UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

An Autonomous System for the Evaluation of a Robot Tool Center Point using Computer Vision

Relatore:
Enrico Pagello

Laureando:
Federico Toffano

April 20, 2015
A.Y. 2014/2015

A mia mamma

Abstract

The main purpose of this thesis is to study a method for finding the geometric relationship between the TCP (tool center point) of a tool attached to the robot flange (the last axis) and the robot itself using a camera.

The proposed method requires the rotation matrix between the reference systems related to the robot and the camera and an initial tool positioning moving the robot using the manual control system in order to make the TCP detectable by the camera.

The image processing algorithms were designed to find the TCP on the image of any symmetric tools. Once identified the TCP on the image, the method to find the geometric relationship between robot and the three-dimensional TCP can be applied to any tool.

From a theoretical point of view the TCP computed is not an approximation. Any error is due to intrinsic system errors as camera discretization, robot pose accuracy and image processing algorithms accuracy.

The proposed method computes the TCP performing two rotations from the initial flange pose around two different rotation axis. There is no constraints about the rotation angle but wide rotation will reduce the error amplification. An analysis of errors has been performed to understand limitations of this method.

Contents

List of Figures

1 INTRODUCTION

1.1 BACKGROUND

Usually robots delivered by the manufacturer are calibrated during production. The calibration procedure allows to know position and orientation of each axis during a movement.

On last axis (tool flange) can be attached any tool. A point called TCP (tool center point) has to be calibrated to move the robot with respect to the tool assembled: it is the origin of a reference system representing the tool position and orientation.

For each tool can be defined a different TCP, meaning that changing a tool the relative calibration has to be performed. Furthermore, while the robot is moving, it could collide with something thus requiring a new TCP calibration.

Typically each robot has a manual procedure to calibrate the TCP. It can be computed using manual move controls to define four different positions keeping the TCP on the same point: for each combination of three of the four flange poses, a sphere whose center should be the TCP is computed. Then the TCP estimation will be the point that minimize the distance from each sphere center found. The result of this method depends on the operator skill to positioning the robot on the different poses keeping the TCP on the same position. Usually the accuracy achievable is around $0{,}5mm$.

1.2 OBJECTIVES

The company Euclid Labs requires an autonomous method to calibrate the TCP of symmetric tools using a camera. The Objective is to improve the accuracy of the manual method.

At the request of Euclid Labs no external image processing library have to be used. The vision algorithms have to be written with the integrated development environment Visual Studio using C# as programming language.

Figure 1.1 – Euclid Labs logo

The final demonstration software has to be a window in which every robot movements have to be simulated before the real execution.

1.3 STATE OF THE ART

1.3.1 Electronic calibration systems

Some manufacturers develop tool calibration system for robots that uses rotation-symmetric tools. These systems measure the tool's position electronically across all six dimensions and automatically adjusts the tool's position using an in-line sensor system. On Figure 1.2 is shown advintec TCP-3D designed by Leoni that is an istance of these kind of systems.

Figure 1.2 – Leoni TCP calibration system

There are several problems regarding these kind of systems that lead to search for others solutions:

- the workspace is restricted
- there is no user interface
- “black box” system

1.3.2 Camera based calibration systems

There are some patents with several solutions to solve this problem using a camera but even in this case there are some restrictions. The common basic idea is to move the flange in at least three different poses keeping the TCP on the same point and then compute the relationship between TCP and flange defining and solving a set of constraints related to these flange poses.

The easier solution computes the TCP without approximations from a theoretical point of view but it requires two robot flange rotations of 90 degree around two perpendicular axis [3] [4]. The main problem of this solution is that there could not be enough space to perform this kind of rotations and an initial TCP approximation is required.

Another solution to compute the TCP makes use of mean squares error algorithm [2] [8]. In this case there is no restrictions about the rotation angles amplitude and it can be applied even without initial TCP approximations but to reach a reasonable accuracy many flange poses has to be used to define the system of equations.

A third solution computes the TCP defining a set of equations using several flange poses and keeping the TCP on the camera optical axis [2]. This method requires at least four flange poses and allows to define system of second order equations. This method has no approximations from a theoretical point of view but second order constraints lead to high intrinsic system errors amplification as camera discretization and robot pose inaccuracy.

1.4 SYSTEM SETUP

System components are:

- Manipulator: KUKA AGILUS1100 SIXX
- Camera: Blaser Ace acA2040-25gm (2048x2048 pixels)
- Lens: Goyo GMTHR35028MCN (F/2.8 50mm)
- Led backlight: PHLOX SLLUB (200 X 200 mm, IP-65)
- PC: Asus asus n550jk (Intel Core i7-4700HQ (2.40 GHz))

Manipulator, camera and PC are connected to a local Ethernet network.

Manipulator and PC communicate reading and writing two shared variables stored on the robot control unit:

- *flag:* a Boolean variable that will be set to $true$ when a new movement has to be performed. Once the movement has been performed it will be set to $false$.
- *pos:* contains the manipulator position to reach. It can be expressed in different coordinate systems.

The PC handles the main software. It elaborates camera images and computes manipulator positions.

The following routine is used to command the manipulator from the PC:

1. set the new position on the variable *pos*
2. set to true the variable *flag*
3. suspend execution until the manipulator will set the variable *flag* to false.

While the manipulator executes the following loop:

1. check the variable *flag*
2. if *flag* value is $true$ move the active TCP to the new position *pos*
3. set *flag* to $false$.

The Backlight is an illuminated surface positioned in front of the camera and behind the tool with the purpose to enhance the contrast to detect more easily the TCP using the camera.

1.5 ORGANISATION OF THE DOCUMENT

0 Errore. Il risultato non è valido per una tabella.	Describes the purpose of this thesis and the configuration of the devices used
1 Errore. Il risultato non è valido per una tabella.	Detailed information about the manipulator and its reference systems and a brief description of the simulator used
3 Image processing	All the theory regarding image processing algorithms used and the explanation of the method used to detect the TCP with the camera
4 TCP calibration	Detailed analysis of the method used to calibrate the TCP including the interaction between software and camera
5 Results	The results obtained with several test to analyse the precision and the accuracy of the method proposed
6 Conclusions	Final discussion of the results achieved and possible future developments
A Appendix	Explanation of the least squares method with normal equations

Table 1.1 – Organisation of the document

2 ROBOTICS

2.1 KUKA AGILUS1100 SIXX

This small industrial robot is a 6-axis arm and it have been designed to achieve high velocity, at the expense of a reduced payload.

2.1.1 Description

The robot is composed of the following components visible in Figure 2.1:

1. In-line wrist A4, A5, A6
2. Arm A3
3. Link arm A2
4. Rotating column A1
5. Electrical installations
6. Base frame

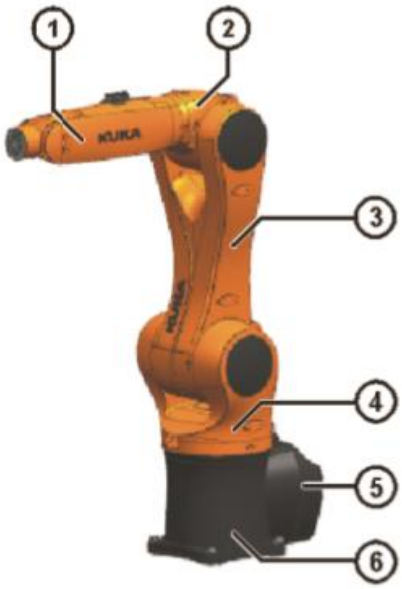


Figure 2.1 – Kuka Agiuls1100 sixx

In-line wrist A4, A5, A6	The robot is fitted with a 3-axis in-line wrist. The in-line wrist consists of axes 4, 5 and 6.

Arm A3	The arm is the link between the in-line wrist and the link arm. The arm is driven by the motor of axis 3.
Link arm A2	The link arm is the assembly located between the arm and the rotating column. It houses the motor and gear unit of axis 2.
Rotating column A1	The rotating column houses the motors of axes 1 and 2. The rotational motion of axis 1 is performed by the rotating column.
Electrical installations	The electrical installations include all the motor and control cables for the motors of axes 1 to 6. The electrical installations also include the RDC box, which is integrated into the robot.
Base frame	The base frame is the base of the robot. Interface A1 is located at the rear of the base frame.

Table 2.1 - Kuka Agiuls1100 sixx components

2.1.2 Technical data

The robot joints are mechanically and software constrained. The maximum values are shown in Table 2.2, with the maximum speed that each axis can achieve.

Axis	**Range of motion, software limited**	**Speed with rated payload**
1	+/-170°	360 °/s
2	+45° to -190°	360 °/s
3	+156° to -120°	360 °/s
4	+/-185°	360 °/s
5	+/-120°	360 °/s
6	+/-350°	360 °/s

Table 2.2 - Kuka Agiuls1100 sixx technical data

The KUKA Agilus Sixx 1100 technical data are listed in Table 2.3. The working area is defined by the size of the robot and the values which can assume the joints represented in Figure 2.2.

Max. reach	1,101 mm
Max. payload	10 kg
Pose repeatability	±0.03 mm
Number of axes	6
Mounting position	Floor, ceiling, wall
Robot footprint	209 mm × 207 mm
Weight (excluding controller), approx	54 kg

Table 2.3 - Kuka Agiuls1100 sixx technical data

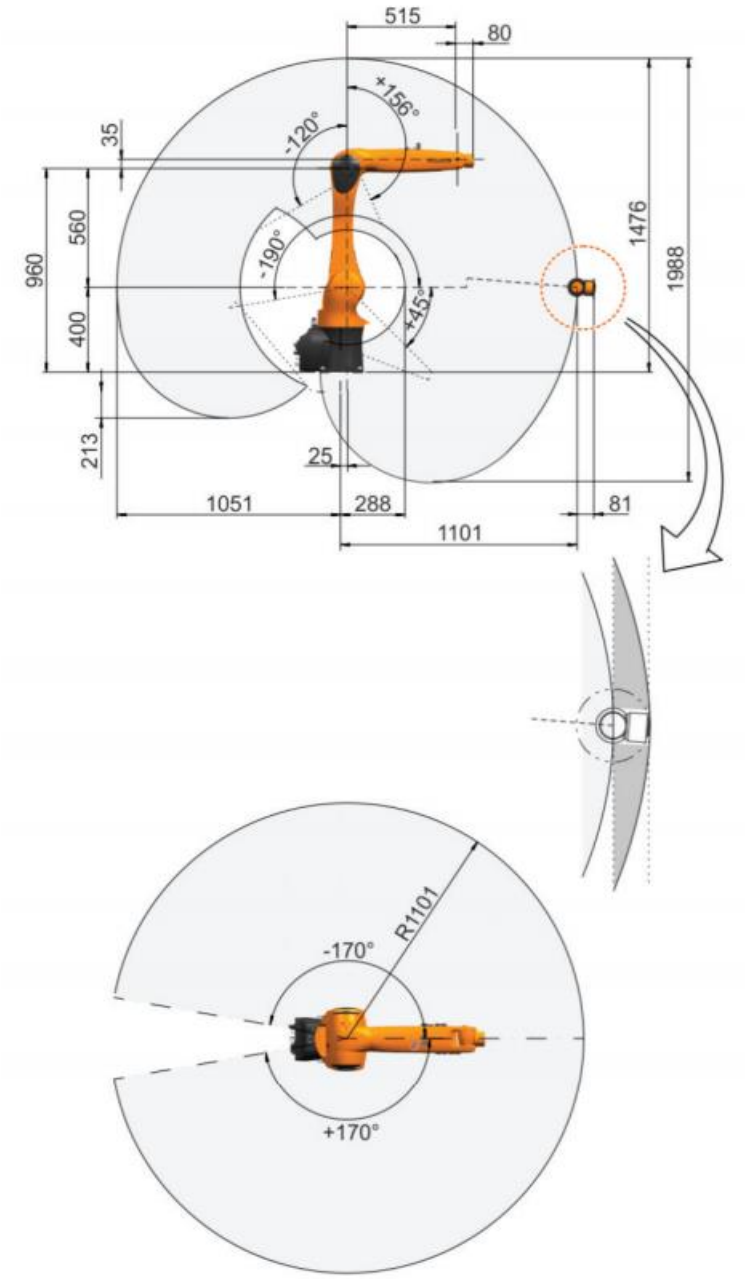


Figure 2.2 – Kuka Agiuls1100 sixx size and joint values

2.2 EUCLID LABS SIMULATOR

The simulator developed by Euclid Labs has a user interface with a window containing the manipulator rendering (built using CAD data) and some buttons to change the current robot position specifying the value of each axes or setting the Cartesian position expressed in the coordinate system selected.

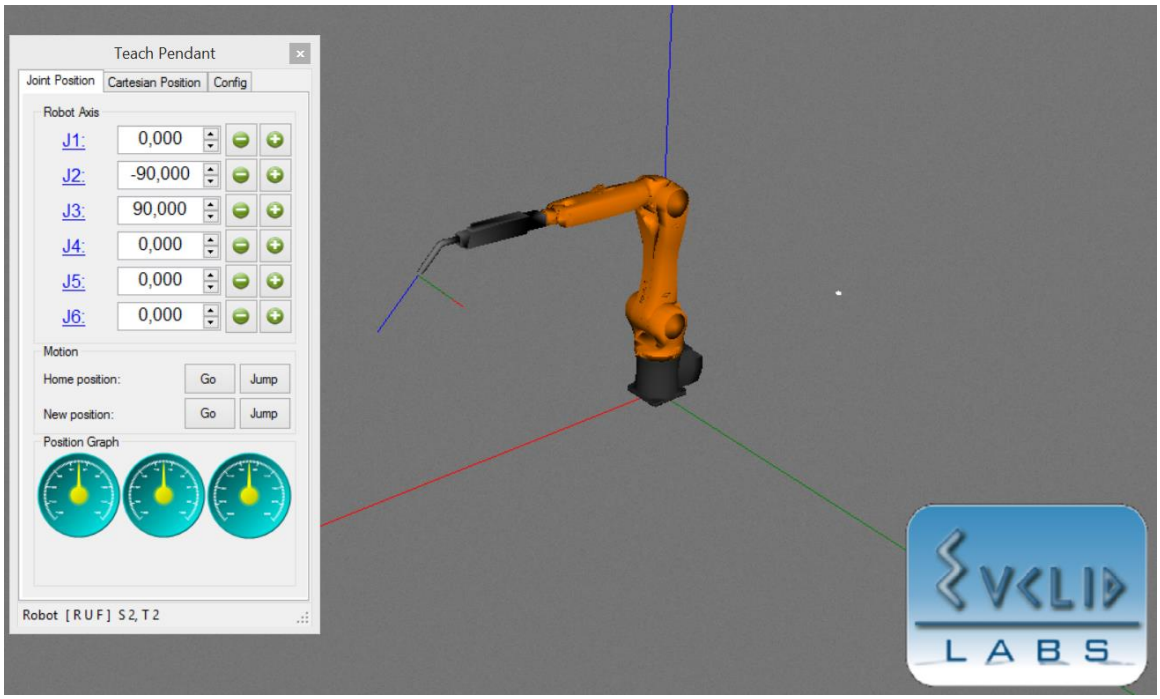


Figure 2.3 – Euclid Labs simulator

2.3 COORDINATE SYSTEMS

2.3.1 Tool Center Point

The TCP (tool center point) is the reference point of the tool attached to the robot flange. Usually the robot movements are programmed defining a path with respect to the TCP that can be represented in different coordinate systems. It could be any point but typically corresponds to the tool tip, e.g. the muzzle of a welding torch. Furthermore, there could be several TCP, but only one at time can be active.

2.3.2 Robot Base frame

The robot base coordinate system corresponds to its base frame. The red, green and blue axis on Figure 2.4 represent the x, y and z axis.

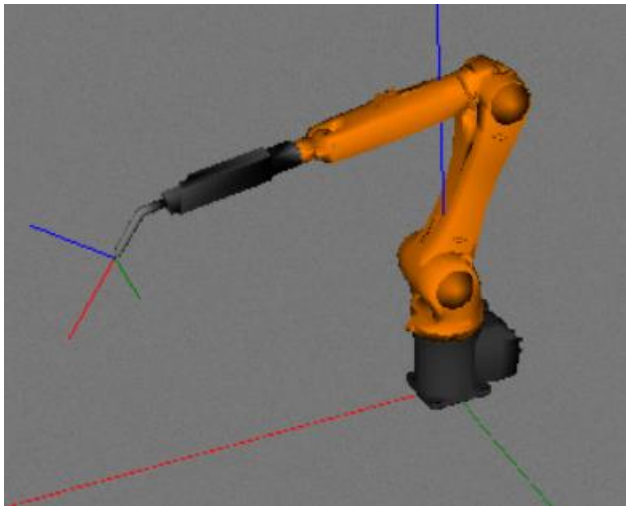


Figure 2.4 – Robot base frame

In this work, the robot base coordinate system corresponds to the world coordinate system.

2.3.3 Wrist frame

The wrist frame corresponds to the robot flange (the last axis of the robot where the tools have to be attached). The origin of the wrist frame is the center of the visible flange surface. The red, green and blue axis on Figure 2.5 represent the x, y and z axis.

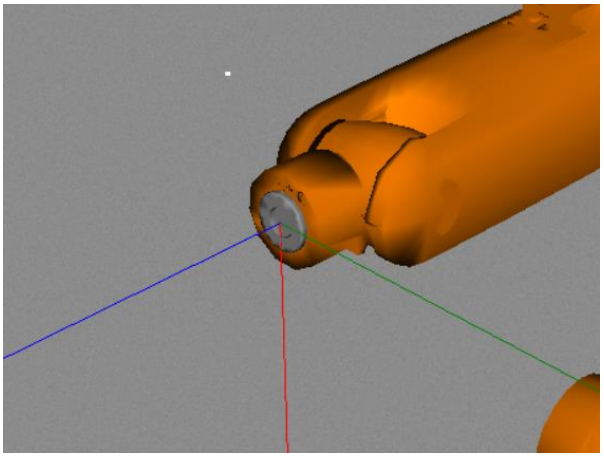


Figure 2.5 – Wrist frame

2.3.4 Tool frame

The tool frame is the coordinate system in which the origin corresponds to the tool center point. The z-axis has usually the same direction of the tool tip and defines its work direction. The red, green and blue axis on Figure 2.6 represent the x, y and z axis.

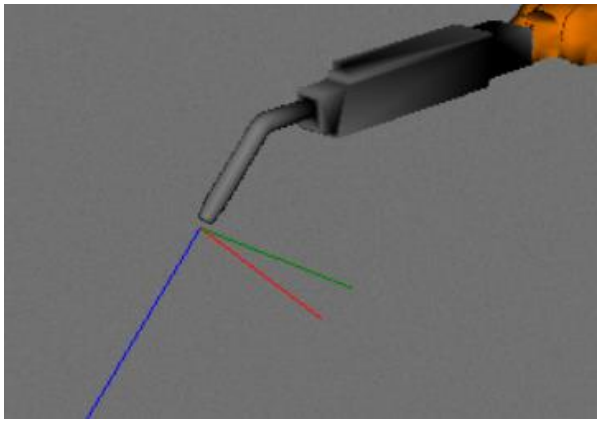


Figure 2.6 – Tool frame

3 IMAGE PROCESSING

3.1 PINHOLE CAMERA MODEL

Pinhole camera model is a projective model that approximate the image acquisition process of a digital camera through simple geometric relationships. The purpose is to find the relation between image points and their position in the world. It performs well as long as the lens is thin and no wide-angle lens is used.

In Figure 3.1 are shown the relationships between the following coordinate systems:

1. *World coordinate system* has origin at point O and its orientation is given by U_x, U_y and U_z.
2. *Camera coordinate system* has the origin at point C that is also the focal point. Axis U_z' is aligned with camera optical axis and its direction points the image plane. Plane given by U_x' and U_y' is parallel to the camera image plane.
3. *Image coordinate system* has axis u and v aligned with U_x' and U_y' of *camera coordinate system*. The origin is the top left corner and the optical axis intersect image plane center.

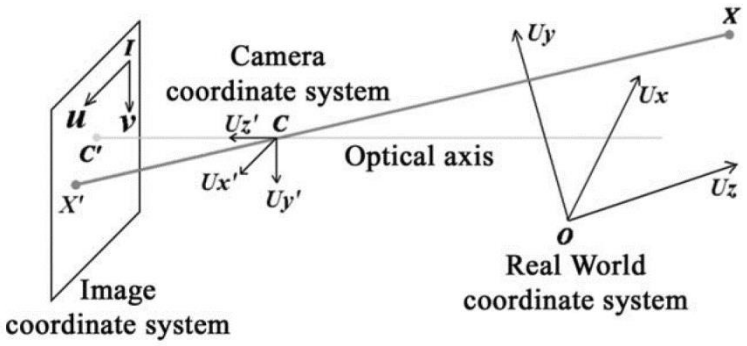


Figure 3.1 – Pinhole camera model

The relationship between these coordinate systems can be represented by the following equation:

$$z_i \begin{bmatrix} u_{n_i} \\ v_{n_i} \\ 1 \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} R_{OC} & T_{OC} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \quad (3.1)$$

Where $[X_i, Y_i, Z_i]^T$ represents a point expressed in *world coordinate system*, $[x_i, y_i, z_i]^T$ is the same point but expressed in *camera coordinate system* and $\left[u_{n_i}, v_{n_i}\right]^T$ is the corresponding normalized image coordinate. $R \in \mathbb{R}^{3x3}$ and $t \in \mathbb{R}^3$ are the *extrinsic parameters* and represent rotation and translation from the *world coordinate system* to the *camera coordinate system*.

Real camera image plane is usually represented with the following transformation from the normalized image coordinate:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & v_0 \\ 0 & \beta & u_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{n_i} \\ v_{n_i} \\ 1 \end{bmatrix} \quad (3.2)$$

Where $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are scaling factors from normalized image plane to real image plane along u_n and v_n directions, $\gamma \in \mathbb{R}$ represents the skew coefficient between u and v axis and $[u_0, v_0] \in \mathbb{R}^2$ represent the intersection of real image plane with optical axis.

A simplified schema about how the image is acquired by the sensor is shown in *Figure 3.2*. The point (x_i, y_i, z_i) expressed in camera coordinate system,

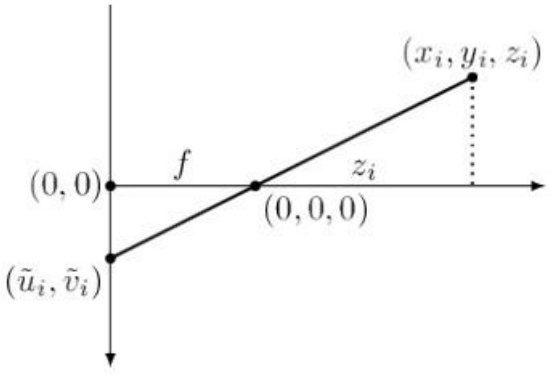


Figure 3.2 – Point projection on image plane

Where f is the focal length.

3.2 IMAGE PROCESSING ALGORITHMS

3.2.1 Grayscale histogram

The histogram of a grayscale image with 256 levels is a function $h_f(x)$ where the domain is all the integers contained into the interval [0,255] that for each pixel represents the possible values of the grayscale image while the codomain is the number of pixels with the corresponding value found: $\rho = \frac{n_l}{k}, k \in \mathbb{R}, k \geq 1$

1. Set all the elements of the array $h_f(x)$ to zero
2. For each pixel (x, y) of the image f, to increment $h_f(f(x, y))$ by 1

Algorithm 3.1 - Computing the histogram

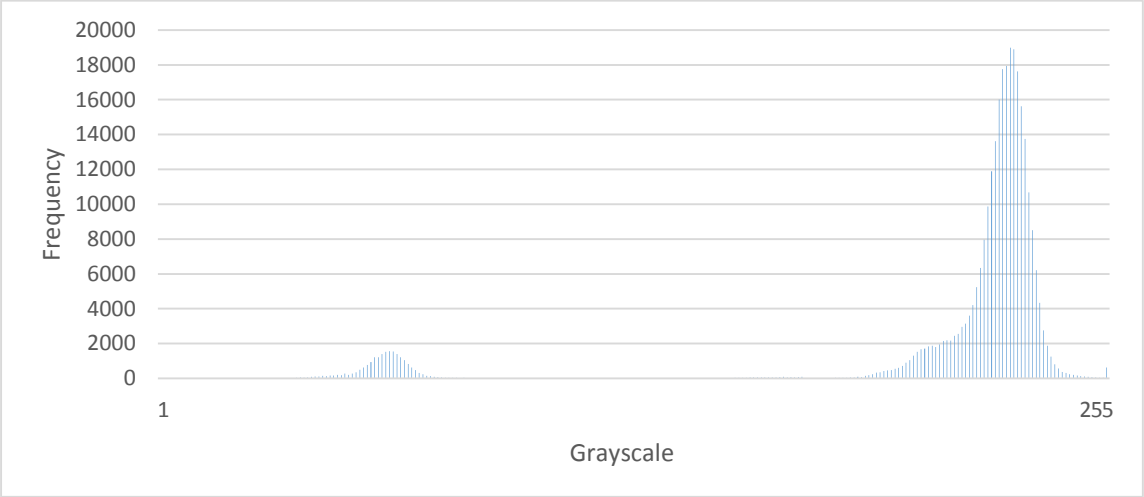


Figure 3.3 - Histogram

3.2.2 Binary image

Since the tool is in backlight, a threshold value can divide the light background from the darker tool. To avoid external light inference, the threshold is not fixed: computing the image histogram the threshold T can be defined dynamically finding the minimum value between the two higher local maximum values, this should corresponds to a value between the mean intensity of the background and the mean intensity of the tool.

Once found the threshold T using the histogram, the binary image $b_f(x, y)$ can be computed as follows:

$$b_f(x,y) = \begin{cases} 0 \; if \; f(x,y) > T \\ 1 \; if \; f(x,y) \leq T \end{cases} \qquad (3.3)$$

3.2.3 Gaussian filter

This filter smooth the image convolving a Gaussian kernel H with the image itself. H is a matrix of $2k+1 \times 2k+1$ elements, with $k \in \mathbb{N}$, where values are defined as follows:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \, e^{-\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}} \qquad (3.4)$$

Where σ is the standard deviation of the probability distribution.

3.2.4 Sobel operator

This operator can be used to approximate the image gradient convolving the image with two kernels $S_x \in \mathbb{Z}^2$ and $S_y \in \mathbb{Z}^2$:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad (3.5)$$

The result is two bidimensional matrix with the same size of the image used as input where, for each image pixel (i, j), the corresponding cells represents an approximation of the horizonantal derivative $G_{x_{ij}}$ and the vertical derivative $G_{y_{ij}}$.

Then $\left(G_{x_{ij}}, G_{y_{ij}}\right)$ is the gradient approximation of the image pixel (i, j) and the corresponding gradient magnitude G_{ij} and direction θ_{ij} can be computed as follows:

$$G_{ij} = \sqrt{G_{x_{ij}}^2 + G_{y_{ij}}^2} \qquad \theta_{ij} = \; arctan\left(\frac{\left|G_{y_{ij}}\right|}{\left|G_{x_{ij}}\right|}\right) \qquad (3.6)$$

3.2.5 Canny's Edge Detection

John F. Canny developed this algorithm in 1986 to detect a wide range of edges in images. This edge detector has three main criteria to solve the problem:

- *The detection*: false detection should be avoided and important edge should not be missed
- *The localization*: minimize the distance between the edge detected and the edge center
- *The one response:* the algorithm should minimize multiple response to a single edge

The basic idea of Canny´s edge detection algorithm is to consider a point in the image as an edge if the first derivatives has local maxima along its gradient direction.

The algorithm can be broken down to five different steps:

1. To remove the noise and obtain better results, the first step of the algorithm is to smooth the image with a *Gaussian filter*. This can be done convolving the image with a Gaussian filter.
2. Compute the *image gradient* to estimate local edge magnitude G_{ij} and direction θ_{ij} for each image pixel (i, j). *Sobel operator* can be used for this purpose.
3. Find the edges location using *Non-maximum suppression* Algorithm:
    a. Round the direction to the nearest multiple of 45°
    b. Compare the gradient magnitude with the one related to the next pixel along the gradient direction
    c. Suppress the current pixel value if its gradient magnitude is lower

    The purpose is to find all local maxima in the gradient magnitudes along the related gradient directions.

4. At this point, the image contains the edge detected but there could be some spurious response. A way to keep only the true edges is to threshold the related gradient magnitude with *hysteresis*:
    a. If the gradient magnitude is above a high threshold then the corresponding pixel is a true edge

b. If the gradient magnitude is under a low threshold then the corresponding pixel is a false edge
c. If the gradient magnitude is between the low and high thresholds then the corresponding pixel is a true edge only if there is at least one other true edge among its neighbours

5. *Feature synthesis* approach: The general principle is to start from the smallest scale, then synthesise the larger scale outputs that would occur if they were the only edges present. Then comparing the large scale output with the synthetized edge response, additional edges are marked only if the large scale output is significantly greater than the synthetic prediction. Different scale for Canny's edge detector are represented by different standard deviation σ of the Gaussian filter.

1. Convolve the image with a *Gaussian filter* with standard deviation σ
2. Compute the *image gradient*
3. Find the location of the edges using *Non-maximum suppression* Algorithm
4. Threshold edges with *hysteresis*
5. Repeat steps 1 – 5 with ascending values of the standard deviation σ
6. Merge the edges found according to *Feature synthesis* approach

Algorithm 3.2 – Canny edge detector

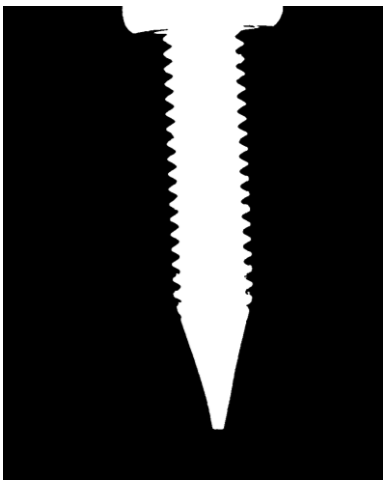


Figure 3.4 – Binary image

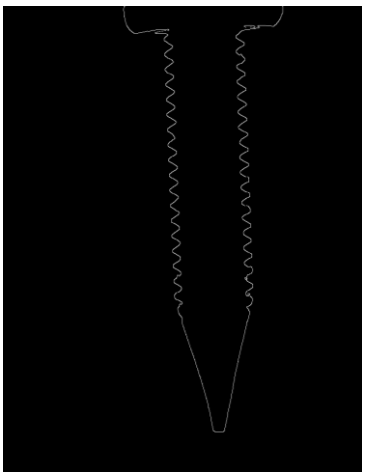


Figure 3.5 – Edge found with Canny's edge detection

3.2.6 Hough transform

Conceived by Hough in 1962, in its first version, the purpose was the recognition of straight lines. The main idea is that any point on the entire image gives a contribution to the overall recognition: given a point, can be defined a sheaf of lines and for each couple of points there is only one common line. Computing a discrete sheaf of lines for each image point, if a specific line appears n times, then there is a line on the image with n points.

A bidimensional line can be represented with two parameters, then defining a bidimensional matrix where columns and rows corresponds to the first and second line parameter, the lines found can be counted incrementing the corresponding cell. The result is an accumulation function defined on the lines parameter space. Changing the parameter space, this algorithm can be used to find various geometric shape.

A line can be represented in several ways, the most common equation used is $y = mx + q$ where m is the gradient of the line, q is the y-intercept of the line and (x, y) represent a line point. Changing m and q all the lines that through the point (x, y) can be defined. With this kind of representation, the parameter m increases immeasurably when the line become vertical and has small variations when the line become horizontal then this representation is not well suited to define a discrete parametric space. A suitable line representation could be $r = x \sin\theta + y \cos\theta$ where $r \in \mathbb{R}$ is the line distance from the center and $\theta \in \mathbb{R}$ is the orientation of r with respect to the abscissas axis.

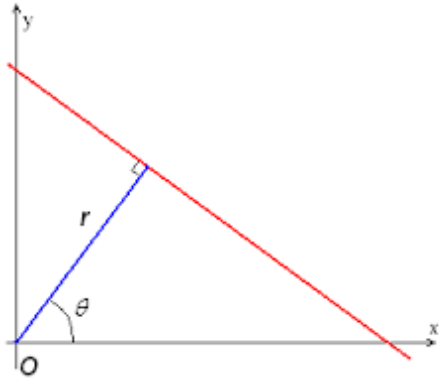


Figure 3.6 - Hough transform line representation

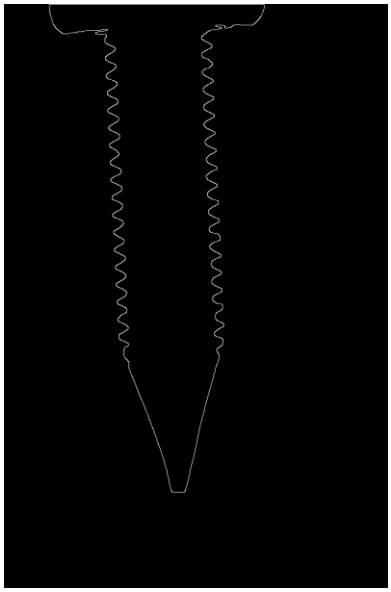
With this representation, for each image point $(x, y) \in tool\ T \subset \mathbb{Z}^2$, the corresponding lines $l_{i_{xy}}$, with $i \in [0,179]$ can be computed varying the orientation θ_i from 0 to 179 and computing the corresponding distance r_i from the center as a function of x, y and θ_l

Once all the pixels were analysed, the points in the parameter space that have accumulated the most number of "votes" represents the lines that have high probability of being present in the image.

A further step is to keep the lines that have a minimum number of points, the decision threshold ρ could be a fraction of the number of points n_l of the largest line found: $\rho = \frac{n_l}{k}, k \in \mathbb{R}, k \geq 1$

1. Define the parameter space (r, θ) with an accumulation matrix M
2. Initialize all the M element to zero
3. For each pixel (x, y) compute $r_i = x \sin\theta_i + y \cos\theta_i$ with $\theta_i \in [0,179]$ and increment by a unit $M(r_i, \theta_i)$
4. Find the points of local maximum of the accumulation matrix M
5. Ignore lines with a number of matches less than ρ

Algorithm 3.3 – Hough transform algorithm



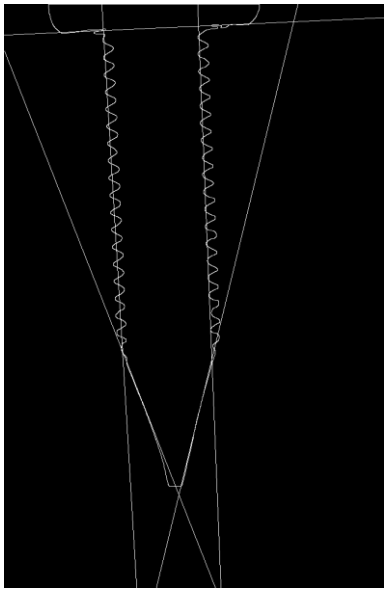


Figure 3.7 – Edge image

Figure 3.8 – Lines found with Hough transform

3.2.7 Thinning

This is a morphological operator used to reduce the area of an element contained into a binary image $X \in \mathbb{Z}^2$. The resulting image $X' \in \mathbb{Z}^2$ is:

$$X' = X \oslash B = X / X \otimes B \quad (3.7)$$

Where $B = (B_1, B_2),\ B_1 \in \mathbb{Z}^2, B_2 \in \mathbb{Z}^2, B_1 \cap B_2 = \emptyset$ is called *composite structuring element* and $X \otimes B = \{x: B_1 \subset X\ and\ B_2 \subset X^c\}$ is the *hit-or-miss* transformation.

In other words, once the thinning operator has been applied, a pixel x equal to one of the initial image will be zero on the resulting image if the set of surroundings pixels B_1 of x are equal to one and the set of surroundings pixels B_2 of x are equal to zero.

This operator has been implemented convolving the image with eight *composite structuring elements* taken from the *Golay alphabet*:

$$B_1 = \begin{bmatrix} 0 & 0 & 0 \\ * & 1 & * \\ 1 & 1 & 1 \end{bmatrix}\ B_2 = \begin{bmatrix} * & 0 & 0 \\ 1 & 1 & 0 \\ * & 1 & * \end{bmatrix}\ B_3 = \begin{bmatrix} 1 & * & 0 \\ 1 & 1 & 0 \\ 1 & * & 0 \end{bmatrix}\ B_4 = \begin{bmatrix} * & 1 & * \\ 1 & 1 & 0 \\ * & 0 & 0 \end{bmatrix}$$
$$B_5 = \begin{bmatrix} 1 & 1 & 1 \\ * & 1 & * \\ 0 & 0 & 0 \end{bmatrix}\ B_6 = \begin{bmatrix} * & 1 & * \\ 0 & 1 & 1 \\ 0 & 0 & * \end{bmatrix}\ B_7 = \begin{bmatrix} 0 & 0 & 0 \\ * & 1 & * \\ 1 & 1 & 1 \end{bmatrix}\ B_8 = \begin{bmatrix} 0 & 0 & * \\ 0 & 1 & 1 \\ * & 1 & * \end{bmatrix} \quad (3.8)$$

3.2.8 Homotopic skeleton detection

To define what is an image skeleton, *maximal ball* needs to be specified: A ball $B(p,r) \subset \mathbb{R}^n$ with radius $r \in \mathbb{R}$ and center $p \in \mathbb{R}^n$ is said to be *maximal* in a set $X \subset \mathbb{R}^n$ if and only if there exists no other ball included in X and containing B:

$$let\ X \subset \mathbb{R}^n and\ \ B\ ball \subset \mathbb{R}^n, then\ B\ is\ maximal\ ball\ of\ X\ if$$

$$\forall\ B' ball \subset \mathbb{R}^n, B \subseteq B' \subseteq X \Rightarrow B' = B$$

The *skeleton* $S(X) \subset \mathbb{R}^n$ by *maximal balls* of a set $X \in \mathbb{R}^n$ is the set of the centers of its maximal balls:

$$S(X) = \{p \in X : \exists\, r \geq 0, B(p,r)\ is\ maximal\ ball\ of\ X\} \qquad (3.9)$$

A sequential *thinning* with the eight *composite structuring elements* defined in section 3.2.7 applied until idempotency is reached, produces a homotopic substitute in $\mathbb{Z}^2$ of the *skeleton* defined by *maximal balls*.

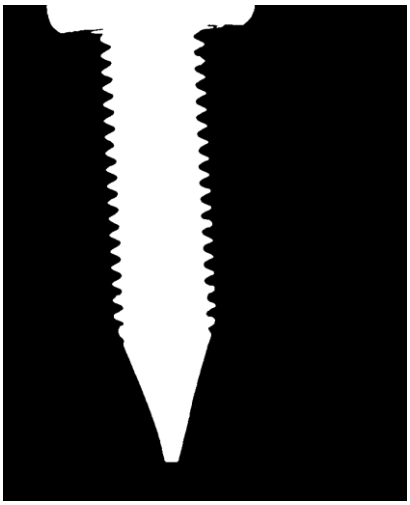


Figure 3.9 – Source image

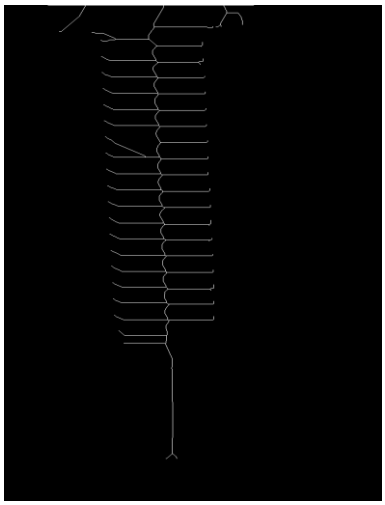


Figure 3.10 – Homotopic skeleton

In some unfortunate cases this algorithm performs a *sequential pixel elimination*: with only one iteration a tool portion become a line due to the tool erosion from only one side (

Figure 3.12) leading to a false skeleton detection. The problem can be avoided adding a flag matrix M reinitialized at the beginning of each cycle with false values and set to true a cell $M(x_i, y_i)$ when the corresponding image pixel $I(x_i, y_i)$ is going to be set to zero. This allows to evaluate a pixel erosion only if the neighbouring pixels that have to be analysed have not been deleted during the current cycle. The resulting skeleton is a little different but it is well suited for the purpose.

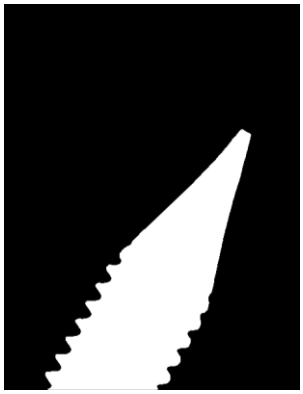


Figure 3.11 – Source image

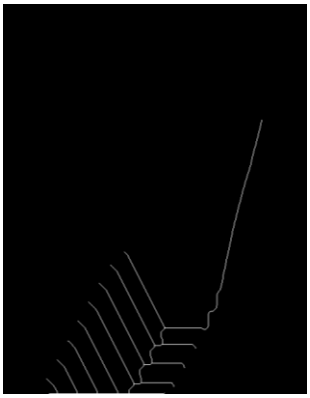


Figure 3.12 – Wrong skeleton detection

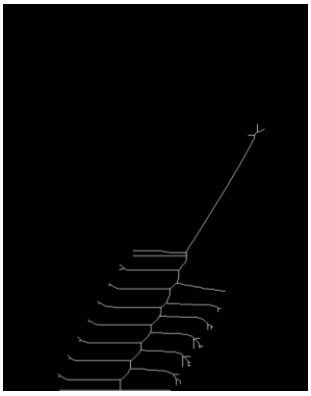


Figure 3.13 – Correct skeleton detection

3.3 KEY POINT DETECTION

The TCP projection on the image can be detected finding tool main direction and then searching the minimum point along the found direction. With symmetric tools, this point should correspond to the tool tip.

3.3.1 Tool direction

The Hough transform 3.2.6 can be used for this purpose. Applying this algorithm on the image, the resulting accumulation matrix contains the number of points belonging to each line found. A line is represented by the line distance from the image center r_i and the rotation angle θ_i of r_i with respect to X axis (see section 3.2.6 for more details).

The direction of a line can be represented by the angle between the X axis and the line itself as shown in Figure 3.14

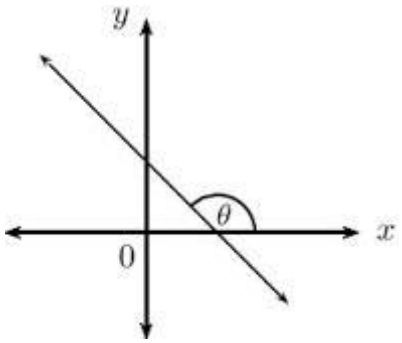


Figure 3.14 - Line direction

With this representation the line direction can be obtained as $\theta_i' = \theta_i + \frac{\pi}{2}$ since it is perpendicular to the direction of r_i.

Now the problem is which line consider as main tool direction.

3.3.1.1 Sum of lines direction

The first idea was to apply the Hough transform on the image containing the edges found with Canny's algorithm and then to find main direction with the following method:

1. Define a new vector of directions D that represents all the possible directions θ_i, where each cell contains the counting of the points of all the lines with the corresponding direction.
2. Then the main tool direction can be found identifying the cell of D with the maximum value and thereafter calculating the line direction θ_i'.

In some cases, this method cannot performs well. For instance applying this method on a triangular tip of a tool, the direction will be not accurate Figure 3.16.

Figure 3.15 – Source image

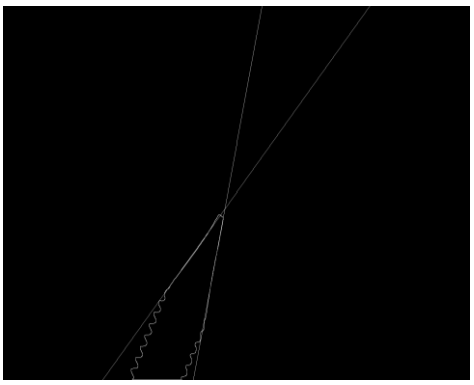


Figure 3.16 – Wrong tool directon

3.3.1.2 Largest line

A way to solve the problem related to 3.3.1.1 is to apply the Hough transform on the skeleton of the tool. With a symmetric tool, the *main skeleton part* is a line equidistant from the edges, which can be reasonably considered as tool direction.

In this case, to identify the line representing main tool direction, could be easier to find the absolute maximum on the accumulation matrix found through the Hough transform since it represents the line with more points found on the image that should correspond to the *main skeleton part*.

Figure 3.17 – Source image

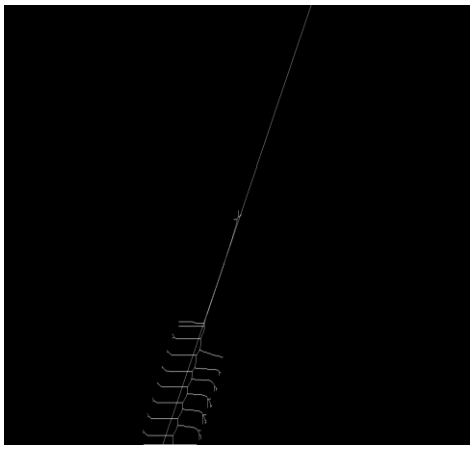


Figure 3.18 – Skeleton direction found

3.3.2 TCP localization

Once found the tool direction θ', rotating the image coordinate space U_{xy} of θ' anticlockwise, the TCP projection can be found among the tool points of local extrema along the new abscissas axis x'.

Let U_{xy} the initial image coordinate space with origin on image centre, the new coordinate space $U_{x'y'}$ with the abscissas axis parallel to tool direction has coordinates:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta' & -\sin\theta' \\ \sin\theta' & \cos\theta' \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (3.10)$$

The maximum and minimum points representing the tool along the x' axis correspond to a point on the tool tip and a point of image borders. Now, evaluating these points found expressed in coordinate system U_{xy}, understand which is the image border point is trivial so TCP is determined too.

3.3.3 TCP optimisation

There could be more than one point of extrema that can represent the TCP (

Figure 3.19). A way to solve this ambiguity is to calculate the mean of the surrounding points along the axis perpendicular to the tool direction: let $T' \in \mathbb{Z}^2$ the set of the points representing the tool with the coordinate system $U_{x'y'}$ (3.10), $(x'_{ext}, y'_{ext}) \in T'$ the point found with *TCP localisation* (section 3.3.2) and $S = \{(x', y') \in T' \ : \ |x' - x'_{ext}| < k, \ k \in \mathbb{N}\}$ the set of its surrounding points, then the TCP is:

$$(x_{tcp}, y_{tcp}) = \begin{bmatrix} \cos -\theta' & -\sin -\theta' \\ \sin -\theta' & \cos -\theta' \end{bmatrix} \begin{bmatrix} x'_{ext} \\ y'_{mean} \end{bmatrix} \ where \ y'_{mean} = \frac{\sum_{(x',y') \in S} y'}{\|S\|} \quad (3.11)$$

A result of this optimisation can be seen in

Figure 3.20 where the set S is represented by purple pixels while the TCP by grey pixels.

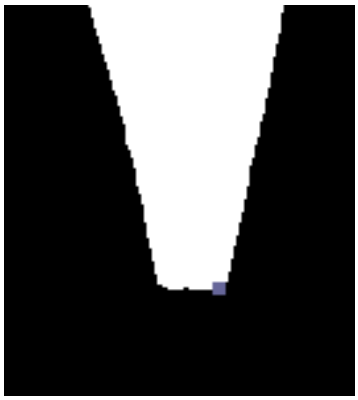


Figure 3.19 – Keypoint without tcp optimisation

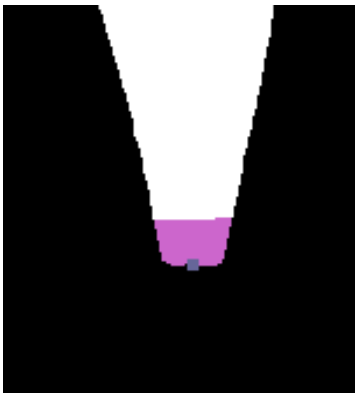


Figure 3.20 – Keypoint with tcp optimisation

4 TCP CALIBRATION

4.1 TOOL CENTER POINT REPRESENTATION

Let F_i the i^{th} position of the robot flange represented with the homogeneous transformation matrix (4.1) expressed in robot base coordinate system.

$$F_i = \begin{bmatrix} R_i & T_i \\ 0 & 1 \end{bmatrix}, \qquad i \in [1, N] \tag{4.1}$$

Where R_i is the three-dimensional rotation matrix from the robot base fame O_w and T_i is the three-dimensional translation vector from the robot base frame O_w. F_i can be obtained with the robot forward kinematics.

The tool center point can be represented with a further homogeneous transformation matrix H_{tcp} with respect to the robot flange coordinate system.

$$H_{tcp} = \begin{bmatrix} R_{tcp} & T_{tcp} \\ 0 & 1 \end{bmatrix} \tag{4.2}$$

Where R_{tcp} is the three-dimensional rotation matrix from the robot flange frame F_i and $T_{tcp} = \left(T_{tcp_x}, T_{tcp_y}, T_{tcp_z}\right)^T$ is the three-dimensional translation vector from the robot flange frame F_i.

4.2 MINIMUM NUMBER OF EQUATIONS

A position $p_i \in \mathbb{R}^3$ of the TCP expressed in robot base coordinate system can be computed as follows:

$$p_i = F_i \cdot t_{tcp}, \qquad i \in [1, N] \tag{4.3}$$

Where the frame F_i represents a robot flange pose and $t_{tcp} = \left(T_{tcp}, 1\right)^T$ is the homogeneous vector representing the TCP with respect to F_i.

For each couple of different robot flange poses F_i and F_j, with the TCP constrained to be on the same position ($p_i = p_j$), a TCP constraint can be defined:

$$F_i \cdot t_{tcp} - F_j \cdot t_{tcp} = p_i - p_j, \qquad i,j \in [1,N], i \neq j$$
$$(p_i = p_j) \Rightarrow [F_i - F_j] \cdot t_{tcp} = D_{ij} \cdot t_{tcp} = 0 \qquad (4.4)$$

The dimension of D_{ij} is 4x4. Since F_i and F_j are homogeneous matrix, all the values of last row of D_{ij} are zero.

The rank of D_{ij} (4.4) is two, this can be demonstrated but some concepts have to be defined.

Let R a three-dimensional rotation matrix:

1 *Orthonormal basis*: in $\mathbb{R}^3$ can be represented with a 3x3 matrix in which the columns represent three unit vector where each of them is parallel to one and only one axis of the three-dimensional space. R may be seen as an o*rthonormal basis* since it is an *orthogonal matrix* with determinant equal to one.

2 *Axis-angle representation*: R can be represented with the *axis-angle representation* $e = (\hat{e}, \theta)$ where θ is the rotation around the *axis of rotation* represented by the three-dimensional vector $\hat{e}$. As the rotation θ grows, any point rotates around the rotation axis $\hat{e}$ (Figure 4.1).

3 *Rotation plane*: it is a plane defined in $\mathbb{R}^3$ in relation with R, it contains the origin of the coordinate system and has the *normal* perpendicular to the axis of rotation $\hat{e}$ (Figure 4.1).

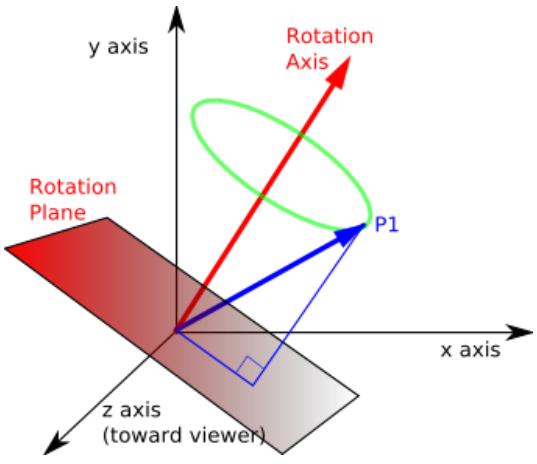
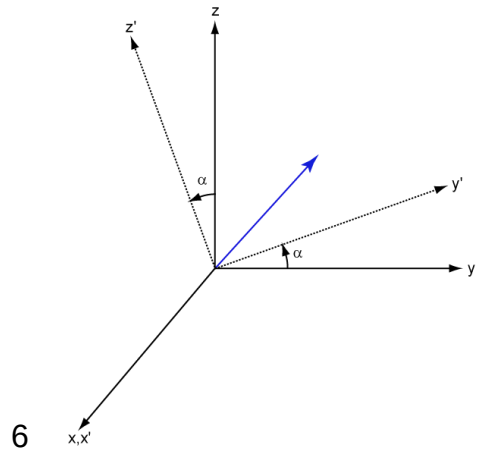


Figure 4.1 – Axis-angle representation

4 *Difference vector:* let p_1 a three-dimensional point and $p_2 = R \cdot p_1$ the point p_1 rotated with the matrix R. A *difference vector of* R is $v_{12} = P_1 - P_2$. All the *difference vectors of* R are contained into its *rotation plane* since they are perpendicular to the axis of rotation $\hat{e}$ of R.

5 *Correlated rotation matrix*: since a rotation matrix is an *orthonormal basis,* for each couple of rotations matrix R_i and R_j exists a *correlated rotation matrix* R_{ij} that rotates the three unit vectors represented by the columns of R_i (Figure 3.2 x, y, z) to a new position defined by the three columns of R_j (Figure 3.2 x', y', z').



6

Figure 4.2 – Rotation between two coordinate system

Statement 4.1: Let R_i and R_j two rotation matrix such that $R_i \neq R_j$. The matrix $R_i - R_j$ has rank equal to two.

Proof: The columns of $R_i - R_j$ can be seen as three *difference vectors* contained into the *rotation plane* of the *correlated rotation matrix* R_{ij} of R_i and R_j Then the rank of $R_i - R_j$ must be less or equal to two [1].

If these *difference vectors* were parallel then would exist $\alpha \in \mathbb{R}, \alpha \neq 0$ such that:

$$R_{i_1} - R_{j_1} = \alpha(R_{i_2} - R_{j_2}) \qquad (4.5)$$

Where $R_{i_1} - R_{j_1}$ and $R_{i_2} - R_{j_2}$ are the first and the second column of $R_i - R_j$, but since R_i and R_j are orthonormal basis:

$$R_{i_1} \perp R_{i_2} \Rightarrow R_{i_1}^T \cdot R_{i_2} = 0$$
$$R_{j_1} \perp R_{j_2} \Rightarrow R_{j_1}^T \cdot R_{j_2} = 0 \qquad (4.6)$$

Where $R_{i_1} \neq 0$, $R_{i_2} \neq 0$, $R_{j_1} \neq 0$ and $R_{j_2} \neq 0$.

Then if (4.5) is true, even the following equations are true:

$$(R_{i_1} - R_{j_1})R_{i_2} = \alpha(R_{i_2} - R_{j_2})R_{i_2}$$
$$R_{i_1}^T R_{i_2} - R_{j_1}^T R_{i_2} = \alpha R_{i_2}^T R_{i_2} - \alpha R_{j_2}^T R_{j_2} \Rightarrow$$
$$-R_{j_1}^T R_{i_2} = \alpha R_{i_2}^T R_{i_2} - \alpha R_{j_2}^T R_{i_2} \Rightarrow$$
$$-R_{j_1} = \alpha R_{i_2} - \alpha R_{j_2} \Rightarrow \qquad (4.7)$$
$$-R_{j_1}^T R_{j_2} = \alpha R_{i_2}^T R_{j_2} - \alpha R_{j_2}^T R_{j_2} \Rightarrow$$
$$0 = \alpha R_{i_2}^T R_{j_2} - \alpha R_{j_2}^T R_{j_2} \Rightarrow$$
$$\alpha(R_{i_2} - R_{j_2}) = 0$$

Then equation (4.5) is true if:

$$R_{i_2} - R_{j_2} = R_{i_1} - R_{j_1} = 0 \qquad (4.8)$$

But since $R_i \neq R_j$, at most one column of R_j can be equal to the same column of R_i then (4.8) cannot be true, meaning that even (4.5) cannot be true then the *difference vectors* cannot be parallel. Since at least two columns of $R_i - R_j$ are not parallel, $R_i - R_j$ has rank grater or equal to two [2].

$[1] \cap [2] \Rightarrow R_i - R_j$ has rank equal to two.

Qed

***Statement 4.2*:** If $R_i \neq R_j$, the last column of D_{ij} (4.4) $T_i - T_j$ is a linear combination of the submatrix columns $R_i - R_j$ of D_{ij}.

Proof: T_i and T_j can be seen as two vectors v_i and v_J expressed in TCP coordinate system:

$$\begin{aligned} T_i &= \mathrm{TCP_w} + v_i \\ T_j &= \mathrm{TCP_w} + v_j \end{aligned} \qquad (4.9)$$

Where $\mathrm{TCP_w}$ is the TCP expressed in robot base coordinate system.

$$T_i - T_j = \mathrm{TCP_w} + v_i - (\mathrm{TCP_w} + v_j) = v_i - v_j \qquad (4.10)$$

Since $v_j = R_{ij} \cdot v_i$, where R_{ij} is the correlated rotation matrix of R_i and R_j, then $v_i - v_j$ is a difference vector of R_{ij}. This means that also $T_i - T_j$ is a difference vector of R_{ij} then it is contained into the rotation plane of R_{ij} as the columns of $R_i - R_j$. Then if $R_i \neq R_j$, $R_i - R_j$ has rank equal to two then $T_i - T_j$ can be expressed as a linear combination of $R_i - R_j$.

Qed

Then if $R_i \neq R_j$, $R_i - R_j$ has rank equal to two, $T_i - T_j$ is a linear combination of $R_i - R_j$ and since the last row D_{ij} (4.4) is zero then D_{ij} has rank equal to two.

This means that two flange poses are not enough to solve the system of equations $D_{ij} \cdot t_{tcp} = 0$, at least another pose F_k is needed.

$$\begin{bmatrix} F_i - F_j \\ F_i - F_k \end{bmatrix} \cdot t_{tcp} = \begin{bmatrix} R_i - R_j & T_i - T_j \\ R_i - R_k & T_i - T_k \end{bmatrix} \cdot t_{tcp} = 0 \qquad (4.11)$$

If $R_i \neq R_j$ and $R_j \neq R_k$ and if $\hat{e}_{ij} \neq \hat{e}_{ik}$, where $\hat{e}$ is the *axis of rotation* of the respective *correlated rotation matrix*, then the *difference vectors* represented by the difference matrix $R_i - R_j$ and $R_i - R_k$ in (4.11) are contained in at least two different r*otation plane.* For each r*otation* plane, as shown above, there are at least two difference *vectors* linearly independent. Then the matrix defined in (4.11) has three

difference vectors linearly independent or, in other words, the system of equations has unique solution.

4.3 CAMERA BASE CALIBRATION

A binding to solve equation (4.11) is that the TCP must be in the same point for each flange pose, but using a camera the TCP can be constrained to lie only on a line.

The optical axis l_o in the three-dimensional space is a line whose projection corresponds to the image center. Then, for each flange pose, two coordinates can be constrained with fairly accuracy moving the flange on a plane parallel to the camera plane in such a way to constraint the TCP to be on the image center.

Let W_π the camera base that is a coordinate system with the origin corresponding to the center of the camera image plane, with the plane xy_π aligned with the camera plane xy_c and the z-axis aligned with the camera optical axis:

$$W_\pi = \begin{bmatrix} R_\pi & T_\pi \\ 0 & 1 \end{bmatrix} = W_O \begin{bmatrix} R_{o\pi} & T_{o\pi} \\ 0 & 1 \end{bmatrix} \qquad (4.12)$$

Where $R_{o\pi} \in \mathbb{R}^{3x3}$ and $T_{o\pi} \in \mathbb{R}^3$ are the rotation matrix and the translation vector from the robot base coordinate system to the camera plane.

Since the purpose is to move the tool with respect to the current position (then without setting absolute positions), only the rotation matrix R_π is required because it allows to define the direction $d_{\pi_i} \in \mathbb{R}^3$ with respect to W_π, along which to move the flange to constraint the TCP projection to be on the image center keeping the flange on the plane xy_π. Let $d_{c_i} \in \mathbb{R}^2$ the direction given by the vector that connect the TCP projection (x_i, y_i) on the image plane to the image center (x_c, y_c), then:

$$d_{\pi i} = \frac{(x_i - x_c\ , y_i - y_c, 0)}{|(x_i - x_c\ , y_i - y_c)|} \qquad (4.13)$$

Note that last value of (4.13) is zero since the coordinate system W_π has the plane xy parallel to camera plane. Then the camera base (4.12) allows to move the

flange along direction $d_{\pi i}$ in such a way to constraint the TCP projection to be on the image center without changing the z-value.

There are several methods to find the camera rotation matrix R_π, two of which are the following:

4.3.1 Kuka base calibration

Kuka provides a procedure to define a new reference system called *base*. This procedure requires the identification of three points $p_1, p_2, p_3 \in \mathbb{R}^3$ moving the robot using the manual control system:

- p_1: origin of the new coordinate system
- p_2: point on x-axis of the new coordinate system
- p_3: point on xy plane of the new coordinate system

The z-axis direction is defined according to *left-hand rule*.

Note that to define the origin with fairly accuracy a pointed tool already calibrated have to be used. Here only the rotation R_π is useful then the *base* can be defined moving a pointed tool on the three points but considering the corresponding flange position to set the values p_1, p_2 and p_3, making sure that during the movements the flange doesn't change its orientation. The resulting rotation matrix is the same since the flange position can be seen as a constant translation of the tool tip. Then the reference system computed will have another origin but the same orientation.

4.3.2 Base calibration using camera images

Another way to compute the rotation matrix R_π is to move the tool on different positions contained into a plane of the 3D space expressed in robot base coordinate system and recording the TCP position found with the camera.

Let (X_w, Y_w, Z_w) a flange position expressed in the robot base coordinate system, (u, v) the pixel coordinate representing the TCP, $R_\pi \in \mathbb{R}^{3x3}$ a rotation matrix, α and β the scaling factors from pixel to robot coordinate system unit along u and v directions then:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = R_\pi \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (4.14)$$

Equation (4.14) represents the relation between flange positions and corresponding TCP found on the image plane. Then the camera rotation matrix can be found recording these planar correspondences and then computing R_π.

4.4 TCP POSITIONING WITH CAMERA

Let $p_{\pi i} \in \mathbb{R}^3$ the current flange position expressed in camera base frame W_π (4.12), (x_i, y_i) the corresponding TCP projection on the camera plane and (x_c, y_c) the image plane center. The flange position expressed in W_π such that the TCP projection on the camera image plane corresponds to the image center is:

$$p_{\pi i-c} = p_{\pi i} + \begin{bmatrix} \alpha(x_c - x_i) \\ \beta(y_c - y_i) \\ 0 \end{bmatrix} \qquad (4.15)$$

Where α and β are the scaling factors from pixel to robot coordinate system unit along x and y directions.

The scaling factor α and β can be defined considering a flange position (f_x, f_x, f_z) expressed in camera coordinate system W_π, the corresponding TCP projection on the camera plane (x_1, y_1), a second flange position $(f_x + d, f_y + d, f_z)$ (that is a translation of $p_{\pi 1}$ along the plane xy) and the corresponding TCP projection on the camera plane (x_2, y_2):

$$\alpha = \frac{d}{x_2 - x_1}$$
$$\beta = \frac{d}{y_2 - y_1} \qquad (4.16)$$

Note that the plane xy is parallel to the camera image plane.

The problem of this method is that changing the flange depth with respect to to W_π (z-component), will change scale factor too. To overcome the problem the following algorithm has been used to move the TCP in such a way that its projection on the image plane is equal to the image plane center:

1. Get flange position $p_{\pi 1}$ and the corresponding TCP projection on the camera plane (x_1, y_1)
2. $d_x = (x_c - x_1)$, $d_y = (y_c - y_1)$
3. Set the flange position $p_{\pi 2} = p_{\pi 1} + \begin{bmatrix} \alpha d_x \\ \beta d_y \\ 0 \end{bmatrix}$ and get the corresponding TCP projection on the camera plane (x_2, y_2)
4. If (x_2, y_2) corresponds to the image center then stop.
5. If $(x_c - x_2) \geq \frac{d_x}{2}$ then $\alpha = 2\alpha$
6. If $(x_c - x_2) \leq -\frac{d_x}{2}$ then $\alpha = \frac{\alpha}{2}$
7. If $(y_c - y_2) \geq \frac{d_y}{2}$ then $\beta = 2\beta$
8. If $(y_c - y_2) \leq -\frac{d_y}{2}$ then $\beta = \frac{\beta}{2}$
9. Set $p_{\pi 1} = p_{\pi 2}$, and $(x_1, y_1) = (x_2, y_2)$
10. repeat from step 2

Algorithm 4.1 – TCP positioning with camera

Algorithm 4.1 changes dynamically the scale factor if the flange movements to position the TCP are too high or too low.

4.5 TCP COMPUTATION

Regardless of the method used to compute TCP translation vector, there are two main problems:

1. During early stages no information are known about the tool, then to change a flange pose keeping the TCP detectable by the camera, only small flange rotation are suitable. The maximum rotation suitable for this purpose is related to the tool geometry.
2. As shown above at least three flange poses are required to solve the system of equations (4.11) and the rotation axis between flange poses must be distinct. Assuming that first rotation has the rotation axis equal to the z-axis with respect to the reference system W_π (4.12), using Algorithm 4.1 the TCP can be positioned on the same three-dimensional point of the first flange pose. Now the next flange pose must have a different rotation axis,

this means, applying Algorithm 4.1 to move the TCP, it's final position will have a different z-coordinate, then an approximation of TCP can be computed using (4.11).

The first idea was to estimate a solution with the method of the *least squares* that compute a result minimizing the error of each TCP position related to the flange poses used. Studying the detail of this approach, to overcome the problem of the third TCP position that have a different z-coordinate, I found another way to solve the problem without approximations from a theoretical point of view using three flange poses.

4.5.1 Least squares method

The least squares method with normal equations A.1 can be used to compute the TCP:

$$T_{tcp} = (A^T A)^{-1} A^T y \qquad (4.17)$$

Where A is the submatrix of (4.11) containing the difference $R_i - R_j$ and $R_i - R_k$ and y represents the last column of (6.1) containing the difference of the translation vectors with opposite sign:

$$Ax - y = \begin{bmatrix} R_i - R_j \\ R_i - R_k \end{bmatrix} T_{tcp} - \begin{bmatrix} T_j - T_i \\ T_k - T_i \end{bmatrix} \qquad (4.18)$$

As shown above, with $R_j = R_z R_i$ and $R_k = R_x R_i$, A has rank three then even $A^T A$ has rank three. This means $A^T A$ is invertible then the system of equations (4.11) can be solved and has unique solution.

Other flange pose could be used to add equations at (4.18), but positioning the flange after a rotation with Algorithm 4.1 there is no guarantee that the TCP will be on the same three-dimensional position. Then the TCP computed will be an approximation.

4.5.2 Analytic solution

Let $TCP_{img-c} \in \mathbb{Z}^2$ the image center point, $TCP_{\pi-c} \in \mathbb{R}^3$ the point used to constraint the TCP expressed in coordinate system W_π whose projection on the image plane

must be equal to TCP_{img-c} and $F_{1\pi}$ the representation of the first flange position with the TCP placed on the point $P_{\pi 1} = TCP_{\pi-c}$ with a rotation matrix $R_{1\pi}$ and a translation vector $T_{1\pi}$.

A rotation R_z around the z-axis applied on $F_{1\pi}$ moves the TCP to a new 3D position without changing the z-value expressed in coordinate syestem W_π. Now using Algorithm 4.1 the flange can be moved along x and y axis in such a way to place the TCP detected with the camera on the image point TCP_{img-c}. The new TCP position $P_{\pi 2} \in \mathbb{R}^3$ will be equal to $P_{\pi 1}$, let $F_{2\pi}$ the corresponding flange pose with rotation matrix $R_{2\pi} = R_z \cdot R_{1\pi}$ and translation vector $T_{2\pi}$.

Note that using the image center point as TCP position constraint, independently in which z-coordinate is placed the TCP, if its projection on the image plane is equal to TCP_{img-c} then x-coordinate and y-coordinate will be equal to those of $TCP_{\pi-c}$.

To found a solution of (4.11), the next flange rotation has to have a different rotation-axis and the TCP has to be placed on $TCP_{\pi-c}$. But using another rotation axis the 3D TCP z-value will be not equal to the z-value of $P_{\pi 1}$ and $P_{\pi 2}$ and the information obtained from camera are not enough to move the flange on a third pose $F_{3\pi}$ such that the related TCP point $P_{\pi 3} \in \mathbb{R}^3$ is equal to $P_{\pi 1}$ and $P_{\pi 2}$, only the x-component and the y-component will be equal. This means that the z-component of the translation vector $T_{3\pi}$ of $F_{3\pi}$ will be affected by error. The following method overcome this problem leading to a solution without errors from a theoretical point of view:

Consider now only the first and the second flange poses:

$$P_{\pi 1} = P_{\pi 2} \Rightarrow [F_{1\pi} - F_{2\pi}] \cdot t_{tcp} = [R_{1\pi} - R_z \cdot R_{1\pi} \quad T_{1\pi} - T_{2\pi}] \cdot t_{tcp} = 0 \qquad (4.19)$$

Where $t_{tcp} = \left(T_{tcp}, 1\right)^T$ is the homogeneous vector representing the TCP expressed in robot flange coordinate system, then (4.19) can be rewritten as follows:

$$[R_{1\pi} - R_z \cdot R_{1\pi}]T_{tcp} + T_{1\pi} - T_{2\pi} = 0 \qquad (4.20)$$

As shown in ***Statement 4.1*** $[R_{1\pi} - R_z \cdot R_{1\pi}]$ has rank equal to two then a solution of $T_{tcp} = (T_{tcp_x}, T_{tcp_y}, T_{tcp_z})$ cannot be found, however a partial solution can be computed.

If $[R_{1\pi} - R_z \cdot R_{1\pi}]$ would have the last column and the last row equal to zero then T_{tcp_x} and T_{tcp_y} could be computed using the first two rows of the system of equations (4.20) but $R_{1\pi}$ could be any rotation matrix.

***Statement 4.3*:**

$$[R_{1\pi} - R_z \cdot R_{1\pi}]T_{tcp} + T_{1\pi} - T_{2\pi} = 0 \quad (4.21)$$

$$\Leftrightarrow$$

$$[I - R_z] \cdot T'_{tcp} + T_{1\pi} - T_{2\pi} = 0 \quad (4.22)$$

Where $T'_{tcp} = R_{1\pi} \cdot T_{tcp}$.

Proof:

Let $A, B, C \in \mathbb{R}^{3x3}$ and $I \in \mathbb{R}^{3x3}$ the identity matrix then:

$$(A + B)C = AC + BC \quad (4.23)$$

$$(AB)C = A(BC) \quad (4.24)$$

$$IA = AI = A \quad (4.25)$$

$R_{1\pi}$ is a rotation matrix then it is invertible:

$$R_{1\pi} \cdot R_{1\pi}^{-1} = I \quad (4.26)$$

Exploiting properties (4.25) and (4.26), (4.21) can be rewritten as:

$$[R_{1\pi} - R_z \cdot R_{1\pi}]R_{1\pi}^{-1} \cdot R_{1\pi} \cdot T_{tcp} + T_{1\pi} - T_{2\pi} = 0 \quad (4.27)$$

Now with (4.23) and (4.24), (4.27) can be rewritten as:

$$[R_{1\pi} \cdot R_{1\pi}^{-1} - R_z \cdot R_{1\pi} \cdot R_{1\pi}^{-1}]R_{1\pi} \cdot T_{tcp} + T_{1\pi} - T_{2\pi} =$$
$$[I - R_z] \cdot T'_{tcp} + T_{1\pi} - T_{2\pi} = 0 \quad (4.28)$$

Where $T'_{tcp} = R_{1\pi} \cdot T_{tcp}$.

Qed

There is a bi-unique relation between T'_{tcp} and T_{tcp}, then finding the solution of T'_{tcp}, even T_{tcp} can be computed.

$$T_{tcp} = R_{1\pi}^{-1} \cdot T'_{tcp} \qquad (4.29)$$

$[I - R_z]$ of (4.22) has the last row and the last column equal to zero:

$$[I - R_z] \cdot T'_{tcp} + T_{1\pi} - T_{2\pi} =$$

$$\begin{bmatrix} 1-\cos\alpha_z & \sin\alpha_z & 0 \\ -\sin\alpha_z & 1-\cos\alpha_z & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T'_{tcp_x} \\ T'_{tcp_y} \\ T'_{tcp_z} \end{bmatrix} + \begin{bmatrix} T_{1\pi_x} - T_{2\pi_x} \\ T_{1\pi_y} - T_{2\pi_y} \\ T_{1\pi_z} - T_{2\pi_z} \end{bmatrix} = 0 \qquad (4.30)$$

Where $\alpha_z \in (0, \pi]$ is the rotation angle around the z-axis and $T'_{tcp} = (T'_{tcp_x}, T'_{tcp_y}, T'_{tcp_z})$.

Then from (4.30), the following system of equations can be extracted:

$$\begin{bmatrix} 1-\cos\alpha_z & \sin\alpha_z \\ -\sin\alpha_z & 1-\cos\alpha_z \end{bmatrix} \begin{bmatrix} T'_{tcp_x} \\ T'_{tcp_y} \end{bmatrix} + \begin{bmatrix} T_{1\pi_x} - T_{2\pi_x} \\ T_{1\pi_y} - T_{2\pi_y} \end{bmatrix} = 0 \qquad (4.31)$$

If $\alpha_z \neq 0$, (4.31) can be solved since the matrix determinant is grater then zero:

$$Det \begin{bmatrix} 1-\cos\alpha_z & \sin\alpha_z \\ -\sin\alpha_z & 1-\cos\alpha_z \end{bmatrix} = (1-\cos\alpha_z)^2 + (\sin\alpha_z)^2 = \qquad (4.32)$$

$$1 + \cos^2\alpha_z - 2\cos\alpha_z + \sin^2\alpha = 2(1-\cos\alpha_z) > 0$$

$$\begin{bmatrix} T'_{tcp_x} \\ T'_{tcp_y} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & -\frac{\sin\alpha_z}{1-\cos\alpha_z} \\ \frac{\sin\alpha_z}{1-\cos\alpha_z} & 1 \end{bmatrix} \begin{bmatrix} T_{2\pi_x} - T_{1\pi_x} \\ T_{2\pi_y} - T_{1\pi_y} \end{bmatrix} \qquad (4.33)$$

Now, if the third flange pose $F_{3\pi}$ is a rotation along the x-axis with respect to the first flange pose $F_{1\pi}$, $R_{3\pi} = R_x \cdot R_{1\pi}$, even T'_{tcp_z} can be computed:

Considering the TCP constraint resulting from the first and the third flange pose:

$$[R_{1\pi} - R_x \cdot R_{1\pi}]T_{tcp} + T_{1\pi} - T_{3\pi} = 0 \qquad (4.34)$$

Using ***Statement 4.3***, even (4.34) can be rewritten as follows:

$$[I - R_x] \cdot T'_{tcp} + T_{1\pi} - T_{2\pi} =$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1-\cos\alpha_x & \sin\alpha_x \\ 0 & -\sin\alpha_x & 1-\cos\alpha_x \end{bmatrix} \begin{bmatrix} T'_{tcp_x} \\ T'_{tcp_y} \\ T'_{tcp_z} \end{bmatrix} + \begin{bmatrix} T_{1\pi_x} - T_{3\pi_x} \\ T_{1\pi_y} - T_{3\pi_y} \\ T_{1\pi_z} - T_{3\pi_z} \end{bmatrix} = 0 \qquad (4.35)$$

Where $\alpha_x \in (0, \pi]$ is the rotation angle around the x-axis.

As mentioned before, any rotation with a rotation-axis not equal to z, leads to a $T_{3\pi_z}$ value affected by error, because using Algorithm 4.1 the TCP related to the two flange poses cannot be constrained to lie on the same point. But knowing T'_{tcp_y} from (4.33), the second row of (4.35) can be used to compute T'_{tcp_z}:

$$(1-\cos\alpha_x)T'_{tcp_y} + (\sin\alpha_x)T'_{tcp_z} + T_{1\pi_y} - T_{3\pi_y} = 0 \qquad (4.36)$$

$$T'_{tcp_z} = \frac{T_{3\pi_y} - T_{1\pi_y} - (1-\cos\alpha_x)T'_{tcp_y}}{\sin\alpha_x} \qquad (4.37)$$

Note that in this case α_x must be less then π to find a solution.

The equations (4.37) and (4.33) can be grouped defining the following system of equations:

$$\begin{bmatrix} T'_{tcp_x} \\ T'_{tcp_y} \\ T'_{tcp_z} \end{bmatrix} =$$

$$\begin{bmatrix} \frac{1}{2} & -\frac{\sin\alpha_z}{2(1-\cos\alpha_z)} & 0 \\ \frac{\sin\alpha_z}{2(1-\cos\alpha_z)} & \frac{1}{2} & 0 \\ \frac{-(1-\cos\alpha_x)}{2\sin\alpha_x}\frac{\sin\alpha_z}{(1-\cos\alpha_z)} & \frac{-(1-\cos\alpha_x)}{2\sin\alpha_x} & \frac{1}{\sin\alpha_x} \end{bmatrix} \begin{bmatrix} T_{2\pi_x} - T_{1\pi_x} \\ T_{2\pi_y} - T_{1\pi_y} \\ T_{3\pi_y} - T_{1\pi_y} \end{bmatrix} \quad (4.38)$$

$$\frac{(1-\cos\alpha)}{\sin\alpha} = \frac{2\sin^2\frac{\alpha}{2}}{2\sin\frac{\alpha}{2}\cos\frac{\alpha}{2}} = \tan\frac{\alpha}{2} \quad (4.39)$$

$$\begin{bmatrix} T'_{tcp_x} \\ T'_{tcp_y} \\ T'_{tcp_z} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2\tan\frac{\alpha_z}{2}} & 0 \\ \frac{1}{2\tan\frac{\alpha_z}{2}} & \frac{1}{2} & 0 \\ -\frac{1}{2}\frac{\tan\frac{\alpha_x}{2}}{\tan\frac{\alpha_z}{2}} & -\frac{1}{2}\tan\frac{\alpha_x}{2} & \frac{1}{\sin\alpha_x} \end{bmatrix} \begin{bmatrix} T_{2\pi_x} - T_{1\pi_x} \\ T_{2\pi_y} - T_{1\pi_y} \\ T_{3\pi_y} - T_{1\pi_y} \end{bmatrix} \quad (4.40)$$

All the terms of (4.40) are known then, as shown in ***Statement 4.3*****,** the solution of T_{tcp} can be computed as follows:

$$T_{tcp} = R_{1\pi}^{-1} \cdot T'_{tcp} \quad (4.41)$$

Note that the second rotation is around the x-axis but the same reasoning could be made rotating the flange around the y-axis.

4.6 ERROR ANALYSIS

Because of the translations performed by the robot and the camera discretization, the analytic solution 4.5.2 to compute the TCP has some unavoidable errors.

Suppose that for each translations difference term of equation (4.40) there is an error $\epsilon_i \in (-\epsilon_{max}, \epsilon_{max})$ where $i \in [1,2,3]$ is the corresponding row index:

$$\begin{bmatrix} T'_{tcp_x} + \epsilon_x \\ T'_{tcp_y} + \epsilon_y \\ T'_{tcp_z} + \epsilon_z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2\tan\frac{\alpha_z}{2}} & 0 \\ \frac{1}{2\tan\frac{\alpha_z}{2}} & \frac{1}{2} & 0 \\ -\frac{1}{2}\frac{\tan\frac{\alpha_x}{2}}{\tan\frac{\alpha_z}{2}} & -\frac{1}{2}\tan\frac{\alpha_x}{2} & \frac{1}{\sin\alpha_x} \end{bmatrix} \begin{bmatrix} T_{2\pi_x} - T_{1\pi_x} + \epsilon_1 \\ T_{2\pi_y} - T_{1\pi_y} + \epsilon_2 \\ T_{3\pi_y} - T_{1\pi_y} + \epsilon_3 \end{bmatrix} \quad (4.42)$$

$(\epsilon_x, \epsilon_y, \epsilon_z)$ is the distance vector from the real solution. This error can be separated from equation (4.42):

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2\tan\frac{\alpha_z}{2}} & 0 \\ \frac{1}{2\tan\frac{\alpha_z}{2}} & \frac{1}{2} & 0 \\ -\frac{1}{2}\frac{\tan\frac{\alpha_x}{2}}{\tan\frac{\alpha_z}{2}} & -\frac{1}{2}\tan\frac{\alpha_x}{2} & \frac{1}{\sin\alpha_x} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} \quad (4.43)$$

Matrix on equation (4.43) represents the error amplification of the method used to compute the TCP as a function of the rotation angle α_x and α_z. Note that with these values tending to zero the errors will tend to infinity.

The minimum amplification for each error corresponds to the optimal values $\alpha_z = \pi$ and $\alpha_x = \frac{\pi}{2}$ obtaining the following matrix:

$$\begin{bmatrix}\epsilon_x\\ \epsilon_y\\ \epsilon_z\end{bmatrix} = \begin{bmatrix}\frac{1}{2} & 0 & 0\\ 0 & \frac{1}{2} & 0\\ 0 & -\frac{1}{2} & 1\end{bmatrix}\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} \quad (4.44)$$

Since the rotations are centered on the flange, to keep the TCP detectable by the camera, the rotation angles must be small, like one or two degrees, but moving α_z and α_x away from optimal values, errors will be amplified. Then applying the method described in section 4.5.2 with this kind of rotations, the T_{tcp} computed will be inaccurate, however the result can be used to set a new rotation center to compute the T_{tcp} a second time using widest rotations and keeping the tool detectable by the camera.

If the errors amplification were still too high, the error ϵ_z could be reduced with another rotation: after the second rotation around the x-axis with $\alpha_x = \frac{\pi}{2}$, the plane xz will be parallel to the camera image plane. Now the same method used to compute T'_{tcp_x} and T'_{tcp_y} (4.30) can be used to compute T'_{tcp_x} and T'_{tcp_z} replacing T'_{tcp_y} with T'_{tcp_z}, performing a the rotation around the y-axis instead of z-axis and moving the flange with Algorithm 4.1 along z-axis instead of y-axis. Then replacing only the resulting row related to T'_{tcp_z}, (4.42) can be rewrite as follows:

$$\begin{bmatrix}T'_{tcp_x}+\epsilon_x\\ T'_{tcp_y}+\epsilon_y\\ T'_{tcp_z}+\epsilon_z\end{bmatrix} = \begin{bmatrix}\frac{1}{2} & -\frac{1}{2\tan\frac{\alpha_z}{2}} & 0\\ \frac{1}{2\tan\frac{\alpha_z}{2}} & \frac{1}{2} & 0\\ \frac{1}{2\tan\frac{\alpha_y}{2}} & 0 & \frac{1}{2}\end{bmatrix}\begin{bmatrix}T_{2\pi_x}-T_{1\pi_x}+\epsilon_1\\ T_{2\pi_y}-T_{1\pi_y}+\epsilon_2\\ T_{4\pi_z}-T_{1\pi_z}+\epsilon_3\end{bmatrix} \quad (4.45)$$

Where α_y is the rotation around the y-axis.

In this case with $\alpha_y = \pi$ the distance vector from the real solution would become:

$$\begin{bmatrix}\epsilon_x\\ \epsilon_y\\ \epsilon_z\end{bmatrix}=\begin{bmatrix}\frac{1}{2} & 0 & 0\\ 0 & \frac{1}{2} & 0\\ 0 & 0 & \frac{1}{2}\end{bmatrix}\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} \quad (4.46)$$

However there could be not enough space to perform this kind of rotations. Then the first method (only two rotation around z-axis and x-axis without constraints on rotations angles) could be more suitable for the purpose.

For instance $\alpha_z = \alpha_x = \frac{\pi}{4}$ on equation (4.42) could be the right compromise between precision and performance:

$$\begin{bmatrix}\epsilon_x\\ \epsilon_y\\ \epsilon_z\end{bmatrix}=\begin{bmatrix}\frac{1}{2} & -\frac{\sqrt{2}}{4-2\sqrt{2}} & 0\\ \frac{\sqrt{2}}{4-2\sqrt{2}} & \frac{1}{2} & 0\\ -\frac{1}{2} & -\frac{2-\sqrt{2}}{2\sqrt{2}} & \frac{\sqrt{2}}{2}\end{bmatrix}\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} \quad (4.47)$$

$$\begin{bmatrix}\epsilon_x\\ \epsilon_y\\ \epsilon_z\end{bmatrix}=\begin{bmatrix}0{,}5 & -1{,}21 & 0\\ 1{,}21 & 0{,}5 & 0\\ -0{,}5 & -0{,}21 & 1{,}41\end{bmatrix}\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix} \quad (4.48)$$

In any case, if the errors were too high and there were physics limitations to perform wide rotations, the least squares method could be applied to find a better solution using a constraints matrix built concatenating the system of equations given by the Analytic solution 4.5.2 using different initial positions.

5 RESULTS

5.1 IMAGE RESOLUTION

The following results have been obtained keeping the tool to roughly $30\ cm$ in front of the camera where one pixel corresponds to approximately $0{,}021\ mm$ along both x-axis and y-axis. These values has been computed as follows:

$$d_x = \frac{1}{n-1}\sum_{i=1}^{n-1}\left|\frac{x_{w_i} - x_{w_{i+1}}}{x_{c_i} - x_{c_{i+1}}}\right|, \qquad d_y = \frac{1}{n-1}\sum_{i=1}^{n-1}\left|\frac{x_{w_i} - x_{w_{i+1}}}{x_{c_i} - x_{c_{i+1}}}\right| \qquad (5.1)$$

Where $x_{w_i} \in \mathbb{R}$ is the x coordinate of a random flange position expressed in camera base coordinate system (4.12), $x_{c_i} \in \mathbb{N}$ is x coordinate of the corresponding pixel representing the TCP projected on the camera image plane and $n = 30$ is the number of random flange poses considered.

5.2 TCP ACCURACY

The accuracy of method 4.5.2 can be computed estimating the TCP translation several times using different initial positions and then evaluating the distance between the TCP found and the real TCP.

The rotation angles used are $\alpha_z = \frac{\pi}{8}$ around the z-axis and $\alpha_x = \frac{\pi}{8}$ around the x-axis.

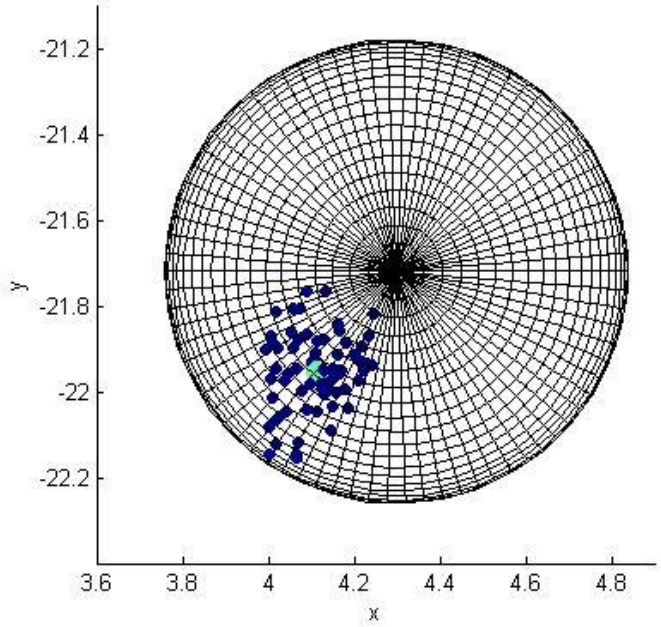


Figure 5.1 – Accuracy x y axis

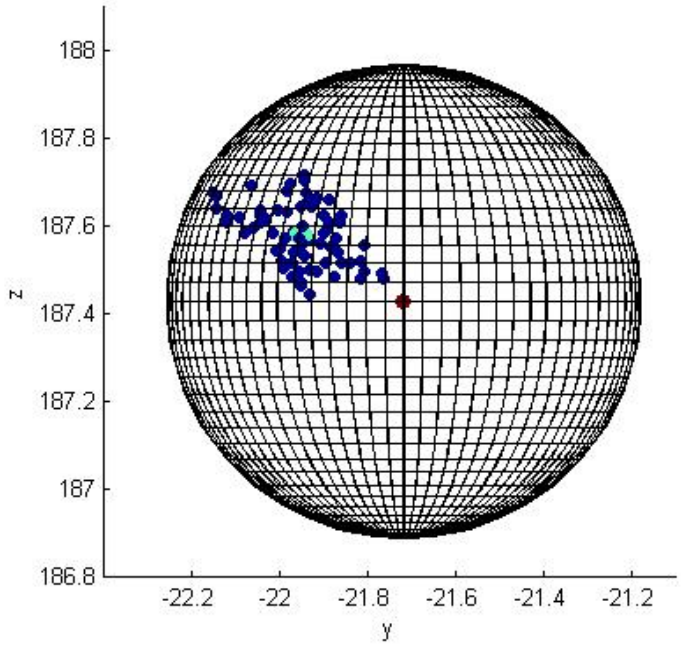


Figure 5.2 – Accuracy y z axis

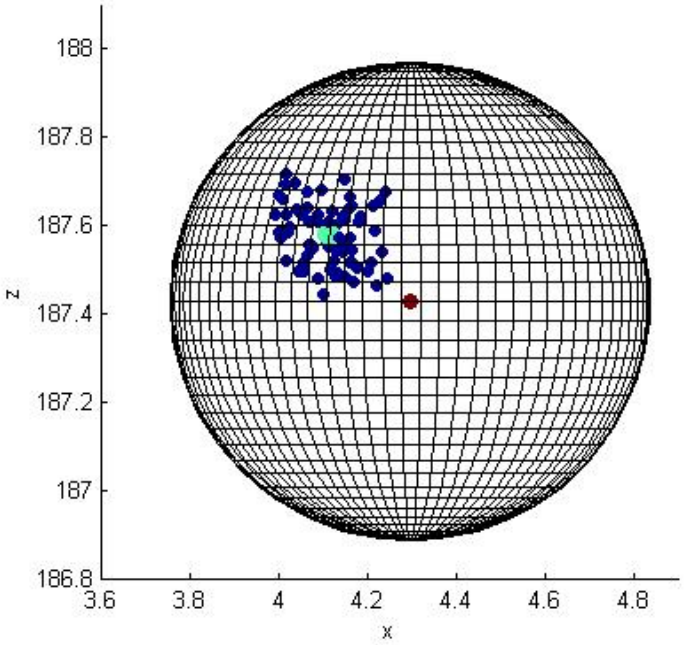


Figure 5.3 – Accuracy x z axis

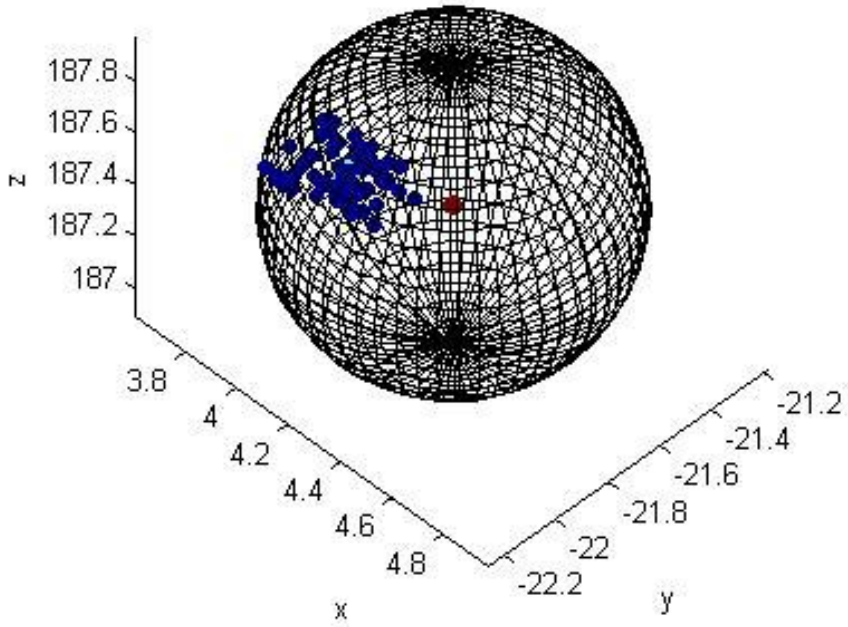


Figure 5.4 – Accuracy x y z axis

Figure 5.1, Figure 5.2, Figure 5.3 and Figure 5.4 legend:

- Blue: TCP found with method 4.5.2
- Green: mean of all the tool center points found with method 4.5.2. $TCP_{mean} = (4.11, -21.95, 187.58)$.
- Red: TCP found using the manual method explained on section 1.1. $TCP_{manual} = (4.29, -21.72, 187.43)$.
- Sphere radius: maximum distance estimated between TCP_{manual} and the real TCP. $r = 0.54\, mm$.

TCP_{mean} can reasonably considered as the real TCP since it is the mean of all the TCP computed.

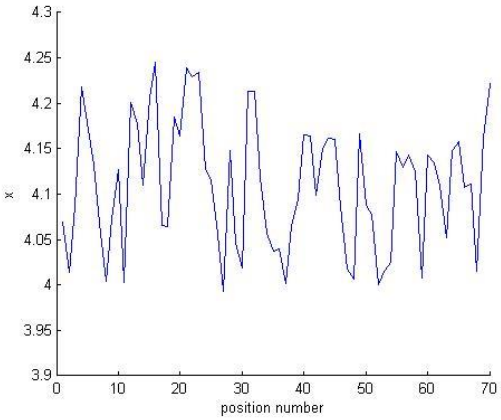


Figure 5.5 – Accuracy x axis

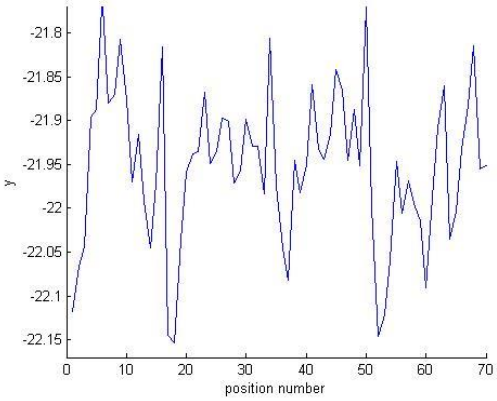


Figure 5.6 – Accuracy y axis

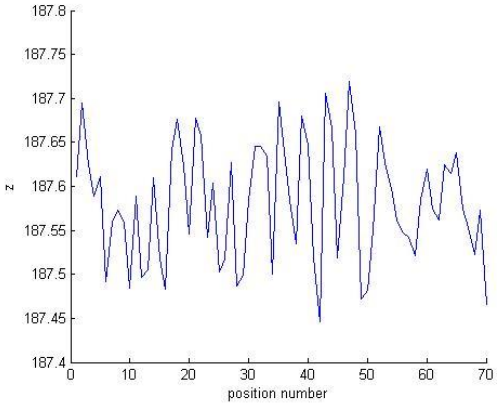


Figure 5.7 – Accuracy z axis

Figure 5.5, Figure 5.6 and Figure 5.7 represent the x, y and z values estimated for each accuracy test done.

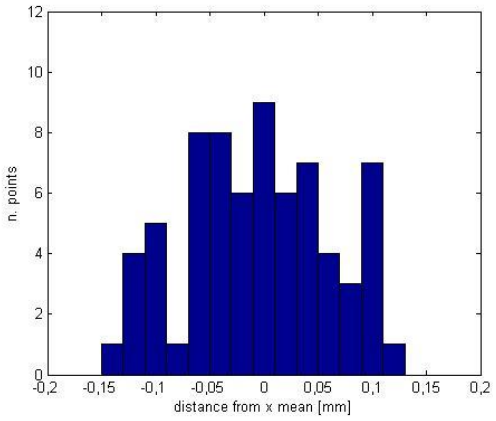


Figure 5.8 – Accuracy - Distance of each TCP_x found from TCP_{mean_x}

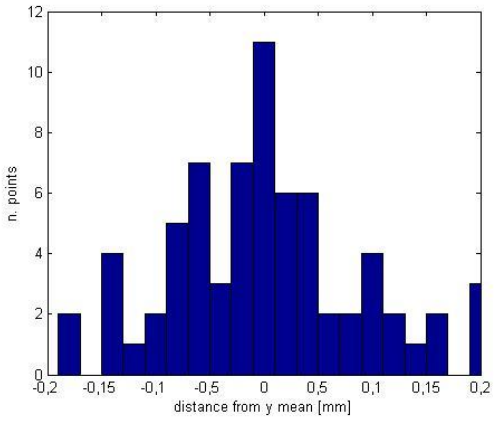


Figure 5.9 – Accuracy - Distance of each TCP_y found from TCP_{mean_y}

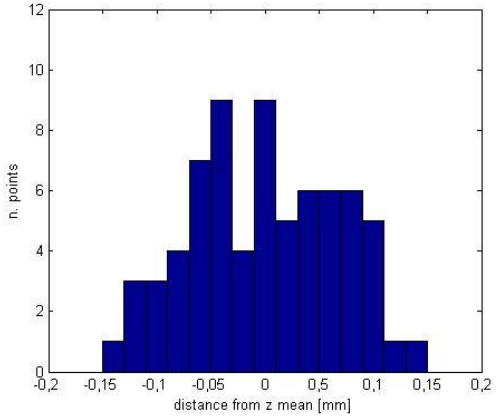


Figure 5.10 – Accuracy - Distance of each TCP_z found from TCP_{mean_z}

Figure 5.8, Figure 5.9 and Figure 5.10 represent the histogram of the distances form the mean TCP found along each axis. The results can be modelled by three normal distributions:

$$X_a = TCP_x - E(TCP_x) \sim N\left(0, {\sigma_a}_x^2\right)$$

$$Y_a = TCP_y - E\left(TCP_y\right) \sim N\left(0, {\sigma_a}_y^2\right) \qquad (5.2)$$

$$Z_a = TCP_z - E(TCP_z) \sim N(0, {\sigma_a}_z^2)$$

Where the mean of normal distributions are $E(X) = E(Y) = E(Z) = 0$ since for any random variable R, $E\big(R - E(R)\big) = 0$.

Value	Max distance from $\boldsymbol{TCP_{mean}}$ [mm]	Standard deviation with respect to $\boldsymbol{TCP_{mean}}$ [mm]
$\boldsymbol{TCP_x}$	0.1346	$\sigma_{a_x} = 0.0586$
$\boldsymbol{TCP_y}$	0.2003	$\sigma_{a_y} = 0.0889$
$\boldsymbol{TCP_z}$	0.1381	$\sigma_{a_z} = 0.0675$

Table 5.1 – TCP Accuracy results of each axis

Note that the standard deviations on Table 5.1 are slightly different because the errors along the camera optical axis are more amplified in respect to the other two axis (see section 4.6 for details).

The absolute distance of each TCP_i found from TCP_{mean} can be computed as follows:

$$\sqrt{\left(TCP_{i_x} - TCP_{mean_x}\right)^2 + \left(TCP_{i_y} - TCP_{mean_y}\right)^2 + \left(TCP_{i_z} - TCP_{mean_z}\right)^2} \quad (5.3)$$

Then the random variable that approximates the error can be represented by the squares root of the sum of the squares of the normalized random variables X_a, Y_a and Z_a (5.2):

$$A_a \sim \sqrt{X_a^2 + Y_a^2 + Z_a^2} \quad (5.4)$$

To simplify the accuracy representation, X_a , Y_a and Z_a should be mutually independent and should have the same standard deviation.

The mutual independence of X , Y and Z can be evaluated computing the *correlation index*:

$$\rho_{a_{xy}} = \frac{\sigma_{a_{xy}}}{\sigma_{a_x}\sigma_{a_y}} = 0.2077$$

$$\rho_{a_{zy}} = \frac{\sigma_{a_{zy}}}{\sigma_{a_z}\sigma_{a_y}} = -0.2104 \qquad (5.5)$$

$$\rho_{a_{xz}} = \frac{\sigma_{a_{xz}}}{\sigma_{a_x}\sigma_{a_z}} = -0.3412$$

Where $\sigma_{a_{xy}}$, $\sigma_{a_{zy}}$ and $\sigma_{a_{xz}}$ represent the covariances. The *correlation index* ranges is $[-1,1]$. When the absolute value of the *correlation index* is close zero, the random variables are independent, then in this case we can assume that X_a, Y_a and Z_a are mutual independent.

Now suppose $\sigma_a = \sigma_{a_x} = \sigma_{a_y} = \sigma_{a_z} = 0.0717$, that is the mean of the standard deviations found during the accuracy test (Table 5.1).

Statement 5.1: If X_a, Y_a and Z_a are three independent normal distributions with zero mean and with the same standard deviation σ_a, then the random variable A_a has a Nakagami distribution with $m = \frac{3}{2}$ and $\Omega = \sigma_a^2$:

$$A_a \sim \sqrt{X_a^2 + Y_a^2 + Z_a^2} \sim Nakagami(\frac{3}{2}, 3\sigma_a^2) \qquad (5.6)$$

Proof

$$X_a \sim N(0,\sigma_a), Y_a \sim N(0,\sigma_a), Z_a \sim N(0,\sigma_a) \Rightarrow$$

$$A \sim \sqrt{\sigma_a^2\frac{X_a^2}{\sigma_a^2} + \sigma_a^2\frac{Y_a^2}{\sigma_a^2} + \sigma_a^2\frac{Z_a^2}{\sigma_a^2}} \sim \sqrt{\sigma_a^2\chi^2 + \sigma_a^2\chi^2 + \sigma_a^2\chi^2} \qquad (5.7)$$

Where χ^2 is the chi-square distribution with one degree of freedom. This random variable is a special case of the Gamma distribution: $\chi^2 = \text{Gamma}(\frac{1}{2}, 2)$. Now using the scaling property of the Gamma distribution, (5.7) can be rewritten as:

$$A \sim \sqrt{\text{Gamma}\left(\frac{1}{2}, 2\sigma_a^2\right) + \text{Gamma}\left(\frac{1}{2}, 2\sigma_a^2\right) + \text{Gamma}\left(\frac{1}{2}, 2\sigma_a^2\right)} \qquad (5.8)$$

Since X_a, Y_a and Z_a are mutual independent with the same scale parameter ($2\sigma_a^2$), the sum of these Gamma distributions can be rewritten as follows:

$$\mathrm{A} \sim \sqrt{\mathrm{Gamma}\left(\frac{3}{2}, 2\sigma_a^2\right)} \qquad (5.9)$$

The square root of a Gamma distribution $\mathrm{Gamma}(k, \theta)$ is a Nakagami distribution $Nakagami(m, \Omega)$ with parameters $m = k$ and $\Omega = m\theta$ then:

$$A \sim Nakagami\left(\frac{3}{2}, 3\sigma_a^2\right) \qquad (5.10)$$

Qed

Then the Nakagami distribution $Nakagami(\frac{3}{2}, 3\sigma_a^2)$ can be used to approximate the accuracy.

$\boldsymbol{Nakagami(\frac{3}{2}, 3\sigma_a^2)}$	
$\boldsymbol{m}$	$\frac{3}{2}$
$\boldsymbol{\Omega}$	0.0154
$\boldsymbol{Mean}$	$\frac{\Gamma\left(m+\frac{1}{2}\right)}{\Gamma(m)}\left(\frac{\Omega}{m}\right)^{\frac{1}{2}} = 0.1143$
$\boldsymbol{Variance}$	$\Omega\left(1-\frac{1}{m}\left(\frac{\Gamma\left(m+\frac{1}{2}\right)}{\Gamma(m)}\right)^2\right) = 0.0073$
$\boldsymbol{Probability\ density\ function}$	$\frac{2\mathrm{m}^{\mathrm{m}}}{\Gamma(\mathrm{m})\Omega^{\mathrm{m}}} x^{2m-1} e^{-\frac{m}{\Omega}x^2} = \frac{2\left(\frac{3}{2}\right)^{\frac{3}{2}}}{\Gamma\left(\frac{3}{2}\right)(3\sigma_a^2)^{\frac{3}{2}}} x^2 e^{-\frac{x^2}{2\sigma_a^2}}$

Table 5.2 – TCP Accuracy represented with Nakagami distribution

Table 5.2 shows some properties of the Nakagami distribution computed to represent the TCP Accuracy. $\Gamma(x)$ is the *gamma function*.

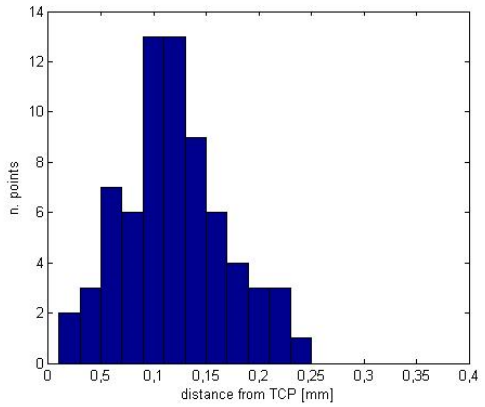


Figure 5.11 – Accuracy - Distances histogram from the mean point TCP_{mean}

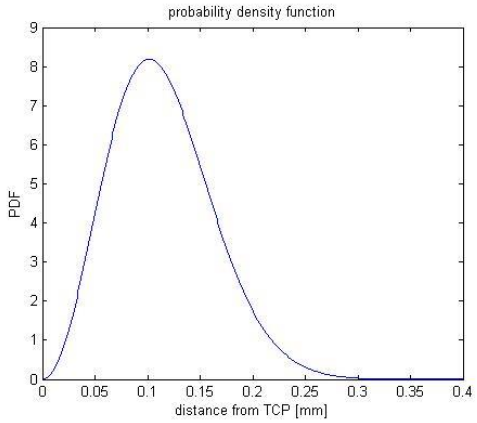


Figure 5.12 – Distance of each TCP_y found from TCP_{mean_y}

Figure 5.11 represents the histogram of the distances between TCP computed and real TCP estimate (5.3), while Figure 5.12 represents the Nakagami distribution computed with mean $0.1143\ mm$ and standard deviation $0.1241\ mm$. Shapes of Figure 5.11 and Figure 5.12 are similar enough to represent the error distribution by this model.

5.3 TCP PRECISION

The TCP precision of method 4.5.2 can be estimated with the same procedure used for estimate the TCP accuracy 5.2, but in this case using the same initial position to compute the TCP translations. The rotation angles used are $\alpha_z = \frac{\pi}{8}$ around the z-axis and $\alpha_x = \frac{\pi}{8}$ around the x-axis.

With 65 samples the following results has been obtained:

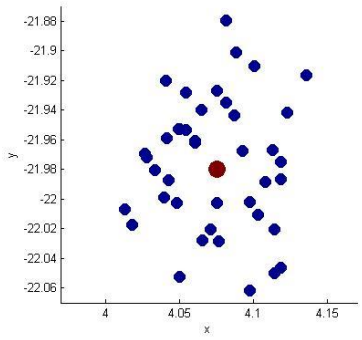


Figure 5.13 – Precision x y axis

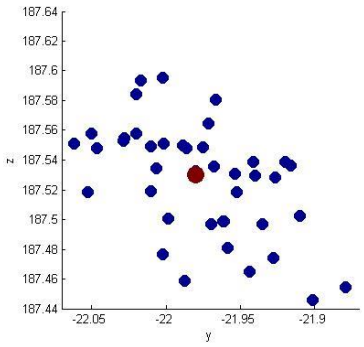


Figure 5.14 – Precision y z axis

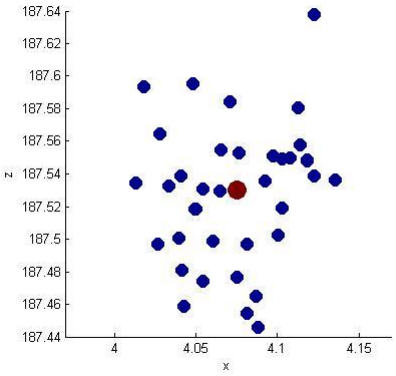


Figure 5.15 – Precision x z axis

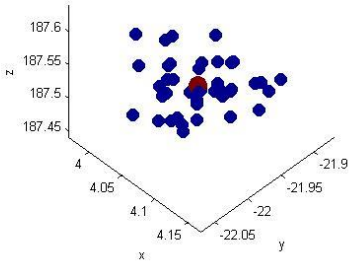


Figure 5.16 – Precision x y z axis

The blue points on figures Figure 5.13, Figure 5.14, Figure 5.15 and Figure 5.16 represent the TCP found while the red point represent the mean.

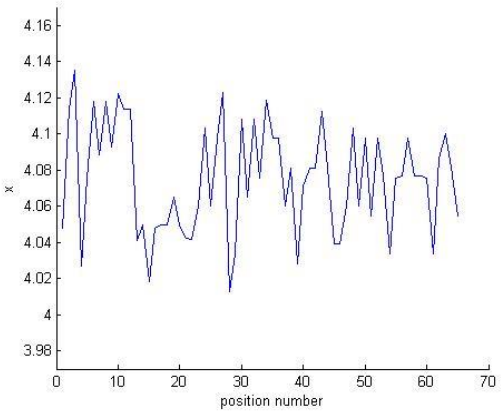


Figure 5.17 – Precision x axis

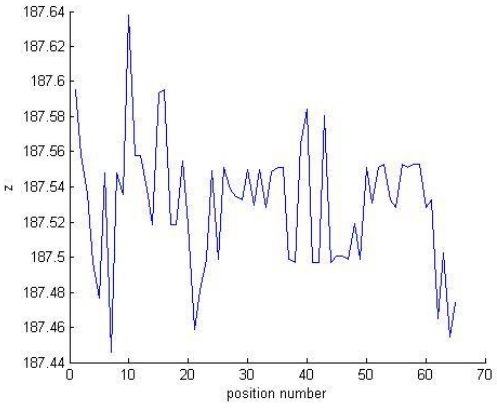


Figure 5.18 – Precision y axis

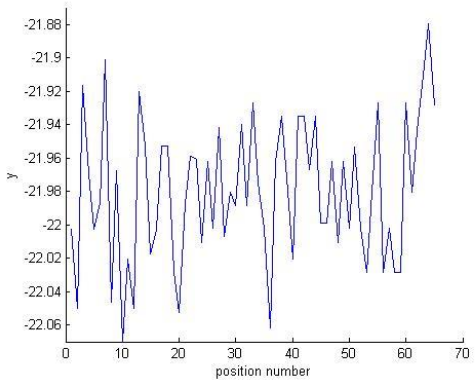


Figure 5.19 – Precision z axis

Figure 5.17, Figure 5.18 and Figure 5.19 represent the x, y and z values estimated for each precision test done.

Value	Max distance from mean [mm]	Standard deviation [mm]
$\boldsymbol{TCP_x}$	0.0622	0.0298
$\boldsymbol{TCP_y}$	0.0929	0.0427
$\boldsymbol{TCP_z}$	0.0842	0.0365

Table 5.3 – TCP precision results

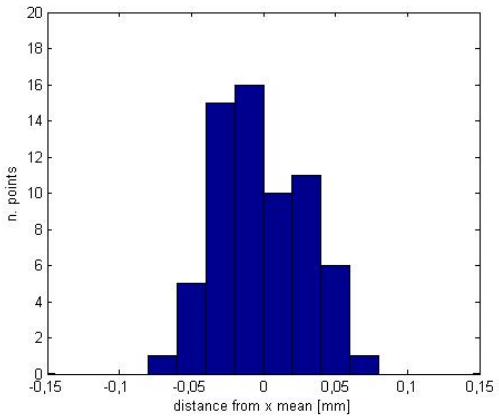


Figure 5.20 – Precision - Distance of each TCP_x found from TCP_{mean_x}

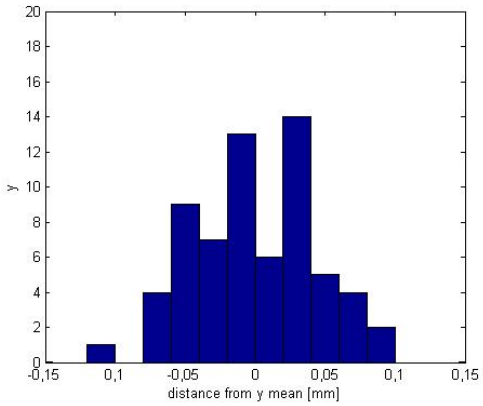


Figure 5.21 – Precision - Distance of each TCP_y found from TCP_{mean_y}

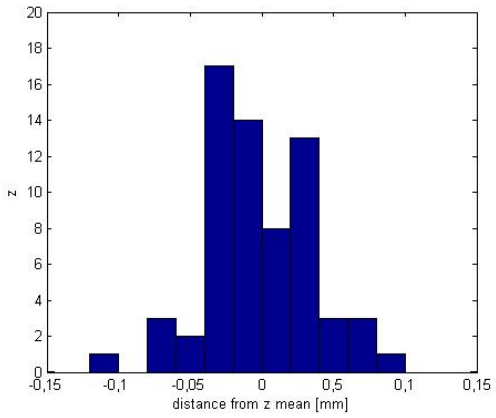


Figure 5.22 – Precision - Distance of each TCP_z found from TCP_{mean_z}

As for the accuracy analysis 5.2, the results can be modelled using three normal distributions:

$$X_p = TCP_x - E(TCP_x) \sim N\left(0, {\sigma_p}_x^2\right)$$
$$Y_p = TCP_y - E\left(TCP_y\right) \sim N\left(0, {\sigma_p}_y^2\right) \qquad (5.11)$$
$$Z_p = TCP_z - E(TCP_z) \sim N(0, {\sigma_p}_z^2)$$

The random variable that approximates the precision can be represented by the square root of the sum of the squares of the normalized random variables X, Y and Z (5.2):

$$A_p \sim \sqrt{X_p^2 + Y_p^2 + Z_p^2} \qquad (5.12)$$

To simplify the precision representation, X_p , Y_p and Z_p should be mutually independent and should have the same standard deviation.

The mutual independence of X_p , Y_p and Z_p can be evaluated computing the *correlation index*

$$\rho_{p_{xy}} = \frac{\sigma_{p_{xy}}}{\sigma_{p_x}\sigma_{p_y}} = -0.1192$$

$$\rho_{p_{zy}} = \frac{\sigma_{p_{zy}}}{\sigma_{p_z}\sigma_{p_y}} = -0.3816 \qquad (5.13)$$

$$\rho_{p_{xz}} = \frac{\sigma_{p_{xz}}}{\sigma_{p_x}\sigma_{p_z}} = -0.2334$$

Where $\sigma_{p_{xy}}$, $\sigma_{p_{zy}}$ and $\sigma_{p_{xz}}$ represent the covariances. Given the values found even in this case we can assume that X_p, Y_p and Z_p are mutual independent.

Suppose now $\sigma_p = \sigma_{p_x} = \sigma_{p_y} = \sigma_{p_z} = 0.0363$ that is the mean of the standard deviations found during the precision test (Table 5.3).

With these conditions, as shown on Statement 5.1, The Nakagami distribution of Table 5.4 can be used to represent the TCP precision.

$\boldsymbol{Nakagami(\frac{3}{2}, 3\sigma_p^2)}$	
$\boldsymbol{m}$	$\frac{3}{2}$
$\boldsymbol{\Omega}$	0.0039
$\boldsymbol{Mean}$	$\frac{\Gamma\left(m+\frac{1}{2}\right)}{\Gamma(m)}\left(\frac{\Omega}{m}\right)^{\frac{1}{2}} = 0.0575$
$\boldsymbol{Variance}$	$\Omega\left(1-\frac{1}{m}\left(\frac{\Gamma\left(m+\frac{1}{2}\right)}{\Gamma(m)}\right)^2\right) = 0.0018$
$\boldsymbol{Probability\ density\ function}$	$\frac{2\mathrm{m}^{\mathrm{m}}}{\Gamma(\mathrm{m})\Omega^{\mathrm{m}}}x^{2m-1}e^{-\frac{m}{\Omega}x^2} = \frac{2\left(\frac{3}{2}\right)^{\frac{3}{2}}}{\Gamma\left(\frac{3}{2}\right)(3\sigma_p^2)^{\frac{3}{2}}}x^2 e^{-\frac{x^2}{2\sigma_{\mathrm{p}}^2}}$

Table 5.4 – TCP Precision represented with Nakagami distribution

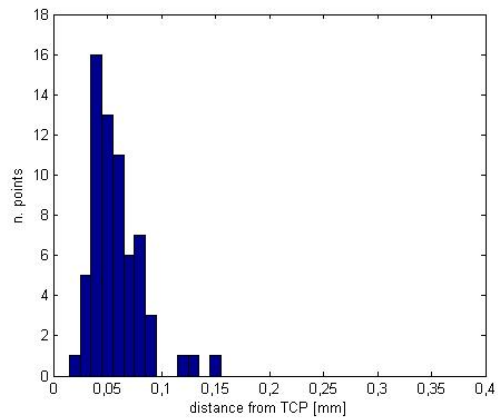


Figure 5.23 – Accuracy - Distances histogram from the mean point TCP_{mean}

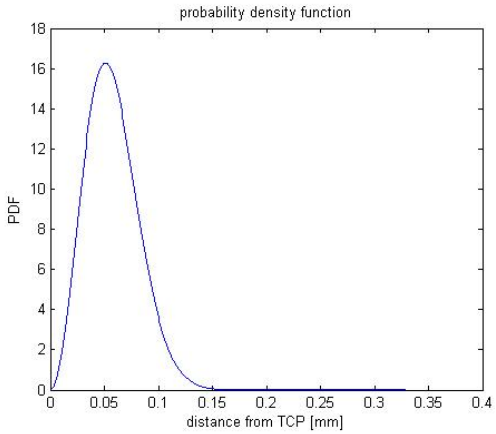


Figure 5.24 – Distance of each TCP_y found from TCP_{mean_y}

Figure 5.23 represents the histogram of the distances between TCP computed during the precision evaluation and TCP estimate (5.3), while Figure 5.24 represents the Nakagami distribution computed with mean $0.0575\,mm$ and standard deviation $0.0424\,mm$. Even in this case the shapes of Figure 5.23 and Figure 5.24 are similar enough to represent the errors distribution by this model.

6 CONCLUSIONS

6.1 RESULTS

The objective of this work was to study a method to compute the tool center point of symmetric tools using computer vision.

Unlike other methods, the solution proposed has several advantages:

- From a theoretical point of view the result is not approximated.
- No initial tool information have to be known.
- It requires three different flange poses that is the minimum number to solve the problem.
- There is no constraints on the rotation amplitude to change flange pose.
- The system of equations to solve is linear.
- The vision algorithms are suitable to any symmetric tool.

To calibrate the TCP an operator has to move the tool on a position detectable by the camera, after this the method works autonomously.

Three different flange poses are required; Initially the rotations to change the flange poses have to be centered on the flange itself since no information regarding the tool are known. Then to keep the TCP detectable by the camera the rotation angles must be small (in this case one degree). The accuracy of this method is proportional to the rotation angles, then with small rotations the TCP computed will be inaccurate, but the result can be used to set a new rotation center to compute the TCP a second time using widest rotations and keeping the tool detectable by the camera.

The accuracy achieved ($0{,}2mm$), computed with rotation angles of $\frac{\pi}{8}$ radians, satisfies the requirements since it improves the performance achievable using the manual tool center point calibration.

Increasing the rotation angles until $\frac{\pi}{2}$ radians, the accuracy would improve but generally the work space may be limited.

6.2 FUTURE WORKS

The method presented doesn't compute the tool orientation. For this purpose the Hough transform applied on the image skeleton could be exploited since it returns a line on the image plane that should represent the projection of the tool direction.

Usually symmetric tools have the z-axis equal to the axis of symmetry. Positioning the tool in such a way that the z-axis is parallel to the vertical axis of the image plane, if the direction found using image processing doesn't match the vertical axis of the image plane, the tool will have a wrong orientation.

Computing the angle between the tool direction found using image processing and the vertical axis of the image plane, the tool can be adjusted rotating around the optical axis by the angle just found and using the TCP as rotation center.

Obviously on the camera plane there is only a bidimensional projection of the tool, then to make sure that the tool has the right direction, a rotation of 90 degree around the vertical axis of the image plane has to be performed. Now adjusting a second time the direction using the new angle between the tool direction found and the vertical axis of the image plane, the tool orientation should be correct.

A. APPENDIX

A.1 LEAST SQUARES METHOD WITH NORMAL EQUATIONS

This is an optimization technique (or regression) that allows to find a function, represented by an optimal curve (or regression curve), which is as close as possible to a set of data. In particular, the function found must be the one that minimizes the sum of squares of the distances between the observed data and the curve that represents the function itself.

The *least squares* method can be written as follow:

$$\min_x \|Ax - y\|^2 \qquad (6.1)$$

The minimization argument of (6.1) can be expanded as follow:

$$\|Ax - y\|^2 = (Ax - y)^T(Ax - y) = x^T A^T Ax - x^T A^T y - y^T Ax + y^T y \qquad (6.2)$$

Note that $x^T A^T y = y^T Ax$ is a scalar then the quantity to minimize becomes:

$$x^T A^T Ax - 2x^T A^T y + y^T y \qquad (6.3)$$

A solution of (6.1) can be obtained differentiating (6.3) with respect to x and placing it equal to 0:

$$A^T Ax - A^T y = 0 \qquad (6.4)$$

The equations (6.4) are called *normal equations*.

$$x = (A^T A)^{-1} A^T y \qquad (6.5)$$

BIBLIOGRAPHY

[1] Milan Sonka, Vaclav Hlavac, and Roger Boyle. Image Processing, Analysis, and Machine Vision. Thomson-Engineering, 2007.

[2] B. Eldridge, S.G. Carey, and L.F. Guymon. Method and system for finding a tool center point for a robot using an external camera, May 7 2009. US Patent App. 12/264,159.

[3] B.A. Bordyn, M.D. Markey, and M.J. Kleemann. External system for robotic accuracy enhancement, September 24 2009. US Patent App. 12/408,958.

[4] L.E. Chiu, S.J. Fu, and G. Zhao. Vision correction method for tool center point of a robot manipulator, February 7 2013. US Patent App. 13/414,919.

[5] G. Li, Y. Genc, S. CHHATPAR, D. Sacco, S. NAIK, A. Gelbman, and R. Barr. Methods and apparatus to calibrate an orientation between a robot gripper and a camera, February 14 2013. WO Patent App. PCT/US2012/050,288.

[6] H.F. Thorne. Tool center point calibration for spot welding guns, June 8 1999. US Patent 5,910,719.

[7] H.F. Thorne. Tool center point calibration apparatus and method, October 10 1995. US Patent 5,457,367.

[8] J.P. Huissoon. Method and device for robot tool frame calibration, March 28 2000. US Patent 6,044,308.

[9] T. Nemmers, D.E. Jenkins, and T.L. Tupper. System and method for setting the tool center point of a robotic tool, March 26 2013. US Patent 8,406,922.

[10] E. Roos. Method and device for calibrating robot measuring stations, manipulators and associated optical measuring devices, September 2 2003. US Patent 6,615,112.

[11] D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, N. Mahmood, J. Saragih, and R. Shilkrot. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing, Limited, 2012. recommended: advanced OpenCV project support/examples inc. iOS and Android examples.

[12] Q. Tang, H. Brantmark, Z. Gan, and T. Brogardh. Robot machining tool position and orientation calibration, September 6 2005. US Patent 6,941,192.

[13] Kuka. *KR C4 compact*, spez kr c4 compact v5 edition, 04 2014.

[14] Kuka. *KR 10 R1100 sixx*, spez kr agilus sixx v11 edition, 11 2014.

[15] Laurenson, Dave (1994). "Nakagami Distribution". Indoor Radio Channel Propagation Modelling by Ray Tracing Techniques. Retrieved 2007-08-04.

[16] Mitra, Rangeet; Mishra, Amit Kumar; Choubisa, Tarun (2012). "Maximum Likelihood Estimate of Parameters of Nakagami-m Distribution". International Conference on Communications, Devices and Intelligent Systems (CODIS), 2012: 9-12.

[17] R. V. Hogg and A. T. Craig (1978) Introduction to Mathematical Statistics, 4th edition. New York: Macmillan.