatrix}
0.0018 & 0.0005 & 0.0052 & 2.6332 \\
0.0006 & 0.0018 & 0.0039 & 1.9181 \\
0.0043 & 0.0049 & 0.0011 & 1.2989 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[\alpha_1_{\text{max_error}} = 0.034^\circ\]
\[\alpha_2_{\text{max_error}} = -0.25^\circ\]
\[\alpha_3_{\text{max_error}} = 0.28^\circ\]
\[T_x_{\text{max_error}} = 2.63\text{mm}\]

and
\[T_y_{\text{max_error}} = 1.92\text{mm}\]
\[T_z_{\text{max_error}} = 1.3\text{mm}\]
The next lens system used in the calibration was with a larger field of view and an *f* of 3.5-8mm. The same calculations and the same method to evaluate the accuracy were made. Below are the accuracy results and the camera parameters are stated in chapter 4.2. The average transformation matrix, the average error and the maximum error are shown below. Both error matrices are the absolute value of the error.

\[
\begin{bmatrix}
-0.0088 & 0.9999 & -0.0112 & 88.3381 \\
-0.9993 & -0.0084 & 0.0358 & 0.3388 \\
0.0357 & 0.0115 & 0.9999 & 209.5504 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[tTc\_average = \begin{bmatrix}
\alpha_1\_tTc\_average = -89.36^\circ \\
\alpha_2\_tTc\_average = -2.05^\circ \\
\alpha_3\_tTc\_average = 0.66^\circ \\
\end{bmatrix}
\]

\[T_x\_tTc\_average = 88.34\text{mm} \quad T_y\_tTc\_average = 0.34\text{mm} \quad T_z\_tTc\_average = 209.55\text{mm}\]

\[
\begin{bmatrix}
0.0010 & 0.0002 & 0.0021 & 0.8904 \\
0.0003 & 0.0011 & 0.0021 & 0.7876 \\
0.0023 & 0.0018 & 0.0005 & 0.5623 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[\begin{bmatrix}
\alpha_1\_average\_error = 0.017^\circ \\
\alpha_2\_average\_error = -0.13^\circ \\
\alpha_3\_average\_error = 0.10^\circ \\
\end{bmatrix}
\]

\[T_x\_average\_error = 0.89\text{mm} \quad T_y\_average\_error = 0.79\text{mm} \quad T_z\_average\_error = 0.56\text{mm}\]

\[
\begin{bmatrix}
0.0018 & 0.0006 & 0.0034 & 1.6896 \\
0.0006 & 0.0021 & 0.0038 & 1.6053 \\
0.0044 & 0.0028 & 0.0012 & 1.5061 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[max\_error = \begin{bmatrix}
\end{bmatrix}\]
Machine vision camera calibration and robot communication

\[ \alpha_1 \text{ max\_error} = 0,034^\circ \]

which gives

\[ \alpha_2 \text{ max\_error} = -0,25^\circ \]

\[ \alpha_3 \text{ max\_error} = 0,16^\circ \]

\[ T_z \text{ max\_error} = 1,69\text{mm} \]

and

\[ T_y \text{ max\_error} = 1,61\text{mm} \]

\[ T_z \text{ max\_error} = 1,51\text{mm} \]

Each \( \bar{T}_{c\text{\_average}} \) is calculated from cosine and sine components in \( T_{c_1} - T_{c_{10}} \). This was done because the MatLab camera calibration toolbox presented the results in cosine and sine components and it was easier to recalculate the robot positions into components then the calibration results into angels. The drawback of doing the calculations this way is the risk of an overrepresented system with a small error for each \( T_c \) calculation. The final result, \( rT_{c\_average} \) then shows a bigger error. A more correct way to do the calculation can be to recalculate the results from the camera calibration into angles and then calculate \( T_{c_1} - T_{c_{10}} \). This may give a more accurate \( \bar{T}_{c\_average} \) and smaller \( \text{average_error} \) and \( \text{max_error} \) matrices.

The results for the illumination evaluation show that it is very important with good lightning to obtain good contrast in the images. When trying to find edges and patterns in images different filters are used. For example a sobel filter is used to try to detect an edge; the filter looks for contrast changes in the pixel values of the image. The pixel values can be manipulated using different filters to adjust the contrast and brightness in the image but if the lightning is poor from the beginning the results will also be poor, shown in Figure 5.4. The same filters are used when detecting the edge in Figure 5.5 and the results are much better. Good lightning in a machine vision system is as important as the filters and algorithms used in the application.

The robot communication developed in this project was designed to receive an undefined number of translation coordinates in X, Y and Z direction. The robot program had to be able to receive, store and create a path from these coordinates. To test the robot program a test program was developed in LabVIEW. This program could send a number of user defined coordinates to the robot. The robot program was then tested with a number of different coordinate and work perfectly. The robot was also programmed with a interrupt sequence that could handle eventual interrupts in the communication. This function was designed to avoid the robot to stop if no data was found on the serial channel. This function also worked perfectly.
8 Conclusions

This thesis aims to calibrate a camera and to evaluate the accuracy of a camera calibration. When satisfied with the calibration a transformation matrix should be calculated to transform the robot tool coordinate system into the camera coordinate system. When this is done the camera location is known for every position the robot’s tool is known. The calibration and the calculations were performed for two different lenses to evaluate which one would give the best result. The lens system with the narrower field of view, lens 1, had the benefit of lower lens distortion but a greater distance to the object, the lens system with the wider field of view, lens 2, had the benefit of being closer to the object but with the drawback of bigger lens distortion. The results are presented in previous chapter and the conclusion is that lens 2 gave a better result in accuracy. It is still hard to determine if the camera system has a good enough accuracy to calculate the weld path. If the finished vision program calculates coordinates that are not exactly on the line for the weld path the robot will receive coordinates that are slightly off the target. More errors will then be added to the final result when the coordinates are sent to the robot for absolute positioning.

The conclusions of the test using different lightning sources when obtaining images is that with bad lightning you get bad contrast in the image and it is harder to find edges and shapes. The recommendation is to set up good and stable lightning before starting with the programming of filters and functions for your vision application.

The serial communication between the robot and LabVIEW created a lot of problem in the beginning. The reason for that was the COM2 port in the robot did not work. The configuration in the PC and the robot was the same but the signal got corrupt going from the PC to the robot. After error detection no error was found and I simply made a new cable to try the COM1 port. That port worked and after that no further problem accrued. The communication work perfectly with the limitation of just receiving the X, Y and Z coordinate. The program can not manage reorientations.
8.1 Recommendations for Further Work

Due to that this part in a bigger project the first recommendation is to create a program that extracts local pixel coordinates along the detected weld curve and translates them into world coordinates. This has to be done so that the robot can create a path to follow in the welding procedure and because the object may move from weld to weld. Thereafter the systems accuracy can be measured and evaluated. If the accuracy of the system is inadequate it has to be improved. Some suggestions how to improve the accuracy of the parameters in this report are stated below:

- Use an absolute calibrated robot to gain better positioning in camera calibration stage and movement along the curve.

- Try a camera with higher resolution and better optics. This may reduce the pixel errors in the calibration stage and therefore minimize the positioning uncertainty in the orientation matrix and translation vector.
9 References


A. Flow Chart LabVIEW Program

LabVIEW program flow chart. Program used to test and evaluate the robots communication and path generating program. Allowing the user to decide which coordinates (x, y, z) in millimetres and how many sets of coordinates that are going to be sent. The user stops the transmission by pressing the move button on the front panel.
B. Flow Chart Robot Program

Robot communication and path creating program. The program can receive an undefined number of locations. The locations are offsets from the defined reference coordinate system in the robot. It can receive up to 1000 coordinates and create a path from these coordinates starting with the first received and finish with the last. To receive a set of coordinates, the word “start” has to be found on the serial port. To stop the path generating and movements the word “stop” has to be found on the serial port.
C. Robot Program, RAPID

The rapid code for the robot communication and path generating.

%%%
VERSION:1
LANGUAGE:ENGLISH
%%%

MODULE SERIAL
PERS wobjdata wobj_cam:=[FALSE,TRUE,"",[[868.161,-137.22,698.186], 
[0.78105,0.000481,0.000525,0.624468]],[[0.0,0],[1.0,0,0]]];
CONST robtarget p10:=[[0.0,0],[0.059989,0.770207,0.633675,-0.040468],
[964.179,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR num always:=0;
VAR num x_coord:=0;
VAR num y_coord:=0;
VAR num z_coord:=0;
VAR num path_vector_x{1000};
VAR num path_vector_y{1000};
VAR num path_vector_z{1000};
VAR num counter_1:=1;
VAR num counter_2:=1;
VAR num error_check:=0;
VAR iodev com_ch_read;
VAR iodev com_ch_write;
VAR string recieved_string:="";
VAR bool ok;
PROC main()
WHILE always<1 DO \Infinite loop
  home: \Label for restarting the program
  Close com_ch_read; \Close serial channel for reading
  Close com_ch_write; \Close serial channel for writing
  Open "sio1",com_ch_read\Read; \Open serial channel for reading
  received_string:=ReadStr(com_ch_read); \Read serial channel for 60s and copy to recieved_string
  IF error_check=1 THEN \Checking for error
    error_check:=0;
    GOTO home;
  ENDIF
  Close com_ch_read; \Close serial channel for reading
  IF recieved_string="start" THEN \Checking if coordinates are to be recieved
    Open "sio1",com_ch_read\Read; \Open serial channel for reading
    received_string:=ReadStr(com_ch_read); \Read serial channel for 60s and copy to recieved_string
    IF error_check=1 THEN \Checking for error
      error_check:=0;
      GOTO home;
    ENDIF
    ok:=StrToVal(recieved_string,x_coord); \Convert string to number
    IF ok=TRUE THEN \Checking if succeeded
      TPWrite "X= ""Num:=x_coord; \If so, write x-coordinate on teach pedant
      Write com_ch_write,"OK"; \Write OK
      Close com_ch_write; \Close serial channel for writing
    ENDIF
    Open "sio1",com_ch_read\Read; \Open serial channel for reading
    recieved_string:=ReadStr(com_ch_read); \Read serial channel for 60s and copy to recieved_string
    IF error_check=1 THEN \Checking for error
      error_check:=0;
      GOTO home;
    ENDIF
    \Close serial channel for reading
    ok:=StrToVal(recieved_string,y_coord); \Convert string to number
    IF ok=TRUE THEN \Checking if succeeded
      TPWrite "Y= ""Num:=y_coord; \If so, write y-coordinate on teach pedant
      Write com_ch_write,"OK"; \Write OK
      Close com_ch_write; \Close serial channel for writing
    ENDIF
    Open "sio1",com_ch_read\Read; \Open serial channel for reading
    recieved_string:=ReadStr(com_ch_read); \Read serial channel for 60s and copy to recieved_string
    IF error_check=1 THEN \Checking for error
      error_check:=0;
      GOTO home;
    ENDIF
    \Close serial channel for reading
    ok:=StrToVal(recieved_string,z_coord); \Convert string to number
    IF ok=TRUE THEN \Checking if succeeded
      TPWrite "Z= ""Num:=z_coord; \If so, write z-coordinate on teach pedant
      Write com_ch_write,"OK"; \Write OK
Machine vision camera calibration and robot communication

Close com_ch_write; \Close serial channel for writing
ENDIF
Open "sio1:\com_ch_read\Read; \Open serial channel for reading
reieved_string:=ReadStr(com_ch_read); \Read serial channel for 60s and copy to received_string
IF error_check=1 THEN  \Checking for error
error_check:=0;
GOTO home;
ENDIF
Close com_ch_read;  \Close serial channel for reading
ok:=StrToVal(received_string,z_coord); \Convert string to number
IF ok=TRUE THEN  \Checking if succeeded
TPWrite "Z= "\Num:=z_coord;
IF ok=TRUE THEN  \If so, write y-coordinate on teach pendant
Write com_ch_write,"OK";
Close com_ch_write;  \Close serial channel for writing
ENDIF
path_vector_x{counter_1}:=x_coord; \Setting first x-coordinate in vector
path_vector_y{counter_1}:=y_coord; \Setting first y-coordinate in vector
path_vector_z{counter_1}:=z_coord; \Setting first z-coordinate in vector
counter_1:=counter_1+1;  \Increment counter and position in coordinate vector
TPWrite "Counter 1= "\Num:=counter_1;
ELSEIF received_string="stop" THEN \Checking if transmission is ending
TPWrite "Com stopped";  \Write teach pendant
move Coord;   \Jump to routine move_coord
ELSE
TPWrite "No coordinates to read";
ENDIF
Close com_ch_write;  \Close serial channel for writing
Close com_ch_read;  \Close serial channel for reading
ENDWHILE
ERROR   \Error routine
IF ERRNO = ERR_DEV_MAXTIME THEN \Checking if error is time for reading exceeded, 60s
Close com_ch_read;  \Close serial channel for reading
TPWrite "No data to receive!";
error_check:=1;
TRYNEXT;   \Try next instruction
ENDIF
ENDPROC
PROC move_coord()
WHILE counter_2<counter_1 DO  \Loop for how many coordinates that has been recieved
MoveL Offs(p10,path_vector_x{counter_2}, path_vector_y{counter_2}, path_vector_z{counter_2}), v100,fine,tool0\WObj:=wobj_cam;
counter_2:=counter_2+1;  \Increase counter for next location in coordinate vector
TPWrite "Counter 2= "\Num:=counter_2;
ENDWHILE
INTERSECT counter_1:=1;  \Reset counter
INTERSECT counter_2:=1  \Reset counter
ENDPROC
ENDMODULE