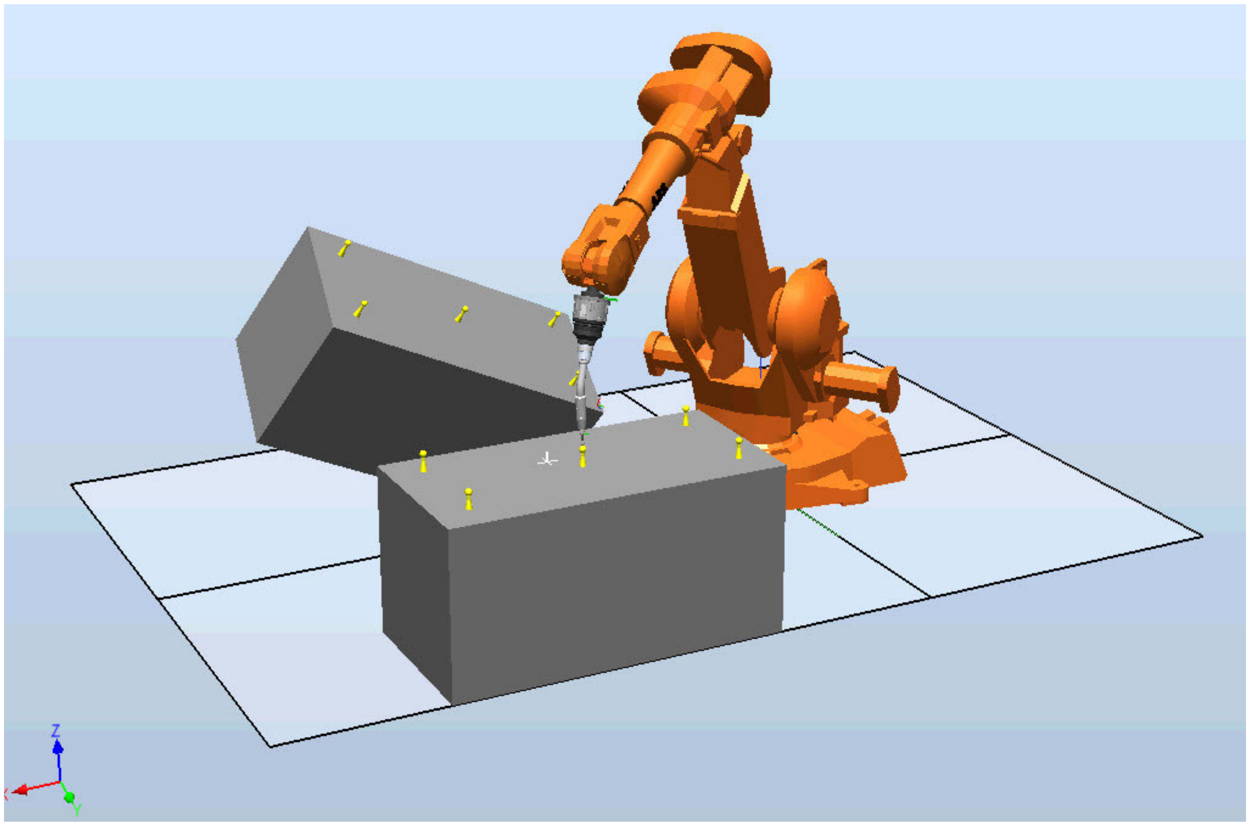
CHALMERS



Method for calibration of off-line generated robot program

Master of Science Thesis

GUSTAV BERGSTRÖM

Department of AUTOMATIC CONTROL
Division of AUTOMATION AND MECHATRONICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2011
Report No. EX099/2011

Table of contents

1 Background

In order to control a robot, a Cartesian coordinates system is used to define the position of the robot arm. This Cartesian coordinate system is more or less unique for each robot. Under the condition that identical position is entered in two different robots, the robots will in most cases not end up in the same physical position. This is explained by the fact that Cartesian coordinate among robots does not have perfect identical origin of the coordinates and orientation. Further on, with respect of the robot foot the coordinates are in most cases not perfect linearly distributed.

1.1 “Teach in” programming

Traditionally by using “teach in” programming method, mechanical imperfections have not been a major issue. This can be explained by the fact that teach in programming is using the actual robot and work piece to obtain the positions. Teach in programming of a robot works like this.

1. The robot is physically moved to the desire position by using a joy-stick or pushbuttons on a handheld device.
2. When desired orientation and position accuracy have been obtained, the position and orientation is stored in the robot program.
3. This process is repeated until every position in the robot program has been taught.

This means in a “taught in” robot program, the stored positions in the controller may not correspond with the physical positions. Most likely there will be deviation errors. However, the deviation errors between values stored in the controller and the physical position values can be excluded as long as the robot repeats with high accuracy to the position.

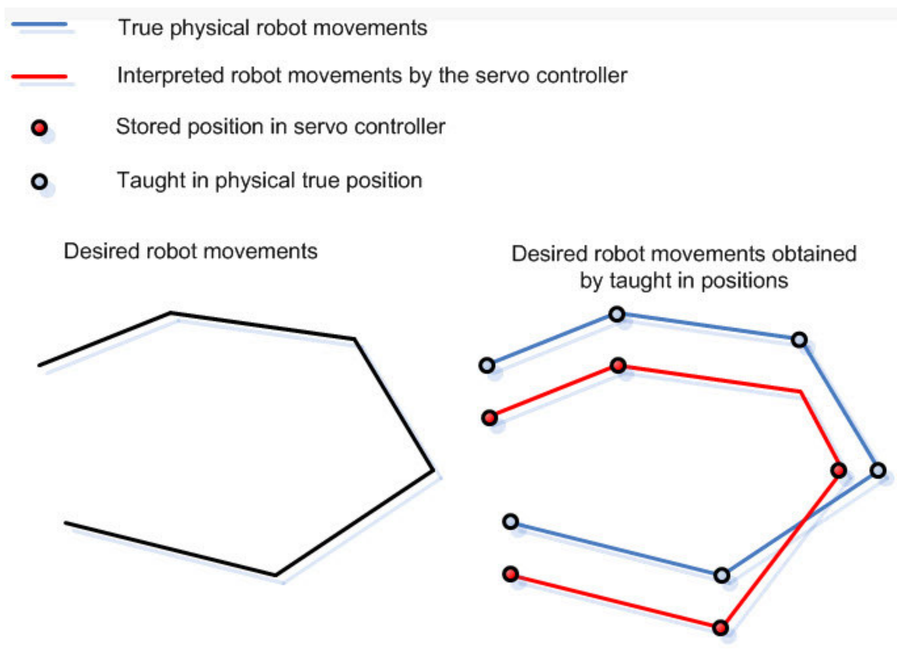
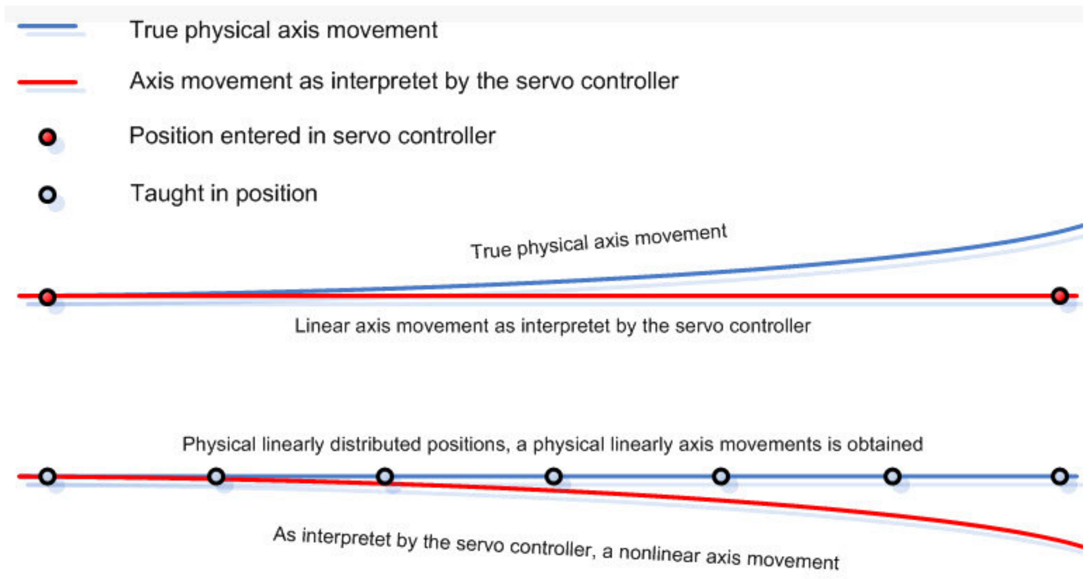

True physical robot movements
Interpreted robot movements by the servo controller
Stored position in servo controller
Taught in physical true position
Desired robot movements
Desired robot movements obtained by taught in positions

Figure 1
This illustration shows the differences between the positions stored in the robot control system and the true physical coordinates. The true physical coordinates can be interpreted as the coordinates which are used for create a robot program offline.

In teach in programming, the nonlinear influences on the axis movements can be reduced by adding way points. By distributing the way points along the desired linearly movement, the robot is forced to pass each one of them. Due to this fact, the nonlinear influences therefore are reduced.



True physical axis movement
Axis movement as interpretet by the servo controller
Position entered in servo controller
Taught in position
True physical axis movement
Linear axis movement as interpretet by the servo controller
Physical linearly distributed positions, a physical linearly axis movements is obtained
As interpretet by the servo controller, a nonlinear axis movement

Figure 2
This illustration shows that a nonlinear robot axis movement can be interpreted by the robot controller as linear. By adding way points a long the desired movement, the nonlinear influences can be reduce in a “taught in program”.

However, a “taught in” program will be unique for the particular robot. If the robot is replaced by another, the stored positions in the robot program may not be applicable. This is explained by the fact that the distribution of the Cartesian positions is unique for each individual robot. Each individual programmed position which requires high accuracy then has to be programmed by physically moving the robot to the desired position.

To summarize, traditionally robot programs have been calibrated based on the actual physical model. Therefore, deviations between the reality and the servo controller have not been an issue for programming of robots. This is explained by the high position repeatability of the robot. In other words, the robot will at high accuracy return to a defined position. However, there are a couple of disadvantages with the “teach in” method.

- The calibration can not be done until the physical robot installation is in place.
- It can be time consuming.
- Robot programming know-how is needed.
- The taught in positions are only applicable for the particular robot.
- If the relationship between the robot and the fixture is changed, each one of the robot positions has to be modified.

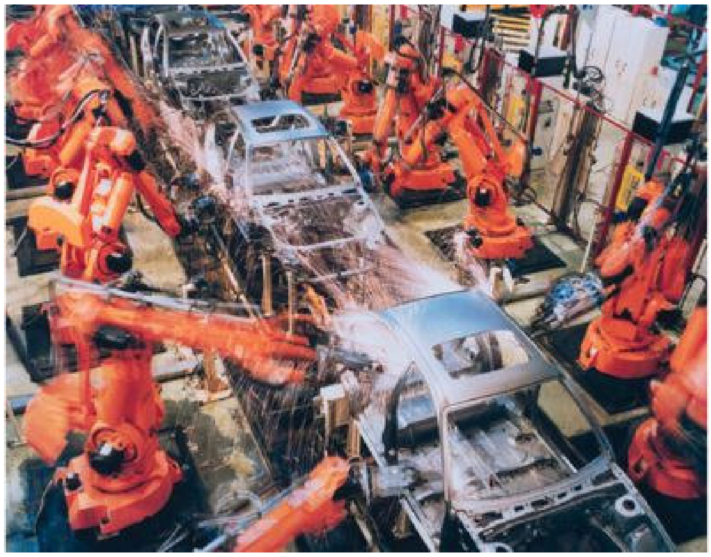


Figure 3
This figure shows a car body spot welding production line.

Imagine a car body spot welding line like in figure 3, the programs often consist of significant number of robot positions. Depending on the number of robots and positions in each robot program, the fine tuning of the robots programs in this spot welding line is a significant part of the commissioning phase.
In order, to cut expense and shorten the commissioning phase, it is desirable to find a fast and accurate method for calibration of robot program.

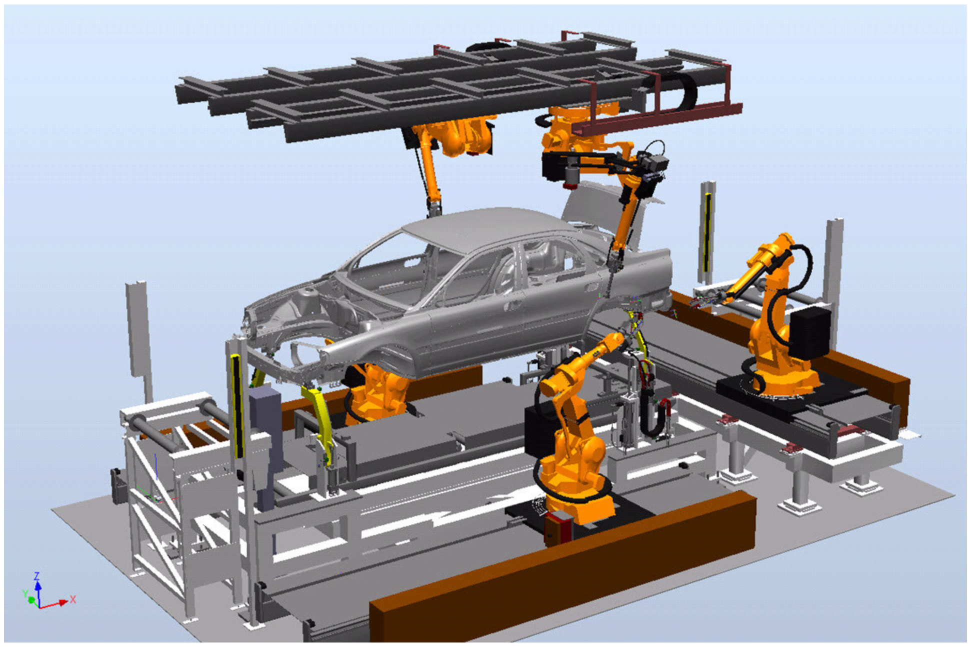


Figure 4
This figure shows a complete robot installation created by PC based graphical tool.

1.2 Offline programming

Due to the fact that it is desired to shorten the programming phase, it has become more common to create robot programs offline by using a PC-based graphic tool. By using this tool, the main programming can be made before the physical model is in place. A complete robot installation can be built up based on cad drawings. In figure 4, an offline generated model of a robot installation is shown. The dimensions and shapes of the objects are based on the CAD drawings.

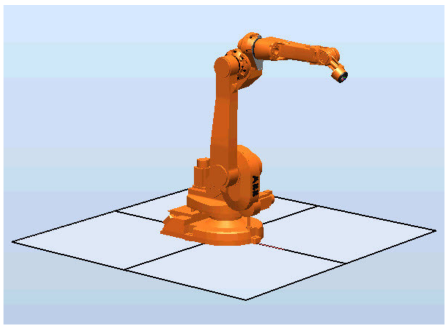
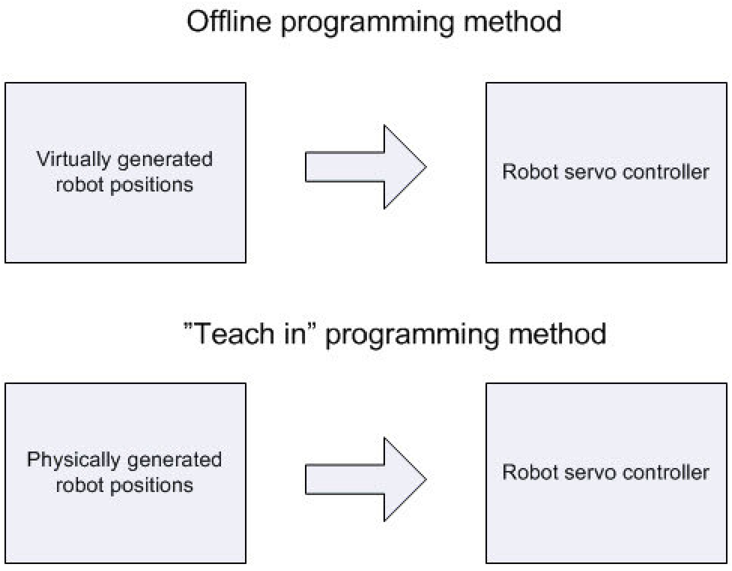


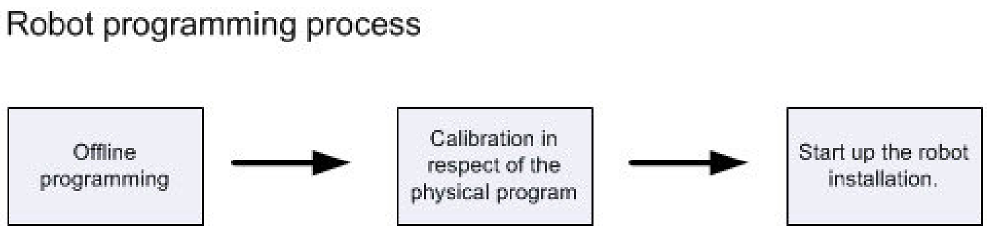
Figure 5
A robot with origin of the coordinates system at the robot foot

In most cases the coordinates system has its origin at the robot foot, see figure 5. The desired robot positions can therefore be based on the locations of the objects in this virtually built up production cell.

This means that the robot program is generated based on the virtually built up installation. The offline generated positions then will be transferred to the robot servo controller. In comparison to teach in programming where the programming is based on the physical model in reality.


Offline programming method
Virtually generated robot positions
Robot servo controller
"Teach in" programming method
Physically generated robot positions
Robot servo controller

However, the virtually generated robot program can in most cases not directly be transferred in to the servo controller and then be used by the robot in the real physical environment. The programmed positions will most likely deviate in the real physical environment. Due to this fact, the programmed positions in the physical installation have to be calibrated.


Robot programming process
Offline programming
Calibration in respect of the physical program
Start up the robot installation.

The deviations can be split up in to two categories, deviations related to the robot and deviations due to differences between the virtually generated robot installation and the real physical environment.

Deviations related to the robot are:

- Position and orientation deviation errors of the robot coordinate system in respect of the robot foot. Such errors can be caused by the tolerances in robots mechanical structure.

- Nonlinear robot movements. In the off line generated program the robot is considered to have perfect linear movements.

Deviation related to the offline generated robot installation and the physical installation is:

- Position and orientation discrepancies of the work objects in the physical environment compared to the virtual built up robot installation.

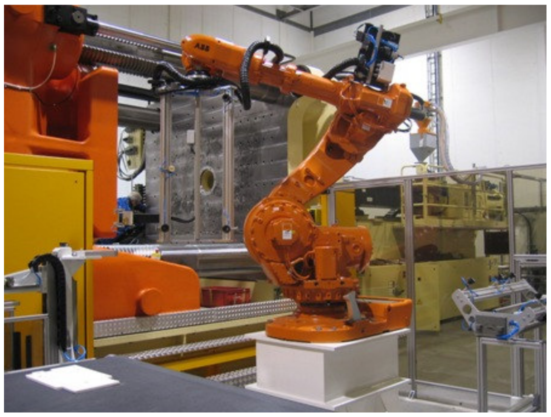


Figure 6
Any deviation in position or orientation of the robot’s installation base in comparison to the virtually created installation will generate discrepancies on positions and orientations of work objects.

1.3 Potential economic savings

Investing in robot automation is in many cases an investment in uptime and quality assurance. Compared to a human being, a robot can operate continuously day and night without need for breaks. The robot will also perform its task more or less identically each time. Any deviations due to human mistakes will therefore be eliminated.

As rule of thumb, the value generated in car production line is estimated to 50 000 sek per minute. This is more or less valid for producing standard as well for premium cars. This is explained by the fact that a standard car in general has shorter execution time than a premium car. Based on this fact, the cost due production downtime will in short period of time reach significant figures.

The loss due to production down time can in most cases never be regained. In other words, what is lost is lost. The goal from this point of view therefore is to minimize production downtime.

The same conclusion can be drawn when considering product defects due to quality issues. If the produced product does not reach the quality tolerances, the outcome is that it has to be scrapped. This means that failure to reach quality requirements can be considered as production downtime.

With a fast and simple calibration method the following savings will be obtained.

- If the robot breaks down, the robot can be replaced. After conducted calibration production can be resumed with minimum loss of production time.

- The calibration of the robot program can be verified at scheduled intervals. This means that the quality of the robot movements can be ensured by frequent calibration. This means economical savings due to reduced cost of poor quality

- Fast calibration will also allow reduce the transition time of introducing and changing fixtures. Shorter transitions will then reduce the cost due to downtime.

- The programming can be done in such a manner that the robot can be replaced without the need to retune every programmed position.

1.4 *Robot control system*

In figure 7, a main overview of the robot control system is shown. In order to control the robot arms, the robot needs instructions. For an ABB controller, these instructions are programmed by the user in the high level programming language RAPID. High level program language means that the program is quite user friendly.

1.4.1 RAPID

Rapid is the program language used to control ABB robots. The program allows control of the robot axes movements, mathematical calculations, I/O signals etc. The syntax of the program is similar to Basic.

Path positions in Rapid motion control instructions are compiled and transferred to the interpolator and path robot path planner.

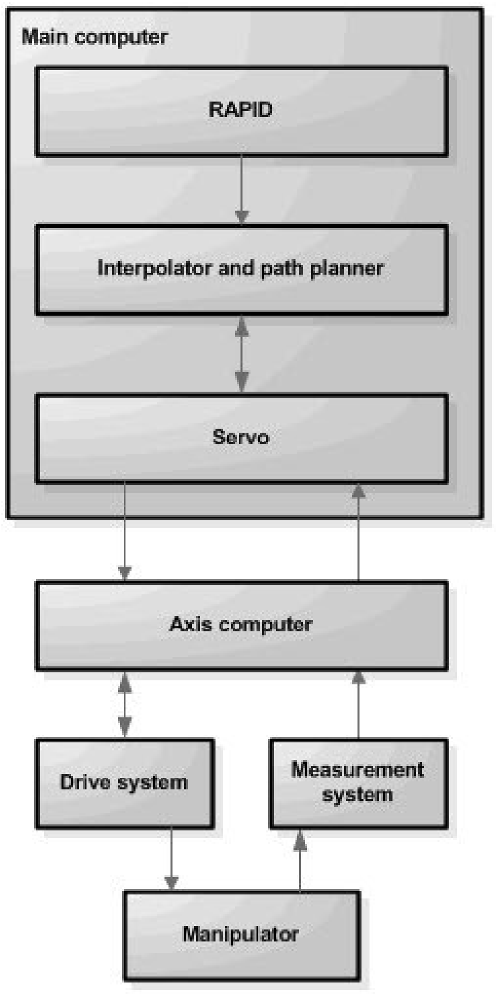


Figure 7
This figure shows the main overview of the robot control system.

1.4.2 Interpolator and path planner

The interpolator in collaboration with the path planner converts the positions in Cartesian coordinates into corresponding joint angle references. These calculations are internally supervised in order to prevent hazardous path references due to internal numerical calculation error.

Depending on the ordered instruction, the path planner will plan the path in different manners. In general the path planning consists of three different modes; joint movement, linear movement and circular movement.

Joint movement

Joint movement is used to instruct the robot to move from one position to another as fast as possible. This is obtained by minimizing the repositioning of the axes angle. This means that the movement of the tool centre point between the points in most cases will not be linear. In other words, the movement of the tool centre point between the points can not be controlled by this instruction.

Linear movement

Linear movement will instruct the robot to interpolate the tool centre point linearly between two positions at a constant programmed speed. Compared with the joint movement instructions this instruction requires a lot more data processing.

Circular movements

Circular movement will instruct the robot to move the tool on a circle arc. The instruction is based on three positions. Start position, bending radius position and end position. During the movement, the orientation is in most cases unchanged relative to the circle.

1.4.3 Servo controller

The joint references are further modified in the servo controller. As an additional precaution, the Servo also supervises movements with respect to numerical calculation errors. The servo also supervises the axes movements based on the joints angles values. Actual position, speed, torque values are continuously sampled and compared with reference values predicted by a virtual model of the mechanical robot. Collision is detected by a sudden difference between the actual and predicted torque required to maintain the programmed motion.

In most cases the robot has six independent axes. In order to control the robot to the desired position of each axis; the closed loop method is used. The closed loop function is based on PID (Proportional Integral Derivative) controller.

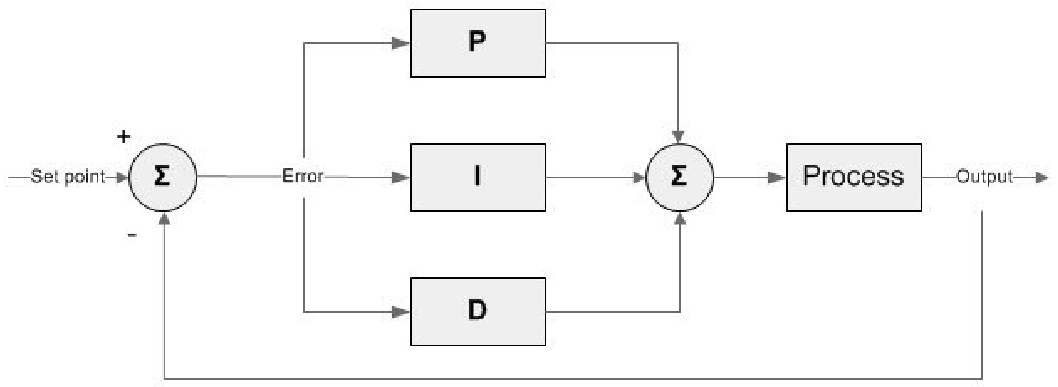


Figure 8
This figure shows a block diagram of a closed loop PID controller.

Proportional term

The proportional effect will increase the output proportionally to the actual position error. By changing the proportional gain the responses of the controller can be adjusted. However, a pure proportional controller may not converge to the target value; instead the controller will converge to a steady state error.

Integral term

The integral term is calculated by summing position errors over a period of time. The integral effect will then accelerate the processes towards the set point as long as the error remains. On The steady state error which often occurs in a pure proportional controller will be eliminated.

Derivative term

By using the derivative term, the rate of change can be adjusted. This term is determined by calculating the slope of the error.

1.4.4 Axis computer

The joint reference values are transferred to the axes computer. In the axes computer, the main control loops are located. The axes computer also supervises and processes the joint feedback values from the joint position measurement system.

1.4.5 Drive system

The drive system is the unit in the control system which supplies the electrical power to the robot motors. The control is obtained by the variable frequency method. This means that the system controls the frequency of the electrical power supplied to the alternate current motor. In other words, the rotation speed of the motor will be controlled as function of provided frequency of the electrical power.

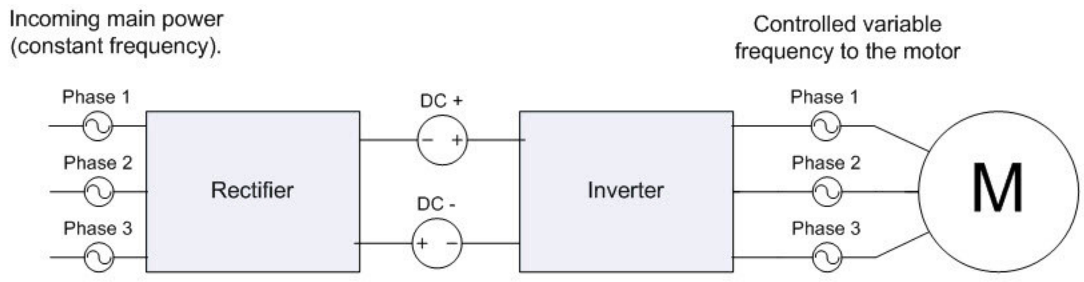

Incoming main power (constant frequency).
Controlled variable frequency to the motor
Phase 1
Phase 2
Phase 3
Rectifier
DC +
DC -
Inverter
Phase 1
Phase 2
Phase 3
M

Figure 9
This figure shows a main overview of the drive system.

1. The three phase alternate current is rectified in to positive and negative direct currents.

2. The direct current is converted in to AC power using an inverter switching circuit.

3. The switching circuit mainly consists of IGBT, insulated-gate bipolar transistors. These transistors are in turn controlled by logic signals.

1.4.6 Manipulator

In order to control the arms of the robot, each axis is equipped with a resolver which is an analogue device that returns the angular position of the robot axis. The position information is used to calculate position errors fed into the PID regulator of the servo controller.

However, if the robot is programmed repeatedly to the identical position, the robot may not end up to the physical identical position even if all resolvers are returning the correct feedback values. This is explained by the fact that there is no possibility to measure the deviations which may occur between the resolvers. Such deviations could be generated by thermal expansion and flexibility of the robot arms, play in gearboxes etc. In figure 11, an illustration of these deviations are shown. The conclusion is that the resolver values may not show the complete picture. However, if the operating conditions for the robot do not change, the repeatability to the programmed position is very high.

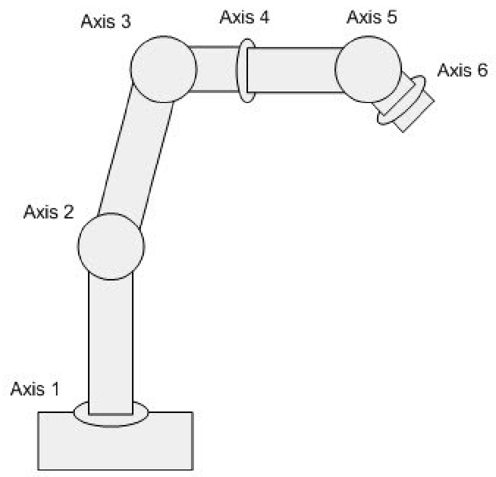


Figure 10
This figure shows the mechanical structure of robot. At each axis, a resolver is returning the angular position.

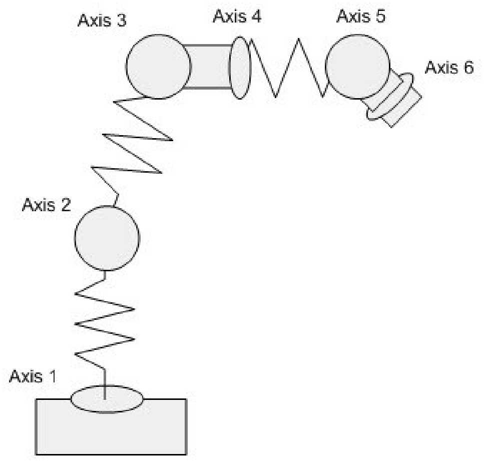


Figure 11
This figure shows potential deviations between the axes in the robot's mechanical structure.

1.5 Robot coordinate system

In order to determine and control the position of the robot, a three dimensional Cartesian coordinate system is used. The overlaying coordinate system is named World Coordinate System. In this coordinate system several others can be expressed, these systems are:

- Base coordinate system
- Wrist coordinate system
- Tool coordinate system
- User and object coordinate systems

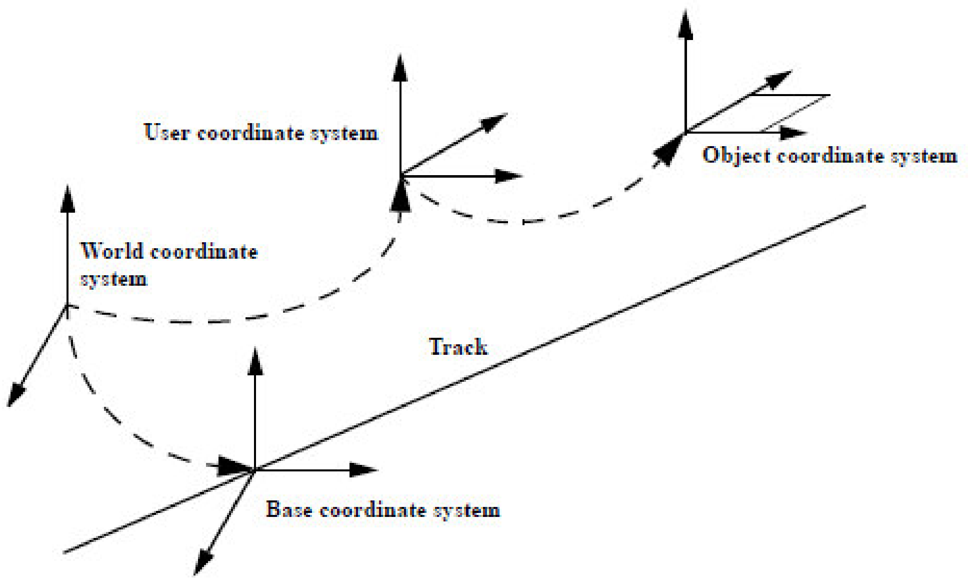


Figure 12
The relations between the robot coordinate systems.

1.5.1 Base coordinate system

The base coordinate system defines the position and orientation of the robot fin the world coordinate system. As default, the base coordinate system is equivalent with the world coordinate system.

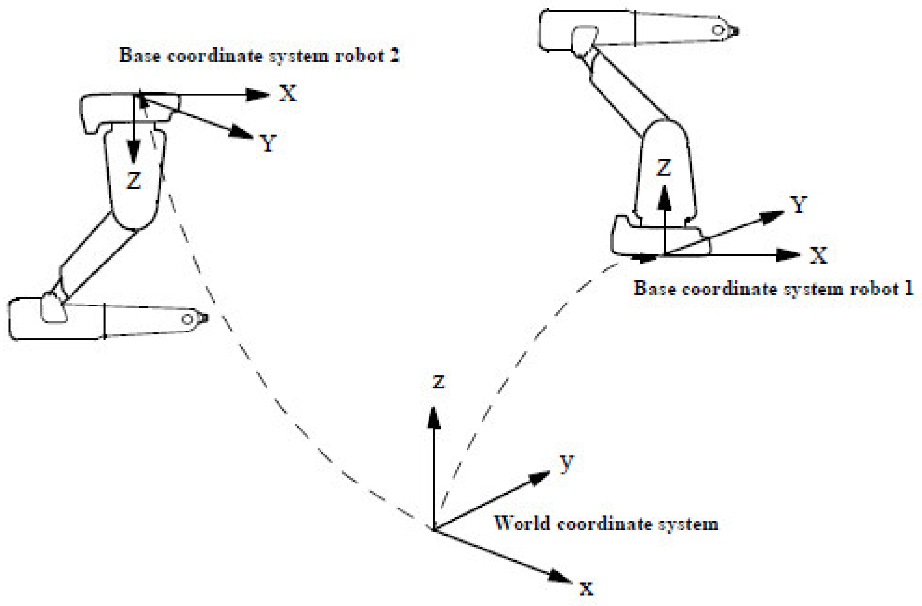

Base coordinate system robot 2
X
Y
Z
Z
Y
X
Base coordinate system robot 1
z
y
World coordinate system
x

Figure 13
Base coordinate system for two robots are expressed in the world coordinate system

1.5.2 Wrist coordinate system

The wrist coordinate system defines the position of the wrist in the world coordinate system. This coordinate system is static on the mounting flange.

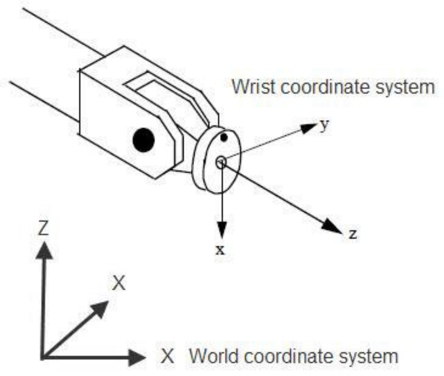


Figure 14
Origin of wrist coordinate system

1.5.3 Tool coordinate system

The tool coordinate system is used to define how tools are positioned and oriented with respect to the mounting flange.

Path positions defined in Rapid programs describe how the tool coordinate system should be position and oriented with respect to world or other used reference coordinate systems.

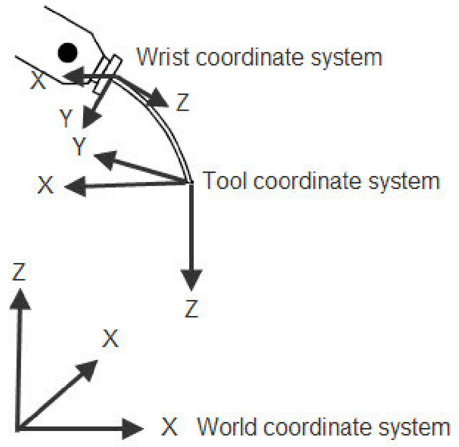


Figure 15
Tool coordinate system is expressed in the wrist coordinate system.

1.5.4 User and object coordinate systems

The combination of the user and object coordinate system is called work object. The user frame is defining a cartesian coordinate system in the world coordinate system. In turn, the object coordinate system is then expressed with respect to the user system.

These two coordinate systems are in many cases used to calibrate robot coordinates to programmed paths. By expressing programmed positions with respect to the object coordinate system, all path positions may be calibrated by tuning the work object to the actual relationship between the robot foot and the fixture.

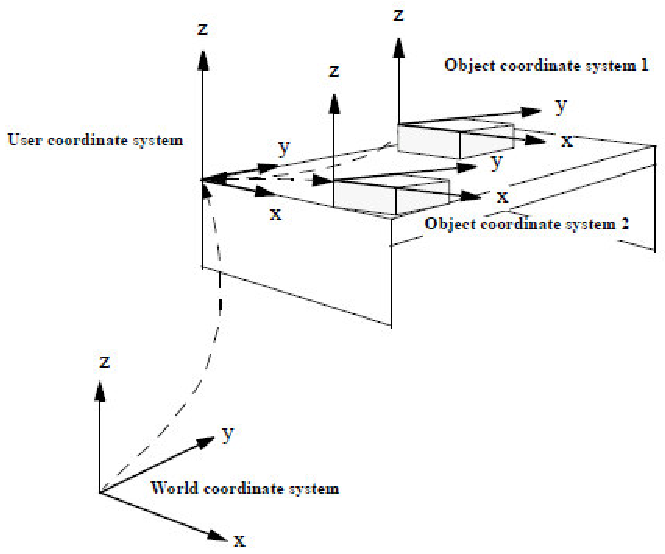


Figure 16
This figure shows the user and object coordinate system which is expressed in the world coordinate system.

1.6 *Robot calibration methods*

In order to determine the location of an object in the robot's coordinate system, several methods are currently used. In this chapter a couple of different methods are presented.

1.6.1 Manual calibration

Traditionally manual methods have been used. In short, these methods consist of moving the robot to reference points defining the work object. By using tapered tools, the edges are placed in front of each other. The position of the object can then be determined.

However, manual calibration methods are not very efficient when there are high demands on accuracy. The results of manual calibration methods tend to be individual. In other words, the result of the calibration can vary depending on the individual person. Due to these facts, it is more or less impossible to evaluate and estimate the capability of this method.

In many cases, this may still be a suitable calibration method. Especially when the robot application does not require high accuracy. From a cost competitive point of view, a manual calibration method is most likely the most suitable solution.

Advantages

- Simple
- Cost competitive

Disadvantages

- Operator dependent method
- Time consuming

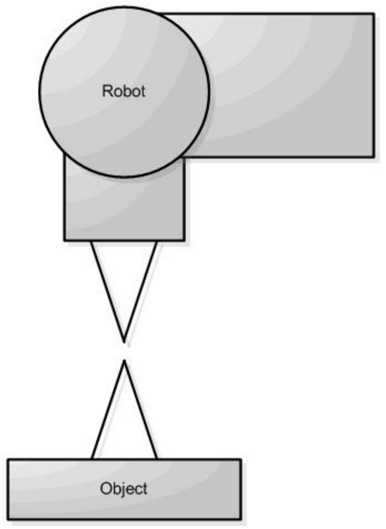


Figure 17
This figure shows a tapered tool mounted on the robot tool flange. Another tapered tool is mounted on the object fixture. By moving the peak of the robot's tapered tool above the peak of the fixed tapered tool as close as possible, the reference position can be determined.

1.6.2 Bulls eye – calibration of TCP

Bull's eye is a calibration device which determines the TCP (Tool Centre Point) and orientation of a robot tool. The TCP which is the origin of the tool coordinate system is defined with respect of the origin of the wrist coordinate system. The method does only work on tool's which has a concentric portion along its centreline.

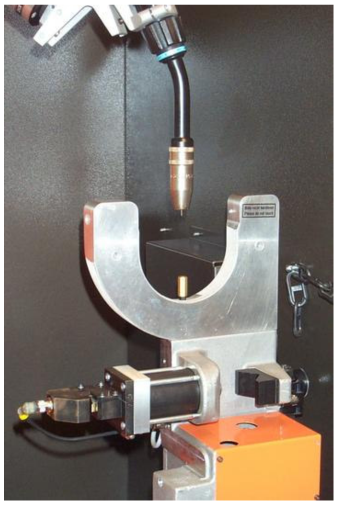
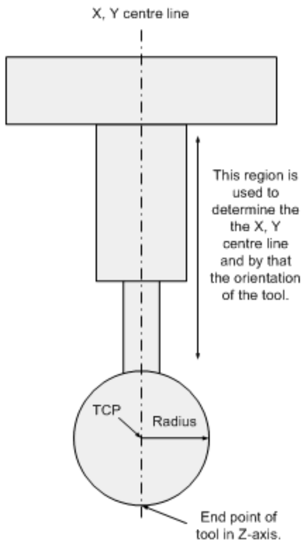


Figure 18.1
Bulls eye installation.

The method is based on laser technology. By moving the robot's tool through the laser beam, the physical width of the concentric portions of the tool can be determined given that the nominal geometry of the tool is known. By measuring slices of the width, the centreline can be determined. Based on the determined centre line, the tool orientation is set. Finally, the end of the tool will be determined.

To summarise, Bulls eye can determine the centreline and the end of tool. Based on this fact, the method can not find the centre point of a probe sphere. However, it will determine the X, Y centre line and the end of the tool in Z-axis. In other words, the TCP can be mathematically determined if radius of the probe is well known, please see figure 18.2.


X, Y centre line
This region is used to determine the the X, Y centre line and by that the orientation of the tool.
TCP
Radius
End point of tool in Z-axis.
Figure 18.2
Calibration the TCP of a spherical probe by using bulls eye.

Advantages

- Define TCP accurate in X and Y axis in the tool coordinate system.
- Determines the orientation of the concentric centre line

Disadvantages

- Relative expensive
- Set up phase.
- Can only be used for concentric tools.

1.6.3 Calibration by Force Control

Force control calibration is designed to use as an add-on for robots equipped with the ABB force control function package which monitors applied forces on an objects using external sensors. In other words the robot can "feel" how much force is applied on to the object. The function is mainly intended to be used for finishing processes such as deburring, grinding and polishing.

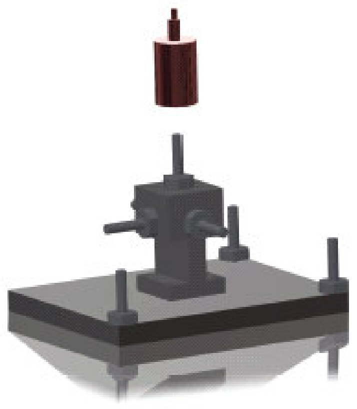


Figure 19.1
Force control calibration device.

In the basic configuration the Force Control system is able to detect and calibrate the tool centre point of the attached tool automatically. As an additional option, the work piece can be defined as well. This is possible by defining the position of thee stationary reference tips on the work object.

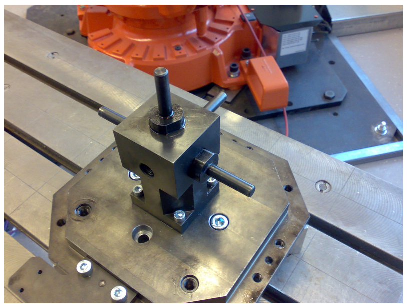


Figure 19.2
The "cube" for the TCP calibration.

Advantages

- It is accurate, according to the product specification the TCP is defined within +/- 0.05-0.15mm depending on robot model.
- It can be used to calibrate work objects.
- Operator independent.

Disadvantages

- Requires additional equipments, as for example the TCP measurement receiver.
- Can not be used without the force control functional package.

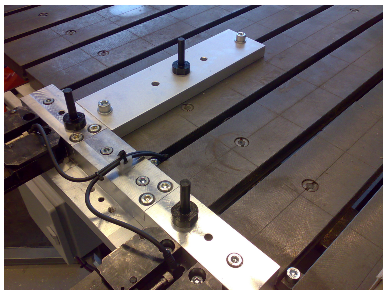


Figure 19.3
Three reference tips for calibrating a work object by force control

1.6.4 Laser LAB

Laser lab is a laser based calibration technology. The method consists of a measuring device called “Laser LAB” and a measuring sphere. The Laser LAB measuring device consists of five individual laser sensors. The laser sensors are positioned as a pentagon in the device and are aligned so that the five laser rays will intersect in one common point.

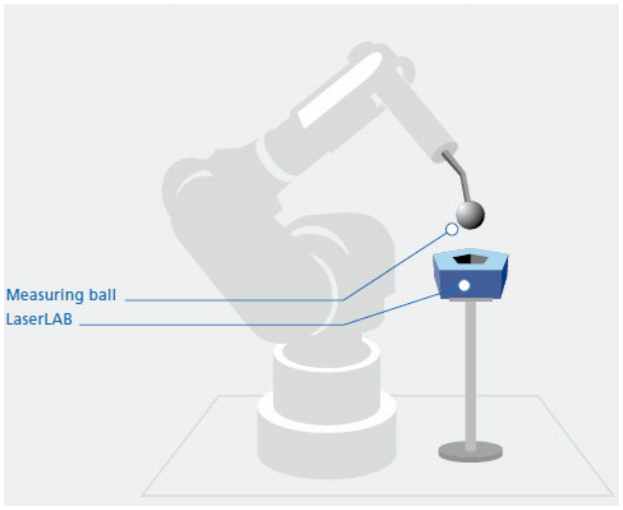


Figure 20.1
Laser LAB calibration equipments.

By positioning the sphere in the laser device, positions on the surface of the sphere can be determined in three dimensions. This is obtained by measuring the distances from each one of the five sensors. By obtaining surface points, the position of the sphere can be determined.

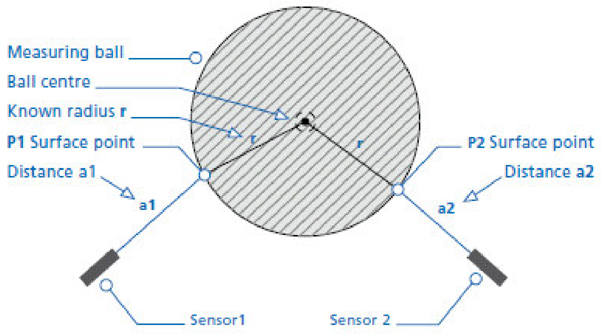


Figure 20.2
By using five sensors, the position of the spherical object can be determined.

Advantages

- Highly accurate, according to the product specification +/- 0.1mm.
- Measure TCP as well as fixtures.
- Operator independent.
- Easy to operate.

Disadvantages

- Requires additional equipments.
- Setup phase.
- Investment.

1.6.5 ABB Absolute accuracy

Absolute accuracy is a laser based method. By using a laser tracker, a number of coordinate positions are determined in the robot's working area. By comparing the theoretical positions in the robot controller and actual position of the robot's mounting flange, a set of corrections to the transformation parameters from the robot axes to the servo controller can be defined. These parameters will compensate the robot's positioning and thereby its movements. In other words, after the calibration the actual position of the mounting flange will more or less correspond to the position value in the robot controller.

According to the product specification, the parameters take into account both the mechanical imperfections in the pattern of the movements and the bending or distortions downwards caused by loads.

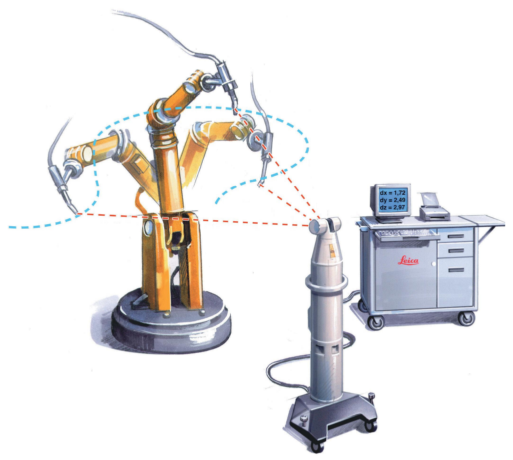


Figure 21
In this figured the robot's TCP is measured by a laser tracker. The robot is instructed to move to 100 different positions. At each position the laser tracker determines the actual position in the reference space. By comparing the actual and the theoretical values the compensations can be calculated.

Advantages

- Highly accurate according to the product specification within 0.2-0.3mm.
- Reduces axis non linearity. According to the product specification the average deviation is reduced to 0.5mm.
- Operator independent.

Disadvantages

- Requires additional equipments.
- Expensive.
- Significant setup phase.
- Special competence required.
- Special environmental conditions required.

1.6.6 PosEye®

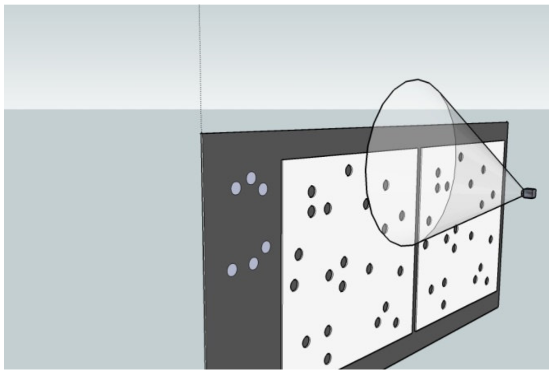


Figure 22
This figure shows the reference system for the Pos Eye optical sensor.

A PosEye®-system measures the position and orientation of the sensor with respect to a reference system consisting of known patterns. The measurement principle is similar to global positioning system. If the sensor or the reference system, or both, are moving is a matter of choice. Most other 3 to 6 degrees of freedom measurement systems uses methods where the object of interest is focused on at a distance – outside-in-systems. These methods may have some advantages but the method does also imply that it is hard to produce high accuracy measurements on the orientation of the object and to cover full re-orientations of the object. PosEye® is an inside-out-system that delivers excellent measurements of the orientation, as well as very good position measurements. The PosEye®-sensor is placed at the point of interest. On a robot this will in most cases be on the mounting flange. The references are passive fixes in the surroundings like black spots on a white background.

Advantages
- Highly accurate.
- Measures six degrees of freedom. This means that both the position and the orientation can be measured.

Disadvantages
- Additional equipment required.
- The sensor requires visual sight to the references.
- Reference spots have to be placed on solid environment; any movement will have an influence of the measurement.
- Requires special competence.
- Initial setup phase.
- Investment.

1.6.7 ABB Navigator

Robot calibration is usually done by localizing calibration points using the robot as measurement equipment. Navigator is an automated, very accurate and user independent technique that replaces these manual steps.

The standard idea of localizing calibration points is used by Navigator as well. However, instead of letting the user manually point out positions the robot is equipped with a spherical probe tool and the robot cell is prepared with mounting holes on the fixture for spherical objects. The calibration is performed by letting the robot locate the spherical objects on the fixture. The sensor mechanism is tactile, i.e. the touch between objects is determined and causes the robot to stop. The fixed sphere is connected to ground, and a voltage is applied to the spherical probe. The tactile touch is detected when an electrical circuit is established.

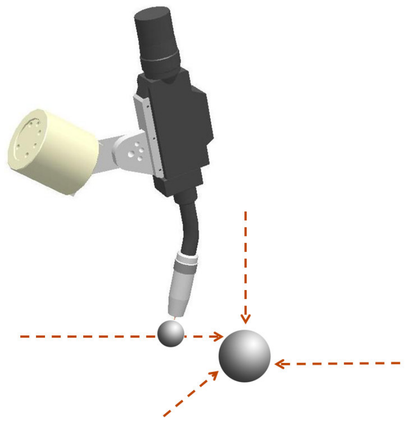


Figure 23
When the two spherical objects get in contact, a electrical circuit is established. At this point an I/O value is changed and the current position is stored.

Advantages

- Operator independent method.
- Relative easy to operate, no special competence is needed.

Disadvantages

- Additional equipment required.
- Requires voltage supply to the spherical probe.
- An accurate method in order to calibrate the TCP of the spherical probe is required.

1.6.8 Summary of calibration methods

Presented calibration methods are more or less weaker and stronger in different aspects. What is common for the calibration methods with superior accuracy are that they are in general expensive; requires additional equipment, time consuming setup phase and special competence. On the other hand, the cheapest calibration method which is the “teach in” manual calibration method does not ensure any accuracy at all. On top of this, it also depends a lot on the operator. From this point of view, this calibration method can be time consuming. If desirable accuracy has not been achieved the process has to be repeated again. In many production environments, the fact is that lost time is equal to lost money.

In this thesis, an operator friendly, cost competitive and accurate calibration method will be evaluated. This calibration method is very similar to the Navigator calibration method. By obtaining positions around of a fixed sphere; the centre position of the fixed sphere can be determined. This means that a reference position has been obtained. The main difference is that it will not require additional equipments in form of power supply and I/O connections as the Navigator calibration method requires.

The main advantages with this method would be:

- It is economical competitive in comparison to most other calibration methods.

- In comparison to other calibration methods, very limited additional equipment is required. This also means faster setup and commission phase.

- It will be easy to operate, no need for special competence. The calibration may for example be conducted by a robot operator.

- Flexible and reduced start up time. Recalibration of work objects can be conducted at scheduled interval. By recalibrating work objects at set time interval, the continuous position accuracy can be ensured.

- Software will be built in the robot program language rapid. This means that no addition devices complicated installations and setups are required.

2 Problem definition

The calibration problem mainly consists of determining the true location of a physical object in the robot's coordinate system. This is required due to the fact that every single robot more or less has unique Cartesian coordinate system.
For example, a robot trajectory has been programmed around the edges of an object. This particular object is modelled in a CAD drawing. The location of the object has been virtually defined with respect to robot coordinates when the program was designed. However, in this case the virtually defined location of the object does not correspond to the true location of the object in the robot's coordinate system. This means that the object has to be calibrated in the true coordinate system.

One method to calibrate the object is by implementing reference points on the object. These reference points have been defined on the cad drawing as well as on the physical object. By letting the robot locate these reference positions; the location of the object can be determined. In order to determine the orientation of the object at least three positions are required.

Based on the defined true positions of the reference points on the physical object, the virtual programmed robot path can be modified. In other words, the virtually designed robot programmed can be used.

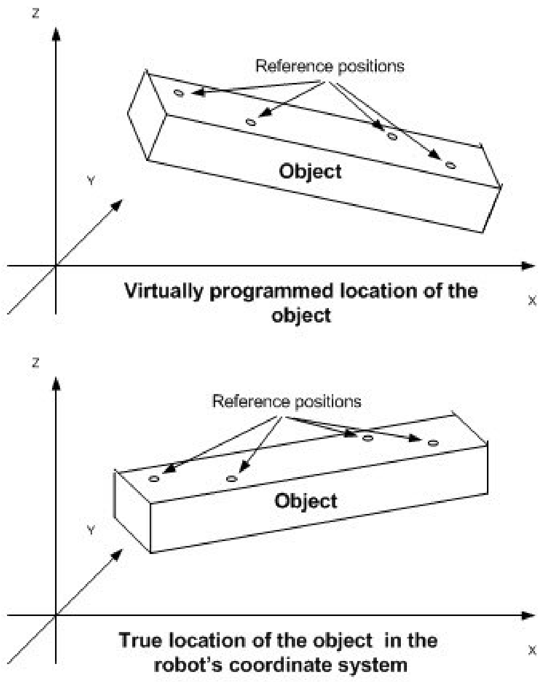

z
Reference positions
Object
Y
Virtually programmed location of the object
x
z
Reference positions
Object
Y
True location of the object in the robot's coordinate system
x

Figure 24
In this figure, an identical object has been defined in two Cartesian coordinate systems. Reference points are located on the object. These positions have different coordinates in the two coordinate systems. The calibration transform is the best fit transformation from nominal virtual references to actual physical references

2.1 Calibration theory

Imagine a linear robot axis movement which has been programmed offline. As earlier mentioned, in reality the robot axis movement is not perfectly linear. However, the servo controller will interpret the movements as perfect linear movements. The consequence of this fact is shown in figure 25.1, the programmed trajectory deviates from the true robot path.

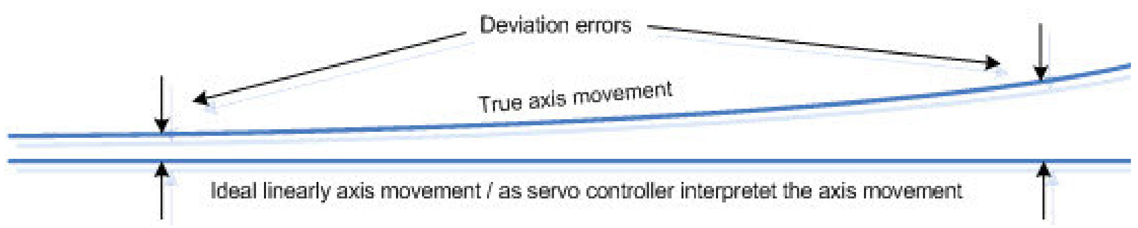

Deviation errors
True axis movement
Ideal linearly axis movement / as servo controller interpretet the axis movement

Figure 25.1
This figure shows the deviation between an offline programmed path and the true physical robot path.

Consider a work object which have been defined and programmed offline. In other words, the position of the object in relation to the robot has been set based on a virtual CAD drawing. In such a case, the programmed robot trajectory in reality will deviate from the desired programmed trajectory. As a consequence, the robot will in reality not move to the positions which has been programmed virtually offline. In order to correct this deviation, the position of the work object has to be calibrated.

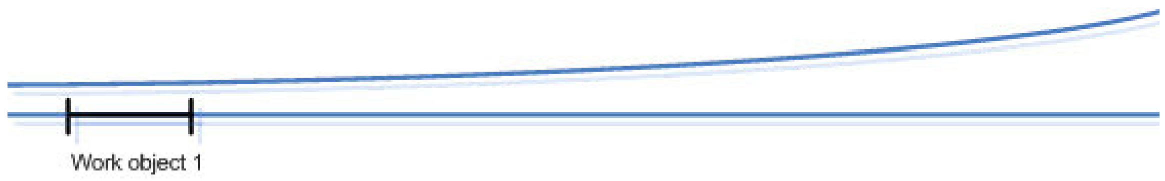


Figure 25.2
This figure shows a local work object which has been programmed offline. As shown, the position of the work object does not correspond to the actual path. Therefore a local calibration based on the local best fit of the actual path has to be done.

By calibrating the robot to this position, the robot will reach this position. A good correction therefore has been obtained close to the origin of the work object.

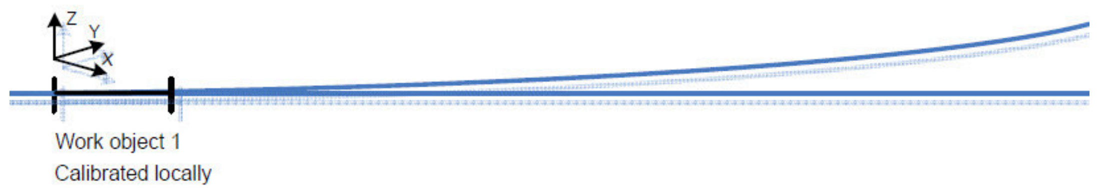


Figure 25.3
This figure shows a calibrated work object which has been calibrated locally based on the local actual path.

However, consider that there is an additional work object positioned in the robot's working area. This object is according to the CAD model equally aligned as object 1 in figure 25.3. However, there is a significant distance between the two objects. In many cases, the robot will not have correct trajectory for the second object. This is explained by the fact that the robot's true linear trajectory somewhat deviate from a perfect linear movement. In other words, the calibration of the first work object will in many cases not be suitable for additional work objects.

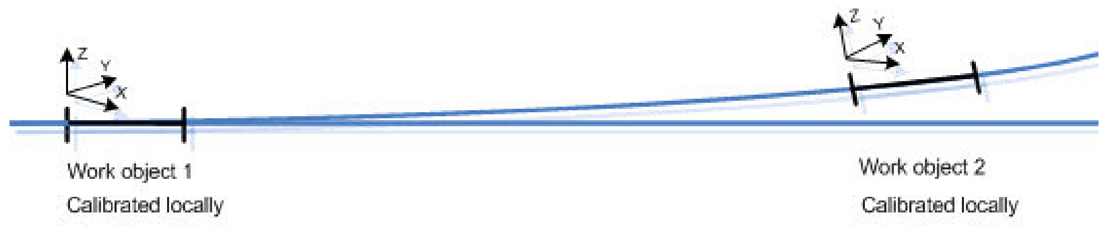


Figure 25.4
Two work objects which have been calibrated by create local coordinate system for each one.

In order to calibrate several work objects, local coordinate system has to be created for each one. In figure 25.4 local coordinate system has been created for work object 1 and work object 2. Each one of these work objects has been individually calibrated. In other words, the local coordinate system will be a best fit approximation for the local nonlinearity.

Within the local coordinate systems, the deviations due to nonlinear movements will be reduced. However, the work objects do have some limits. The work objects can not be too wide. If they are too wide, the nonlinear axis movements may cause undesirable position deviations.

2.2 Calibration of work object coordinate system

Calibration of a frame will be based on reference positions. In the robot program language Rapid this is obtained by the command DefAccFrame. The command uses two sets of positions. One set of positions is the reference positions; the second set of positions is used to define a work object frame. This work object frame will be the calibrated frame. This method compares the relation between the nominal virtual reference positions and the located actual located positions. The calibration transform is the best fit transformation from nominal virtual references to actual physical references

The objective in this thesis is to evaluate the performance of a method for determine reference positions. If the accuracy is good enough; the method can be applied with the built in command DefAccFrame. With this command, the frame then can be calibrated by the best fit method.

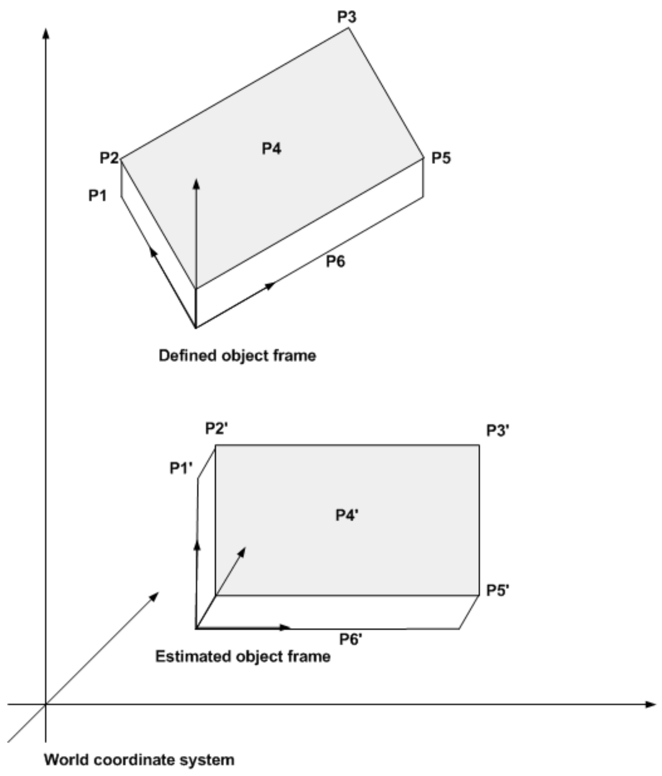


Figure 26
This figure shows how the function is using two sets of reference points. These two sets will be used in order to calibrate a work object frame. By using these points, the DefAccFunction will determine a best fit estimation of the object frame expressed in the world coordinate system.

3 Limitations

- Evaluation tests have only been conducted by the robot model IRB1600ID.
- Tests have not considered significant external loads on the robot.
- Metallic spheres will be considered as more or less perfect spheres.
- Measurement results do not take in to consideration the potential effect of material heat expansion of the robot arms.

4 Method

In this section, a method used to determine the reference position is described. The objective is to find the centre of a sphere by locating positions around it. By using at least four positions around the sphere, the centre point of the sphere can be determined. By that a reference point in the physical real model has been determined.

4.1 Determine the centre point

Due to measurement errors in the model the received values will not be exact. In other words, the distance to the centre point will not be constant due to the variations.

As shown in figure 27, a great number of positions around a sphere are distributed. The distance to the true centre point is individual for each position. Based on this fact, the conclusion is that the equation system will not have an exact solution. An optimization or best fit method in order to estimate the centre point is required. Based on a set of positions, the least square optimization or best fit method can be applied.

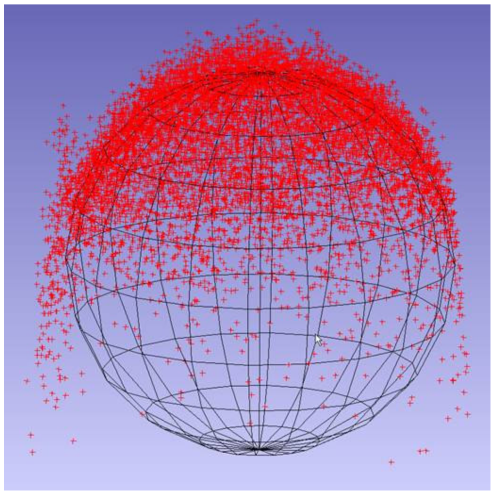


Figure 27
This figure shows a perfect sphere partly covered by a point cloud.

4.1.1 Optimization goal

In order to find the best fit of the centre point, the optimization goal is to minimize the sum of squared radius error. This gives the following equation.

Minimize $E = \Sigma_{n=1} (R_n - R)^2$

Where

$R_n = \sqrt{((X_n-X)^2+(Y_n-Y)^2+(Z_n-Z)^2)}$

X, Y and Z parameters will represent the best fit values for the centre position of the sphere and R the best fit radius. This is a function of second order. In other words, the least square problem has to be solved by a nonlinear method.

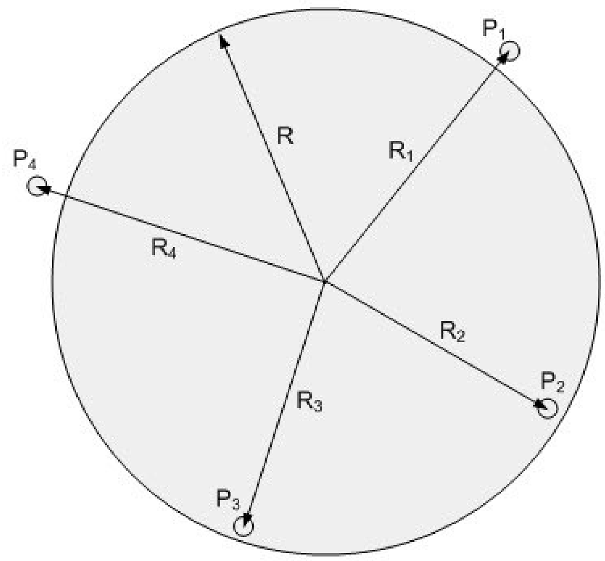


Figure 28
This figure shows a perfect sphere with radius R. Four positions have been obtained around the sphere. The distance from each one of these points deviates all from the radius of the sphere R.

In figure 28, number of points P1 to P4 is located around a perfect sphere. Each one of these positions somewhat deviates from the optimal radius R. Every single deviation can therefore be expressed as the error:

$e_n = R_n - R.$

However, in order to only get positive deviation values, the power of two is added to the function, this gives

$E_n = e_n^2 = (R_n - R)^2.$

In order to determine the best fit of R, the goal is to minimize the sum of squared deviation errors, $E_1 + E_2 +\ldots+E_n$.

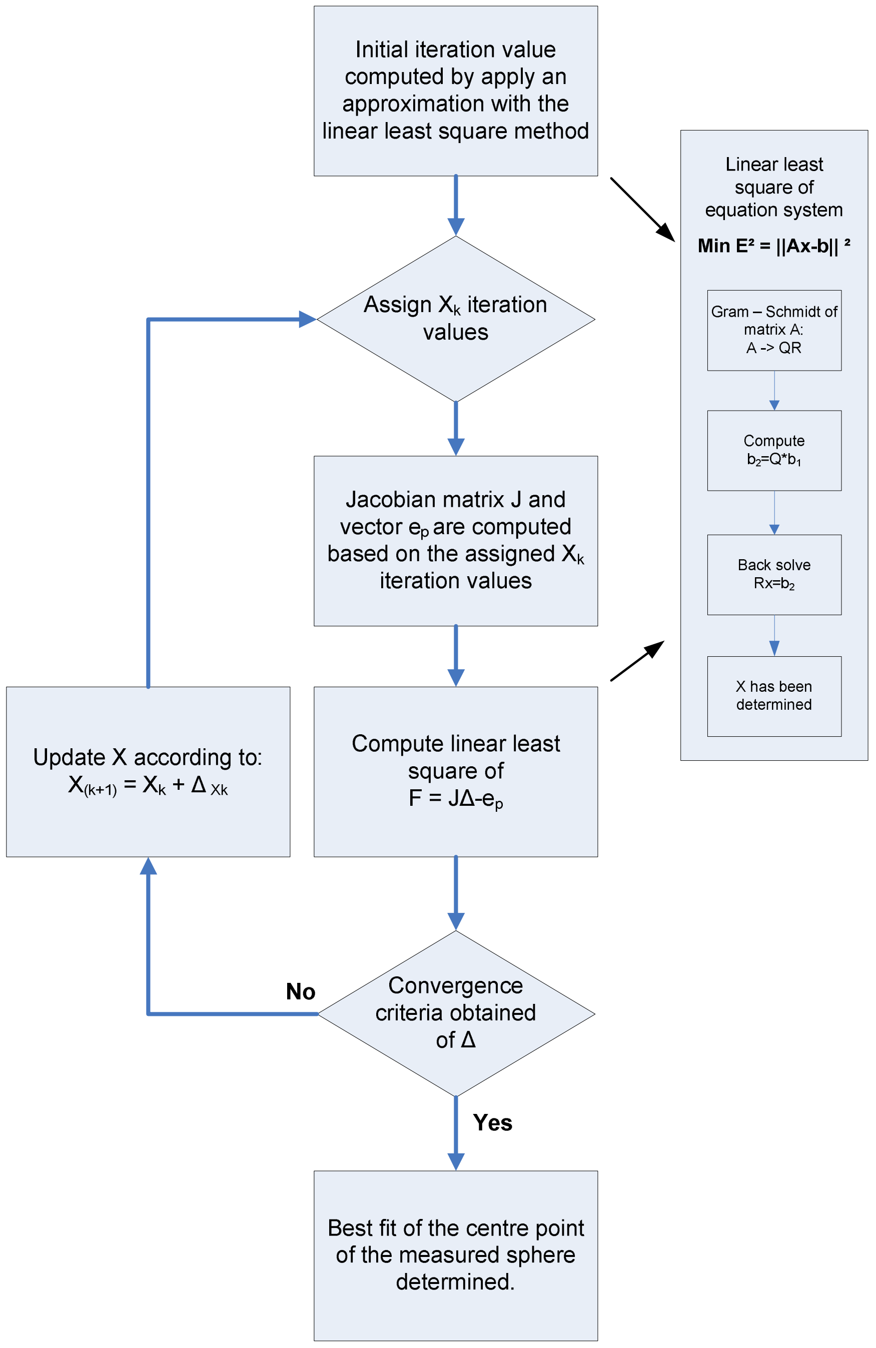

Initial iteration value computed by apply an approximation with the linear least square method
Assign X_k iteration values
Jacobian matrix J and vector e_p are computed based on the assigned X_k iteration values
Compute linear least square of $F = J\Delta - e_p$
Convergence criteria obtained of Δ
No
Yes
Update X according to: $X_{(k+1)} = X_k + \Delta_{Xk}$
Best fit of the centre point of the measured sphere determined.
Linear least square of equation system
Min $E^2 = ||Ax-b||^2$
Gram – Schmidt of matrix A: A -> QR
Compute $b_2=Q^*b_1$
Back solve $Rx=b_2$
X has been determined

Figure 29
A flow chart of the applied optimization or best fit method. The method is built on the Gauss Newton method. An initial iteration value is given by apply an approximation with the linear least square method.

4.1.2 Gauss Newton method

Due to the fact that the optimization function is of second order, the Gauss Newton algorithm has been chosen. This is an iterative algorithm which is based on Newton's method. The choice of this algorithm has been based the following facts:

- Efficient, finds optimum in relative few numbers of iterations.
- It behaves well near the optimal point.
- It does not require a second order derivate.
- Each iterative step will be solved by the linear least square algorithm.

The Gauss Newton method goal is:

$$\text{Min } E=e_p(x)^2=e^Te$$

The Gauss Newton method is built on Newton's method:

$$X_{(k+1)} = X_k+ \Delta$$

Where Δ stands for Newton's optimal direction. Δ is given by:

$$\Delta = - H_E^{-1}*gE$$

H_E is the hessian matrix of the function E. The hessian matrix is approximated as:

$H_E = d^2E/dx_idx_j \approx 2*J^TJ$ Where J is the Jacobian matrix.

gE is the gradient of the function E which gives:

$$gE = dE/dx_i = 2*de/dx_i*e_p(x) = 2*J^T e_p(x)$$

At this stage the gradient of E and the hessian matrix of E are determined, this finally gives.

$$X_{(k+1)} = X_k+ \Delta; \quad \text{where } \Delta=- (J^TJ)^{-1}* J^T e_p(x)$$

At each iteration, the optimal direction Δ will be given by solving $-(J^TJ)^{-1}* J^T e_p(x)$. This function is equivalent to solve a linear least square problem.

The Jacobian matrix J is given by:

$$\begin{matrix} de_1/dX & de_1/dY & de_1/dZ & de_1/dR \\ dE_2/dX & de_2/dY & de_2/dZ & de_2/dR \\ \dots & \dots & \dots & \dots \\ de_n/dX & de_n/dY & de_n/dZ & de_n/dR \end{matrix}$$

In other terms:

$$\begin{matrix} -(X_1-X)/R_1 & -(Y_1-Y)/R_1 & -(Z_1-Z)/R_1 & 1 \\ -(X_2-X)/R_2 & -(Y_2-Y)/R_2 & -(Z_2-Z)/R_2 & 1 \\ \dots & \dots & \dots & \dots \\ -(X_n-X)/R_n & -(Y_n-Y)/R_n & -(Z_n-Z)/R_n & 1 \end{matrix}$$

Where

$$R_n = \sqrt{((X_n-X)^2+(Y_n-Y)^2+(Z_n-Z)^2)}$$

The iteration start with an initial guess of the vector $X_i = [X_k, Y_k, Z_k, R_k]$. Based on this guess, the new reference values for $X_i = [X_k, Y_k, Z_k, R_k]$ will be updated according to:

$$X_{(k+1)} = X_k + \Delta_{Xk}$$
$$Y_{(k+1)} = Y_k + \Delta_{Yk}$$
$$Z_{(k+1)} = Z_k + \Delta_{Zk}$$
$$R_{(k+1)} := R_k + \Delta_{rk}$$

This process will be repeated until the function has fulfilled the convergence criterion. In other words, when all values in the vector Δ gets below the set convergence threshold criteria.

The convergence threshold level is set based on the requirements of the application. Further iterations will improve the final solution. However, the improvements of the solution are due to convexity of the function minor than the threshold level. In other words, if the threshold level is set based on sufficient accuracy, further iterations will insignificantly improve the solution.

4.1.3 Gauss Newton conditions

The Gauss Newton is an unconstrained convex optimization method. In other words; the method will only find the global optimal minimum or mathematically expressed, converge towards the global minimum under the condition that:

- Initial iteration value is set in region where the function to optimize is convex.
- The global minimum has to be located in this particular convex region.

In figure 30, a general three dimensional graph is shown. This graph consists of several optimal minimums, local as well as global. This means that from a global point of view the shown function is not convex. In other words, the Gauss Newton method would converge in to different solutions depending on the start iteration point.

Based on these facts, the Gauss Newton method can not in general be applied on such a function. A determined solution can be local minimum as well as global minimum. However, under condition that the start point is located within the convex region where the global minimum is located, the method will converge towards this point.

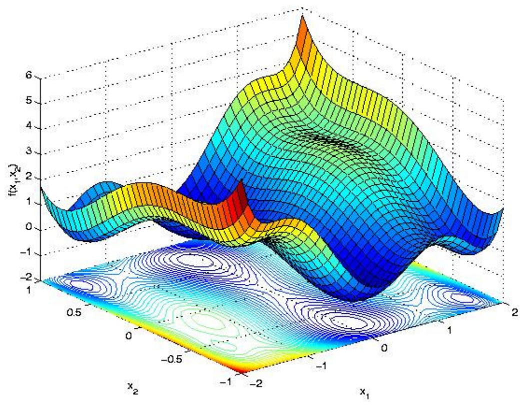


Figure 30
This figure shows a three dimensional graph which contain several local minimum points. However, the graph has only one global minimum.

4.1.4 Gauss Newton initial iteration value

In order to ensure that the Gauss Newton starts to iterate in the region where the global minimum is located, the start iteration point has to be near the optimal minimum. This has been generated by applying an approximation that distance between the centre point and the points around the sphere are identical for all positions around the sphere. In other words, this mathematical expression does not take in to consideration deviation of the radius.

4.1.5 Derivate centre point equation

The undetermined centre point is given by the three dimensional frame coordinates:

X: Cx Y: Cy Z: Cz

Points around the sphere are given by three dimensional coordinates. The distance to the centre of the sphere point is equal for all four points.

1. $P_1 = X_1 Y_1 Z_1$
2. $P_2 = X_2 Y_2 Z_2$
3. $P_3 = X_3 Y_3 Z_3$
4. $P_4 = X_4 Y_4 Z_4$

By applying Euclidian distance theorem on all four points, following four equations are derived:

1. $(R_1+R_2)^2=(X_1-C_X)^2+(Y_1-C_Y)^2+(Z_1-C_Z)^2$
2. $(R_1+R_2)^2=(X_2-C_X)^2+(Y_2-C_Y)^2+(Z_2-C_Z)^2$
3. $(R_1+R_2)^2=(X_3-C_X)^2+(Y_3-C_Y)^2+(Z_3-C_Z)^2$
4. $(R_1+R_2)^2=(X_4-C_X)^2+(Y_4-C_Y)^2+(Z_4-C_Z)^2$

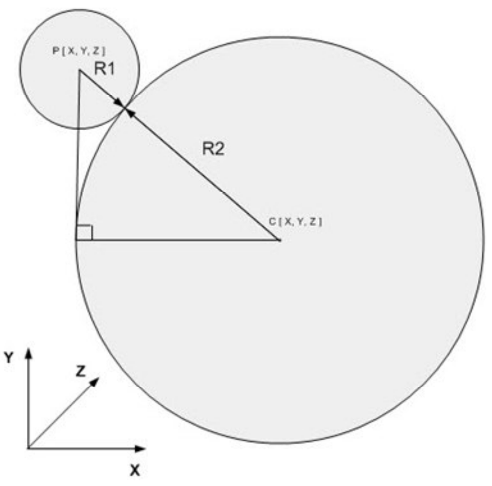


Figure 31
The figure shows two perfect spheres which are in contact with each other. Given that the spheres are perfect the distance between their centre points can be determined by adding their radius i.e. R1 + R2.

By substituting the equations 2 in to1, 3 in to 2 and 4 in to 3, following three equations are derived.

1. $2(X_2-X_1)C_X+2(Y_2-Y_1)C_Y+2(Z_2-Z_1)C_Z=X_2^2-X_1^2+Y_2^2-Y_1^2+Z_2^2-Z_1^2$
2. $2(X_3-X_2)C_X+2(Y_3-Y_2)C_Y+2(Z_3-Z_2)C_Z=X_3^2-X_2^2+Y_3^2-Y_2^2+Z_3^2-Z_2^2$
3. $2(X_4-X_3)C_X+2(Y_4-Y_3)C_Y+2(Z_4-Z_3)C_Z=X_4^2-X_3^2+Y_4^2-Y_3^2+Z_4^2-Z_3^2$

The linear equation system which can be expressed as:

$$A*c=B$$

$$\overset{\textbf{A}}{\begin{bmatrix} 2(X_2-X_1) & 2(Y_2-Y_1) & 2(Z_2-Z_1) \\ 2(X_3-X_2) & 2(Y_3-Y_2) & 2(Z_3-Z_2) \\ 2(X_4-X_3) & 2(Y_4-Y_3) & 2(Z_4-Z_3) \end{bmatrix}} \overset{\textbf{c}}{\begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix}} = \overset{\textbf{B}}{\begin{bmatrix} X_2^2-X_1^2+Y_2^2-Y_1^2+Z_2^2-Z_1^2 \\ X_3^2-X_2^2+Y_3^2-Y_2^2+Z_3^2-Z_2^2 \\ X_4^2-X_3^2+Y_4^2-Y_3^2+Z_4^2-Z_3^2 \end{bmatrix}}$$

The solution to c (Cx, Cy and Cy) will then be given by solving:

$$c = A^{-1} * B$$

The advantage is that this is a linearly equation system. The solution will be found by determine the solution to Cx, Cy and Cz. Therefore, an approximate solution to the centre point can be been determined by using the linear least square method. This approximation will then be used as the initial iteration point for the Gauss Newton method.

4.2 Linear least square

The linear least square method is used to approximate the parameters of an over determined linear equation systems. If the linear equation system has an exact solution the following condition is fulfilled:

$$Ax-b \quad = 0$$

If there is no exact solution, the function can be expressed as:

$$Ax-b \quad = E$$

In this equation E stands for the error. By adding power of two, the function will express the error as a parable. The values will then only be positive.

$$E^2 = ||Ax-b||^2$$

This is a convex function. Given a convex function, a local minimum is also the global minimum. The global minimum is therefore determined by the condition:

$$dE^2/dx = 0$$

$$dE^2/dx = 2A^TAx-2A^Tb=0$$

$$A^TAx \quad = A^Tb$$

Finally the best approximation will be given by the least square equation:

$$x= (A^TA)^{-1*}A^Tb$$

4.2.1 Numerical computing stability

In theory an equation system Ax=b will be solved by Gaussian elimination. However in practice, this is not in every case a suitable method. The computed solution can easily be incorrect due to the fact that a computer can not handle an infinite number of digits. At each computing operation, the result must be considered from the point of round off error. For example, a computer which only can handle three digits will round off a computing operation like this:

$$0.437 + 0.00159 = 0.438$$

As the example shows, the impact of the last two digits in the smaller number has not affected the solution. If round off error like this are multiplied over and over again it can have a significant impact on the accuracy of the final solution. Each individual round off error will contribute to the final solution. Due to this fact, the development must be considered to the following two facts

- Ill conditioned matrices are highly sensitive too small changes.
- Weak computing algorithms can ruin well conditioned matrices.

The theoretical equation in order to solve a linear least square problem is given by the normal equation:

$$x= (A^TA)^{-1*}A^Tb$$

However, from a practical computing point of view, this function is not optimal. Solving a linear equation system by the matrix inverse operation requires a lot of computing operations. On top of that, it is also less accurate than Gaussian elimination. The fact is that the normal equation, i.e. the product of product is always less well conditioned than the original over determined equation system. The condition number for the normal equation is given by:

$$\varepsilon(A^TA) = \varepsilon(A)^2$$

This shows the condition number will be squared which can have a significant impact on the final solution.

4.2.2 QR factorization

In order to increase the numerical stability, QR factorization has been chosen. QR factorization creates one orthogonal matrix and one upper triangular matrix based on the matrix A. The two created matrices fulfil the following condition:

$$A=QR$$

In the equation above Q stands for an orthogonal matrix and R an upper triangular matrix. Due to the special properties of matrix QR, the substitute in the linear least square equation turns out as:

$$x= (A^TA)^{-1}A^Tb$$

$$A=QR$$

$$x= (R^TQ^TQR)^{-1}R^TQ^Tb$$

Due to Q is orthogonal, the product:

$$Q^TQ = I$$

Where matrix I stands for the identity matrix.

$$x= (R^TR)^{-1}R^TQ^Tb$$

Finally due to R is upper triangular and invertible, the product:

$$R^{-1}R=I$$

Finally, the following equation is derived.

$$x=R^{-1}Q^Tb$$

This equation provides advantages compared to the mathematical approach to the least square equation i.e. $x= (A^TA)^{-1}A^Tb$.

- The amount of computations is significantly reduced.
- The equation $x=R^{-1}Q^Tb$ is as well conditioned as the original problem.

4.2.3 Grahm-Schmidt orthogonalization

Determine the orthogonal matrix Q and the upper triangular matrix R is obtained by the Gram Schmidt algorithm. In mathematical theory, the Grahm Schmidt algorithm is an excellent method to create QR factorization. In practical computing, the algorithm is not numerically stable due to round off errors. One way to significantly reduce the influence of round off error is obtained by the MGS algorithm, i.e. Modified Gram Schmidt algorithm. Therefore the MGS algorithm has been chosen to create orthogonal and upper triangular matrices.

The MGS algorithm:

```
$a_k^{(1)} = a_{k,}$ $k=1:n$
FOR $k = 1:n$
    $r_{kk} = || a_k^{(k)} ||_2$
    $q_k = a_k^{(k)} / r_{kk}$
    FOR $j = k+1:n$
        $r_{kj} = q_k^T a_j^{(k)}$
        $a_j^{(k+1)} = a_{j,}^{(k)} - r_{kj} q_k$
    END
END
```

4.2.4 Back solve algorithm

The final solution of the vector x is determined by applying the back solve algorithm on the equation system $Rx = Q^Tb$. This is a fast and efficient method to solve an equations system which consists of an upper triangular matrix.

The back solve algorithm:

```
FOR $i = 1:n$
    FOR $j = 1+1:n$
        $b_i = - a_{ij} x_j$
    END
    $x_i = b_i/a_{ii}$
END
```

4.3 Evaluation of the centre of sphere algorithm

The evaluation has been conducted in Robot studio. The algorithms have been implemented in the robot program language Rapid.

4.3.1 Simulation of efficiency

In order to analyze the efficiency of the best fit method, simulation tests have been conducted in order to verify the accuracy. The simulation is based on the following expression of X, Y and Z in Cartesian coordinates.

$X = R*\sin\theta*\sin\varphi$
$Y = R*\sin\theta*\cos\varphi$
$Z = R*\cos\theta$

By choose radius, the angles θ and φ, a position on the sphere can be obtained. By adding a disturbance to the radius, a deviation at each position is obtained.

$X = (R+disturbance)*\sin\theta*\sin\varphi$
$Y = (R+disturbance)*\sin\theta*\cos\varphi$
$Z = (R+disturbance)*\cos\theta$

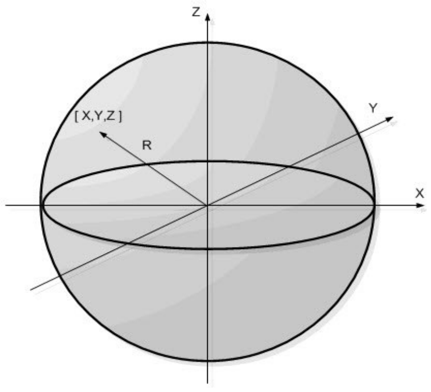


Figure 32
This figure shows a perfect sphere. The position on the surface of the sphere is given by a vector with length R which starts from the centre.

4.3.2 Simulation results

In order to verify the distribution of the results, the best fit method has been simulated. The simulation was conducted with applied disturbances to the radius. The disturbance is normal distributed and has a mean value equal to zero and a standard deviation of 0.01. (The disturbance values have been created by the data analysis function in MS Excel). The distribution of the accuracy is presented as the deviation in X, Y and Z axis values from the optimal value. In this simulation the optimal value is the given centre point without applied disturbance to the radius. This gives the equation:

$$\text{Euclidian distance} = \sqrt{((X_i-X_{centre})^2+ (Y_i-Y_{centre})^2+(Z_i-Z_{centre})^2)}$$

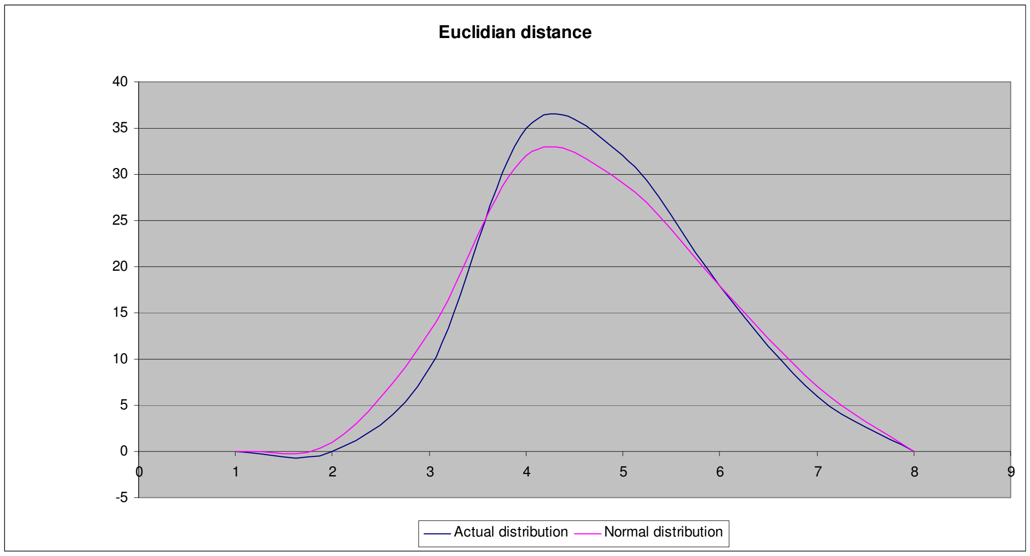


Chart 1

The histogram shows the distribution of data from the optimal point which in this case is zero. In the chart the blue line represent the actual distribution of the data, the pink line shows an ideal normal distribution. The normal distribution is based on the average value and the standard deviation of the actual data. The actual distribution deviates very little from the idea normal distribution. Due to this fact, the confidence interval of the distribution can approximately be treated as normal distributed.

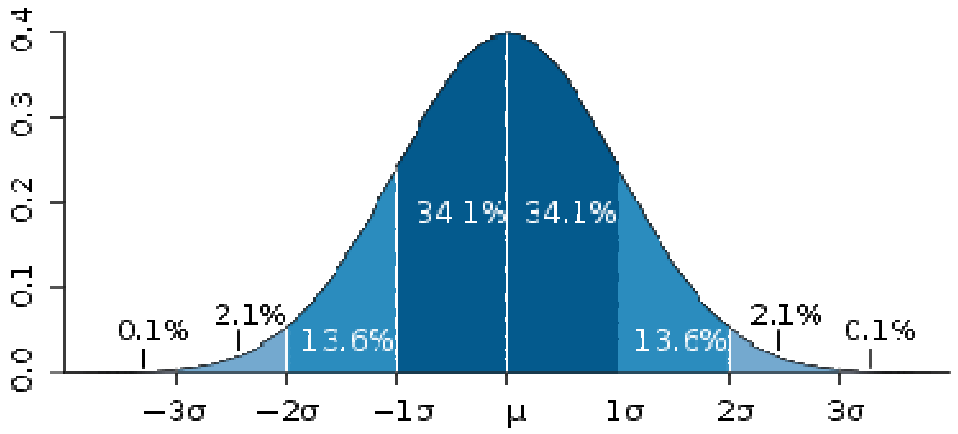


Chart 2
This chart shows the confidence interval of a normal distribution expressed as a function of the standard deviation σ.

As shown in chart 2, the values of a normal distribution are at 68.2% within +/- one standard deviation from the mean μ. However, when evaluating the efficiency of the best fit method, all values on the left side of the mean μ actually are closer to the optimal solution than μ. This is explained by the fact that the error is represented as the distance from the optimal value. In other words, 50% of the values are closer to the optimal value than the mean value μ. Due to this fact; the confidence interval then will be given by:

Mean value:	μ
Standard deviation:	σ

- $\mu \approx 0{,}1\% + 2{,}1\% + 13{,}6\% + 34{,}1\% \approx 50\%$
- $\mu + 1\sigma \approx 50\% + 34{,}1\% \approx 84{,}1\%$
- $\mu + 2\sigma \approx 50\% + 34{,}1\% + 13{,}6\% \approx 97{,}7\%$
- $\mu + 3\sigma \approx 50\% + 34{,}1\% + 13{,}6\% + 2{,}1\% \approx 99{,}9\%$

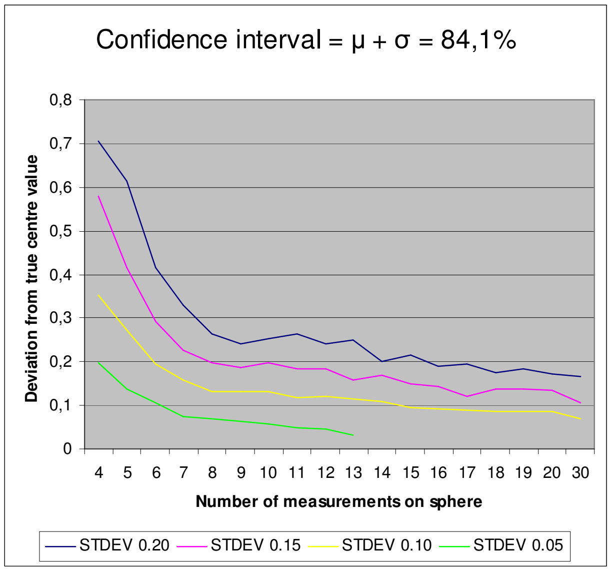


Chart 3.1
This graph shows the performance of the best fit method as function of measurements around a sphere. The deviation is presented at a confidence interval of 84%.

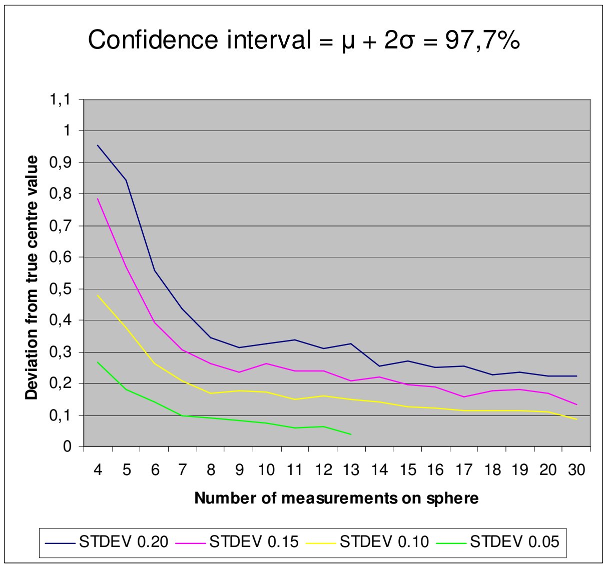


Chart 3.2
This graph shows the performance of the best fit method as function of measurements around a sphere. The deviation is presented at a confidence interval of 98%.

In the two charts 3.1 and 3.2, the efficiency of the best fit method has been simulated. Each of the two graphs shows the confidence interval with four different disturbances applied. The four disturbances are all normal distributed and have standard deviation 0.05, 0.10, 0.15 and 0.20. The confidence interval is presented as a function of number of measurement points on a sphere. The ideal value is zero which means the closer deviation is to zero, the better is the performance of the method.

As shown in the charts, the slope is rather steep until eight measurements. In other words, the improvement of the accuracy is most significant up to this number of measurements. This means that more than eight measurements will not improve the accuracy at the same rate. This is more or less valid for all four simulated disturbances.

The amount of measurements which is needed depends on the accuracy of the robot and the required accuracy for the application. However, in order to benefit from the early improvements, it is recommended to use at least eight measurements at this particular application.

4.3.3 Gauss Newton iterations

In order to determine how many iterations the Gauss Newton algorithm in general requires for this application, the convergence of the method has been monitored. The evaluation was made with parameters which correspond to the application. In chart X, the convergence of the function is plotted as function of iterations. The values are presented as the result of the following function.

$$\text{Value} = \sqrt{(X_{iX}{}^2 + Y_{iY}{}^2 + Z_{iZ}{}^2 + P_{iR}{}^2)}$$

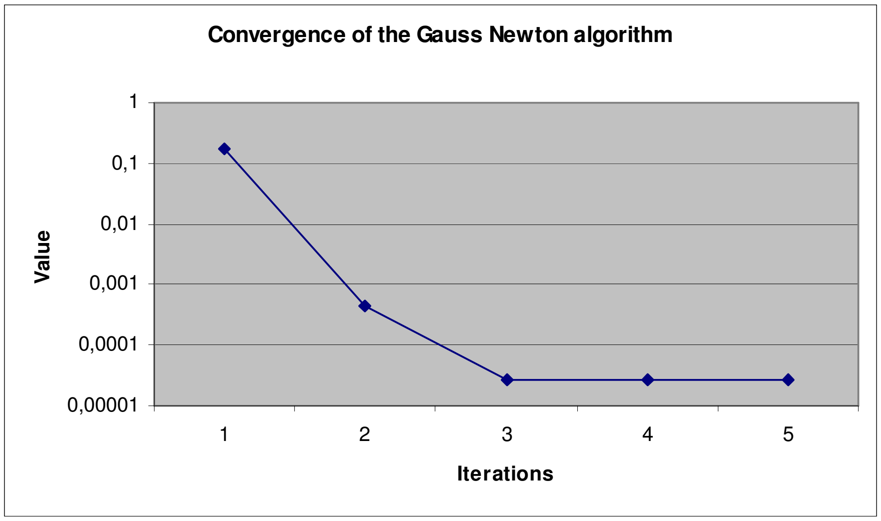


Chart 4

This chart shows the convergence of the Gauss Newton as a function of iterations when used in this application.

As shown in chart 4, the Gauss Newton algorithm converges rapidly when applied in this application. Already after two iterations, the convergence values are smaller than 0,001. In other words, further iterations will refine the final values with accuracy smaller than 0,001mm. Therefore, further iterations will insignificantly improve the solution; the found optimal value is good enough. Sufficient convergence therefore has been achieved.

4.4 Calibration of tool centre point

In order to achieve the best possible result of measured positions, it is crucial that the TCP (Tool centre point) is as correctly defined as possible. In other words, it has to be defined in the very centre of the robot spherical probe tool.

The TCP is where the controller will read the position. In other words, the position of the tool centre point will be expressed in the robot's Cartesian coordinate system in three dimensions X, Y and Z axes.

If the TCP has been inadequate defined, the read positions in space will not be trustful. The radius will not be constant and the determined centre point will be incorrect. This deviation is visually shown in figure 33.

However, under the condition that the TCP is perfectly defined in the centre of the sphere probe, every given point around the measured sphere will in theory have identical distance to the centre of the measured sphere. In other words, the radius to the centre point for every read position then will be the identical.

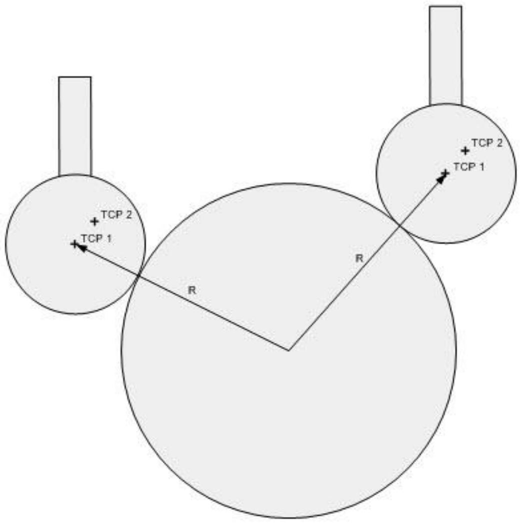


Figure 33
This figure shows different definition of the TCP in the spherical probe. The spherical probe is mounted on the robot's tool mounting flange.

- TCP 1: Well defined TCP in the centre of the sphere
- TCP 2: Inadequate defined TCP, not in the centre of the sphere.

However, in reality it is not very likely that the TCP will be perfectly defined in the centre of the spherical probe. As a consequence, each received position around the measured cell sphere will consist of a deviation from the true ideal position. Based on this fact, this deviation has to be taken in to consideration when the final centre position has been determined.

4.4.1 Calibration of TCP

The TCP of the sphere probe tool will be given by letting the robot push in to a fixed hole while soft servo is activated. By using soft servo, the spherical robot tool will be directed and finally seated in the centre of the hole. The contact will be established without trigger a collision alarm. In other words, the soft servo allows deviations from the defined programmed trajectory by deactivating the proportional part of the PID position control. By repeating this procedure with different orientations of the tool and read the final position the TCP will be defined.

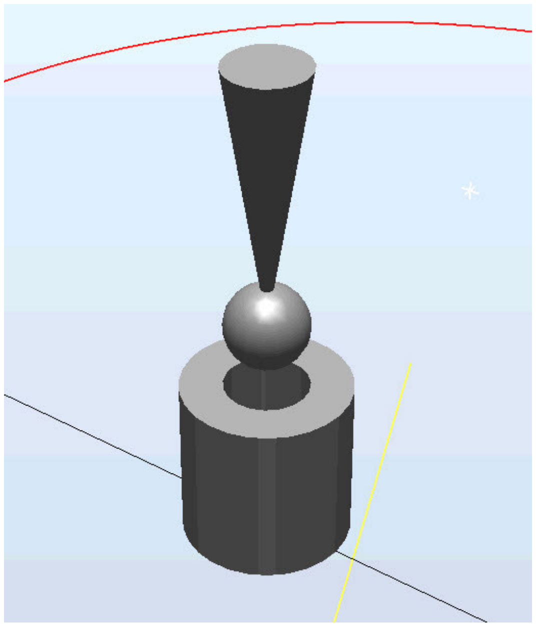


Figure 34.1
The spherical probe is positioned above the calibration cup.

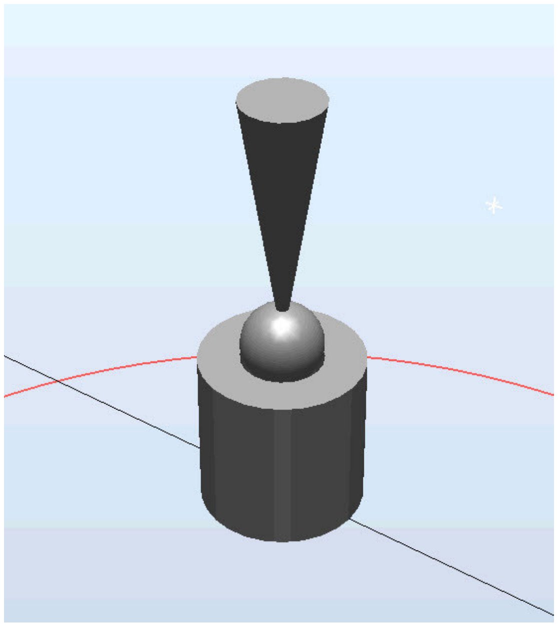


Figure 34.2
The spherical probe is pushed down in the calibration cup.

Under condition that the TCP (Tool Centre Point) is correctly defined, the received positions in Cartesian coordinates shall be identical or have insignificant difference among the received positions. As already mentioned, the received robtarget positions will have identical centre point; however the orientation of the tool will vary.

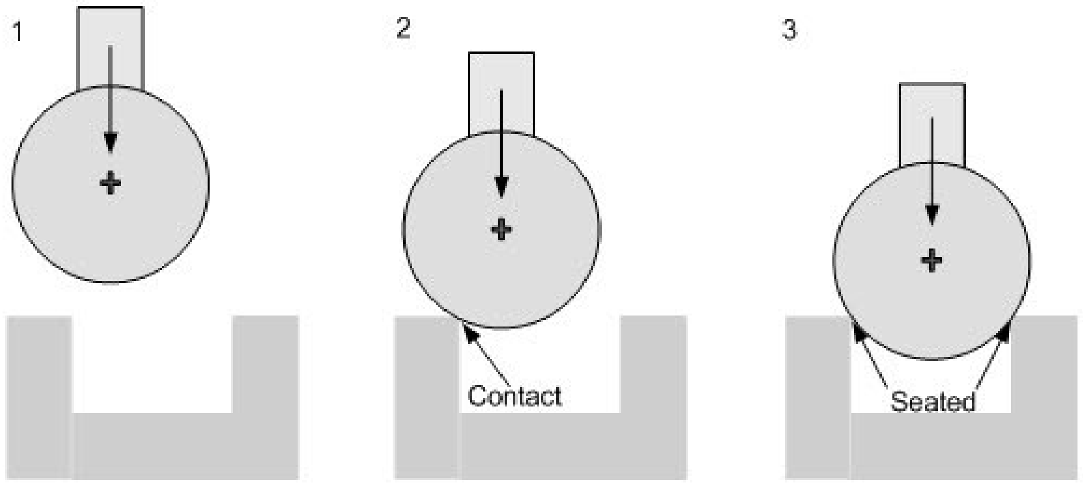


Figure 35
As shown in this figure, the spherical probe is positioned above the fixture. By activate soft servo the probe is moved down in to the fixation. The probe is guided by the edges of the fixture in to the centre. Finally, the probe will be seated in the centre of the fixation. Due to the probe is spherical; the position in Cartesian coordinates of perfectly defined TCP shall be the identical independently of the orientation of the probe. This implies that the spherical probe is properly seated in the fixture.

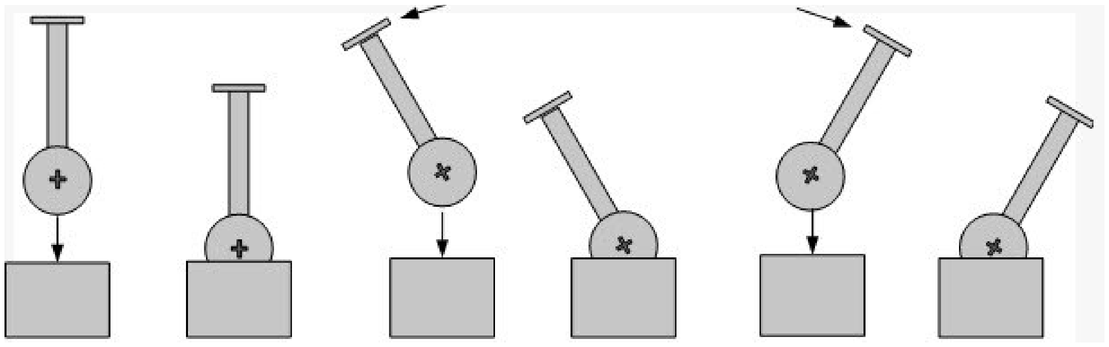


Figure 36
In this figure, the reorientation of the spherical probe is shown. At each reorientation, the probe is pushed down in the calibration cup. By repeating this process at least four times, a TCP in the centre of the probe sphere can be determined.

Under the condition that the tool centre point is not defined in the centre of the spherical probe, the received positions in Cartesian coordinates from the TCP will not be identical. The received points will build a portion of a sphere. The centre point to this portion of sphere will correspond to centre point of the probe. In other words the true centre point of the robot tool sphere is determined when the tool is properly seated in the hole.

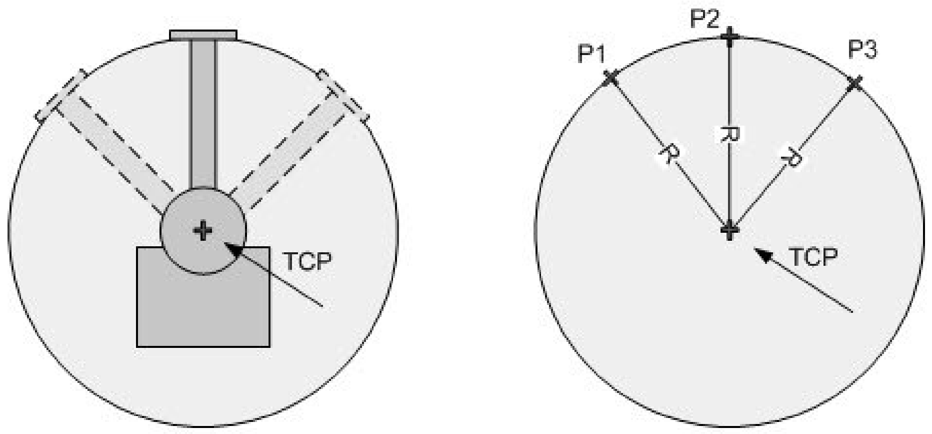


Figure 37
In this figure, the tool centre point calibration principle is shown in two dimensions. By reorient the spherical probe, the robot will be positioned at equivalent radius. These three positions will build a part of a circle. By using these positions, the centre of the circle can be determined. The method is equivalent for three dimensions; instead of determine the centre of circle the method is determining the centre of a sphere.

Due to the physical limitations of orientating the probe sphere, the positions will only be given on a part of circle. In other words, the impact of measurement error is significant greater than if three positions were retrieved on larger span around the circle.

In the calibration of the centre point of the probe which is in three dimensions, the probe is reoriented in to number of different positions. Each measurement positions are obtained by store the position of tool0. In other words, the origin of the robot's wrist coordinates system expressed in the world coordinate system.

When defining a robot tool, the dimension has to be expressed in the wrist Cartesian coordinate system, see figure 38. This means that the centre point of the spherical probe tool has to be defined in this Cartesian coordinate system.

However, the centre point can only be determined when the spherical probe is properly seated in calibration position. On top of that, this centre point is expressed in the world coordinate system.

The defined sphere centre point will also define the TCP. Finally the point will be given by the difference between the location of origin of the wrist coordinate system in the world coordinate system and the location of the determined centre position defined in the world coordinate system, see figure 39. The tool0 position with smallest error will be used in order to determine the position of the TCP.

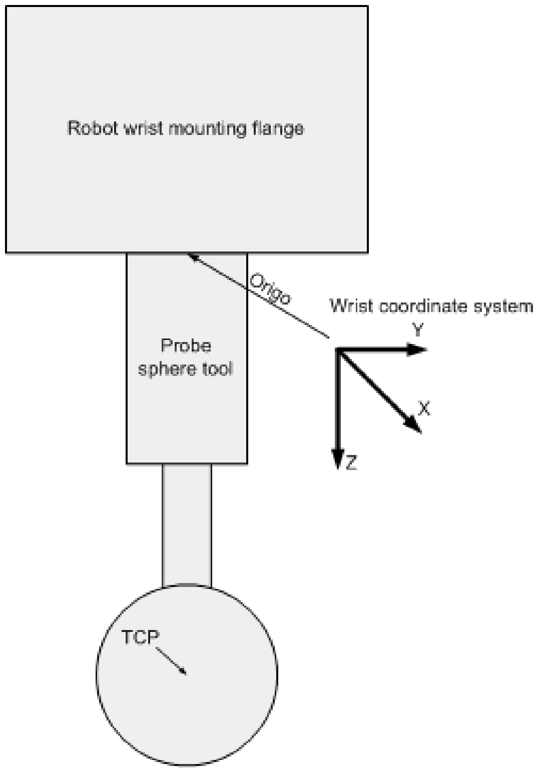


Figure 38
Cross section of the spherical probe tool mounted on the mounting flange of the robot tool.

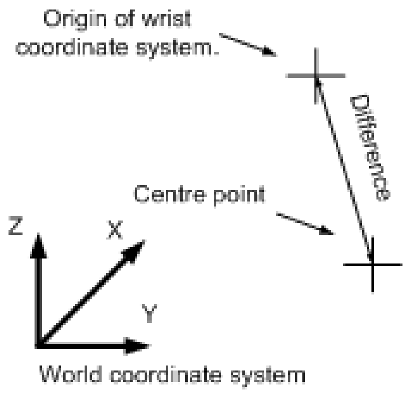


Figure 39
By compare the best fit of the tool0 and the determined position, the tool centre point can be defined.

4.5 Centre of sphere

In this section, the used method to determine the centre of fixed sphere is presented.

4.5.1 Measurement positions

In order to determine the centre of the sphere, at least four Cartesian coordinate positions around a sphere are required. The radius to the centre of the fixed sphere has to be more or less identical for all positions. By moving a sphere probe in to contact with a fixed sphere these positions can be obtained. By activating soft servo on all robot axes, the spherical probe mounted on the robot mounting flange will smoothly get in contact with an object. The contact will be established without trigger a collision alarm. In other words, the soft servo allows deviations from the defined programmed trajectory by deactivating the proportional part of the PID position control.

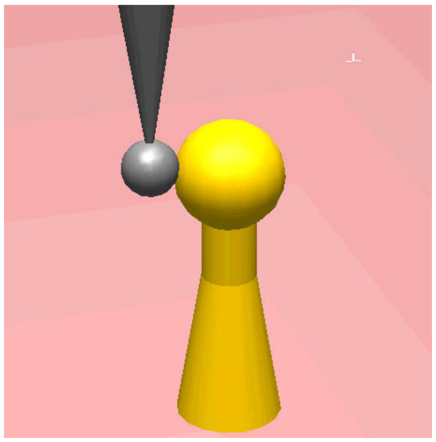


Figure 40
This figure shows the spherical probe and the fixed sphere. The spherical probe is mounted on the tool mounting flange.

The robot is programmed to move the spherical probe to different positions around the sphere. By using more than four positions, the impact of position deviation errors can be reduced. In order to reduce the impact of the play in the gear trains and measurement errors, the positions around the sphere have been distributed in order to get as large span around the sphere as possible.

The process starts by receiving a reference position. This position will be obtained by manually move the spherical probe in to position. This position is vertically above the fixed sphere, as shown in figure 41. The probe has to be positioned very close to the sphere. The orientation of the spherical probe has to be set in order to correspond to the normal of the surface of the fixed sphere.

This means that that the nominal centre of the fixed spheres more or less is located in positive Z-axis direction in the tool coordinate system. Based on this position a reference point which is located near centre of the sphere is set. This has been made by adding a set distance in positive Z-axis of the reference position.

The reference positions around the sphere are obtained by the formulas.

X = R*sinθ*sinφ
Y = R*sinθ*cosφ
Z = R*cosθ

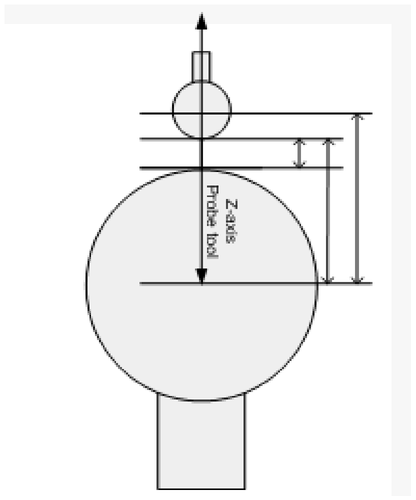


Figure 41
This figure shows the spherical probe positioned above the fixed sphere.

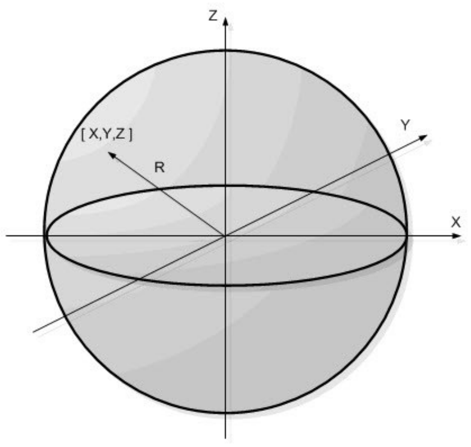
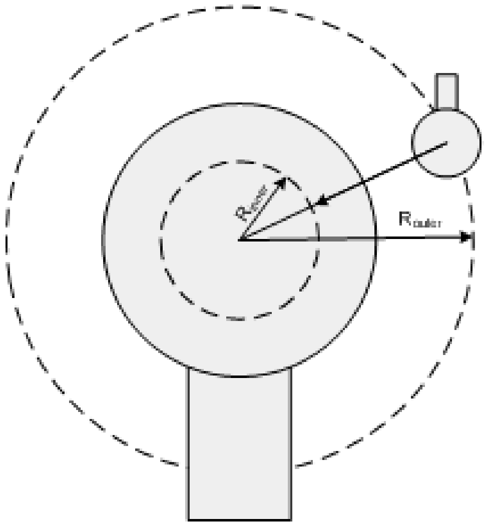


Figure 42
Cartesian coordinates expressed as the radius from a centre point.

The probe is moved to the start position by choosing θ and φ angles and an outer radius. When in position, the probe will be moved in a vector which point at the nominal reference centre position. Finally the probe will get in contact with the fixed sphere. The vector has been obtained by using the same θ and φ angles but with decreased radius. This radius has been set within the radius of the fixed sphere. In other words, the probe will be moved towards the centre until it gets in contact.



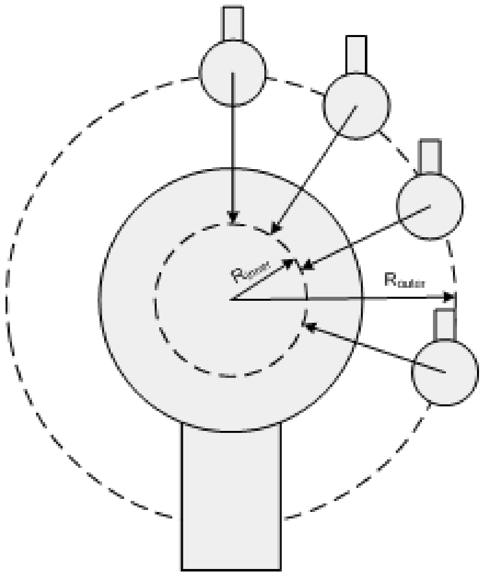


Figure 43.1
This figure shows how the probe is approaching the reference point which is set approximately.

Figure 43.2
The probe starts on top of the sphere. Last measurement will be made at angle of 110degres in relation to the start position.

By repeating this process with various angles, the positions around the sphere are obtained. The φ angle which determines the position in the x, y plane will increase 80º after each measurement. The θ angle which determines the position in Z-axis is set to a maximum of 110 º. By dividing this value by the number of measurements; the positions are evenly distributed on the Z-axis. The pattern of the retrieved positions around the sphere will be distributed as a spiral spring. Minimum measurement positions the method can handle is 4 and maximum has been set to 100.

4.5.2 Soft servo

The soft servo built in function in Rapid permits to soften the robot axes. This is obtained by reducing the proportional effect of the robot's PID controller. As a consequence, the step response will be delayed. The soft servo function can be independently activated on each axis. Deviations from the programmed path will not trigger any alarm.

This function can be used to approach an object which has an unknown position but within a defined range. Instead of a sudden impact which normally would trigger a collision alarm and stop of program execution; the robot will smoothly get in contact with the surface of the object in the defined direction.

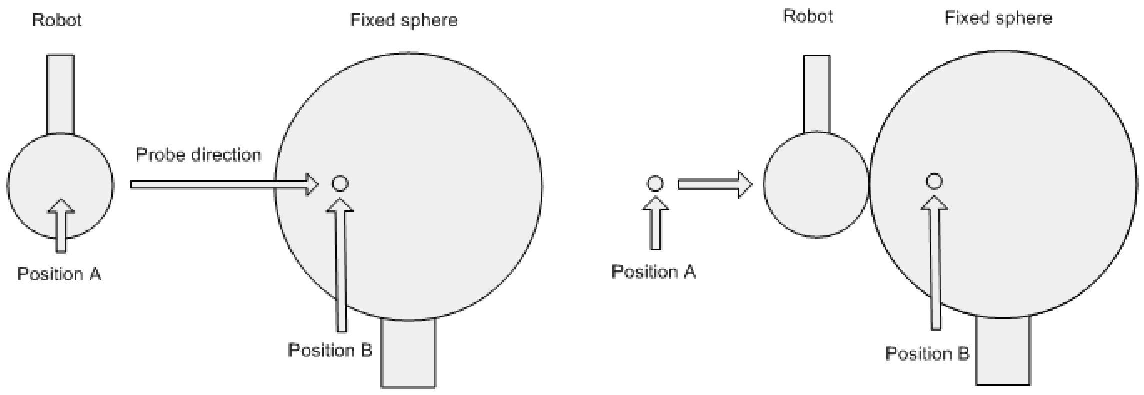


Figure 44
In this figure the function of the soft servo is shown. The robot is programmed to linearly move from point A to be B. Position B is located within the radius of the fixed sphere. In other words, the robot's probe is programmed to collide with the fixed sphere. Normally, this will trigger a collision alarm. However, by activating soft servo during the movement, the robot will move in to the surface smoothly without trigger a collision alarm. The applied contact force will increase as function of the distance between the contact position and the programmed position B.

5 Evaluation

In this section, the efficiency of the TCP calibration and the centre of sphere method have been evaluated based on repeatability tests. These tests have been conducted in actual test equipment.

5.1 *Test equipment*

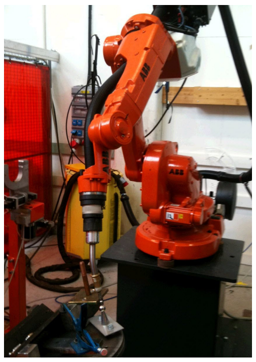


Figure 45.1
Actual test robot, IRB1600ID

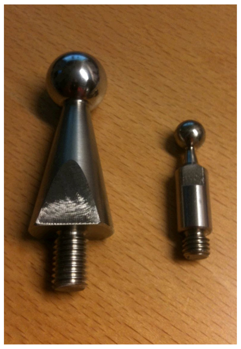


Figure 45.2
Test equipments.
The spherical probe has a radius of 5mm and the fixed sphere 10mm

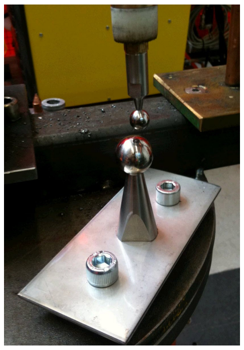


Figure 45.3
Installation of test equipment.

The repeatability and verification tests have been conducted by an IRB1600ID which is shown in figure 45.1. What makes this robot unique is that the process cable is routed within the upper arm and through the mounting flange. Due to this fact, this robot has been developed for mainly be used in Arc Welding and similar applications.

According to the product specification sheet, the robot has a position repeatability of 0,02mm and a path accuracy of 0,48mm. However, the particular test robot did suffer a play in gear train of axis five. Neither does it have the option absolute accuracy. In other words, nonlinearities in axis movements have not been mathematically reduced.

Repetition test has been made by a sphere probe mounted as robot tool and a fixed sphere in steel. The spherical shape of these two pieces is considered to be perfect.

5.2 *Evaluation - TCP calibration*

Due to the fact that the tool centre point will have an impact on all measurements, it is important that the tool centre point is positioned in the very centre of the spherical probe as possible. Based on this fact, a great number of measurements positions have been used to evaluate this method.
In total, 80 measurement positions have been used to define the tool centre point at each repeat.

The positions have been evenly distributed according to the pattern which is shown by figures 46.1 and 46.2. The orientation of the sphere probe tool is at maximum leaned 25 degrees relative the tool's X, Y plane, see figure 46.1. The samples are retrieved by moving the orientation in a straight line in four different directions in the X, Y plane.

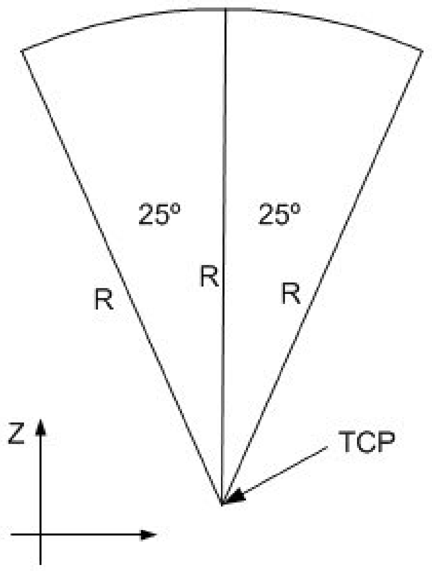


Figure 46.1
Reorientation of the tool in respect of the normal of the reference fixation hole.

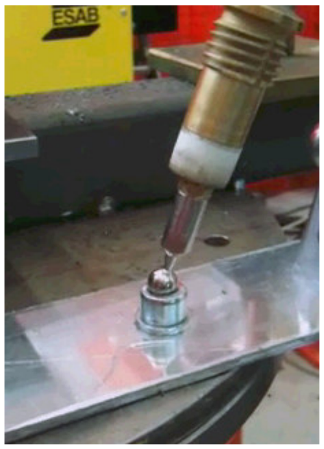


Figure 47
Reorienting tool in the reference fixation hole.

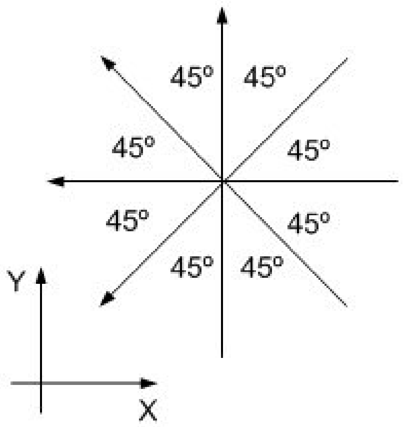


Figure 46.2
Measurement sample pattern showed in X, Y axis of tool coordinate system.

5.2.1 TCP calibration - test results

The result is based on 500 repeats of the calibration process. The distributions of the determined X, Y and Z of the TCP coordinates are shown in charts 5.1- 5.3. The values are compared with normal distribution based on the actual data from the measurements

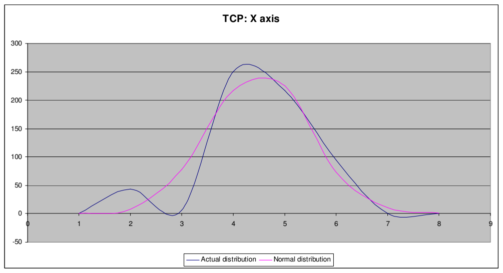


Chart 5.1
Histogram of deviations of the X-axis position value

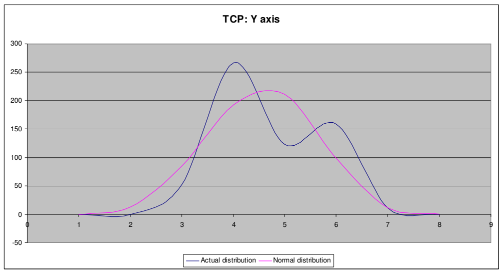


Chart 5.2
Histogram of deviations of the Y-axis position value

vv

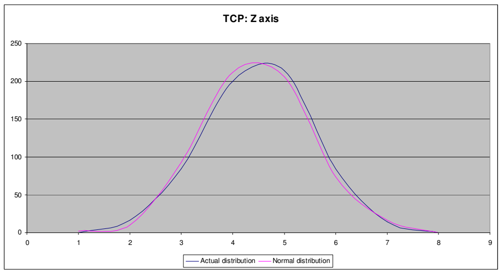


Chart 5.3
Histogram of deviations of the Z-axis position value

In charts 5.1 - 5.3 the blue line represents the actual distribution of the measurements. The pink line shows an ideal normal distribution based on the average value and standard distribution of the actual data.

As shown in charts 5.1 and 5.2, the X and Y values do deviate from an ideal normal distribution. The Z axis values have a distribution which is very close to an ideal normal distribution.

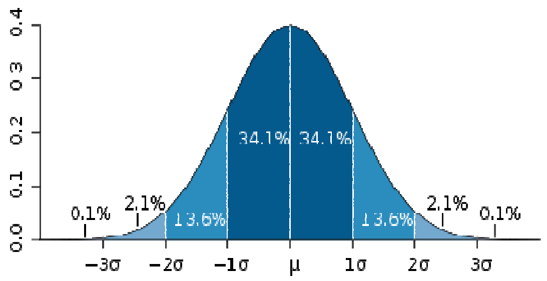


Chart 6
Confidence interval of a normal distribution as a function of the standard deviation σ.

X axis values

In table 1.1, the confidence interval of the measurements in X-axis is shown. The distribution is not ideal normal distributed. However, at two and three sigma the confidence interval of a normal distribution will be a fairly good approximation of the confidence interval.

Std dev	**TCP: X [mm]**
1σ ≈ 68.2%	N/A
2σ ≈ 95.4%	+/- 0.55
3σ ≈ 99.6%	+/- 0.81

Table 1.1

Y axis values

In table 1.2, the confidence interval of the measurements Y-axis is shown. The distribution is not ideal normal distributed. However, at two and three sigma the confidence interval of a normal distribution will be a fairly good approximation of the confidence interval.

Std dev	**TCP: Y [mm]**
1σ ≈ 68.2%	N/A
2σ ≈ 95.4%	+/- 1.80
3σ ≈ 99.6%	+/- 2.70

Table 1.2

Z axis values

In table 1.3, the confidence interval of the measurements Z-axis is shown. As the distribution is nearly ideal, the confidence interval can be approximated at one sigma.

Std dev	**TCP: Z [mm]**
1σ ≈ 68.2%	+/- 0.05
2σ ≈ 95.4%	+/- 0.10
3σ ≈ 99.6%	+/- 0.15

Table 1.3

TCP calibration – final evaluation

	Max dev	1σ ≈ 68%	2σ ≈ 95%
X-axis	+/- 0.10 [mm]	+/- 0.27 [mm]	+/- 0.55 [mm]
Y-axis	+/- 0.10 [mm]	+/- 0.90 [mm]	+/- 1.80 [mm]
Z-axis	+/- 0.10 [mm]	+/- 0.05 [mm]	+/- 0.10 [mm]

Table 2
This table shows the deviations of the defined TCP based on the conducted measurements.

As shown in the table, the desired max deviation has been defined to +/- 0.10mm. Only the Z-axis fulfils this requirement. Most likely, the measurements would have been more satisfying if another robot with less play in the gear train of axis five had been used. The physical limitations of the orientation of the spherical probe also cause the measurement range to be limited. If it was possible to measure on a wider range, the results would most likely be improved.

Based on these figures, this TCP calibration method of the spherical probe is not recommended. The presented deviations among the repeated measurements are too great.

5.3 Evaluation - centre of sphere

In order to evaluate the efficiency of the centre of sphere method in reality, repeatability tests have been conducted. These tests were done by repeatedly measure the fixed sphere with the spherical probe. The centre position and radius of the fixed sphere is estimated. The position of the fixed sphere is determined by X, Y and Z Cartesian coordinates in the robot's world coordinate system.

At these evaluation tests, the TCP of the spherical probe has been defined by the "Bulls eye" TCP calibration method. This calibration method is described in section 1.6.2.

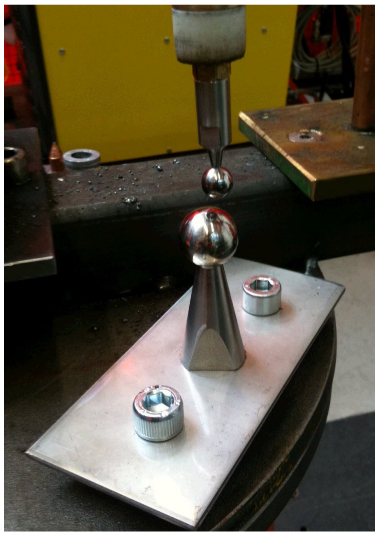


Figure 48
The installation for the repeatability tests.

5.3.1 Centre of sphere - constant start angle

In this section, charts of the measurements of the estimated centre point is shown in X, Y and Z axis, euclidian distance and the determined radius are presented. The charts represent data from 1000 repeats. The start angle of the approaching movements around the sphere is programmed to be identical at each repeat. At each repeat, eight measurements are obtained on the fixed sphere. The positions are obtained according to the description in section 4.5.1.

However, due to the effect of soft servo, the positions when the probe and sphere gets in contact with each other will somewhat deviate. This means that an inadequate defined TCP and the play at axis gear trains will have an impact. However, these variations are considered to be minor. Therefore, the impact of these deviations has been negligible at this repeatability test.

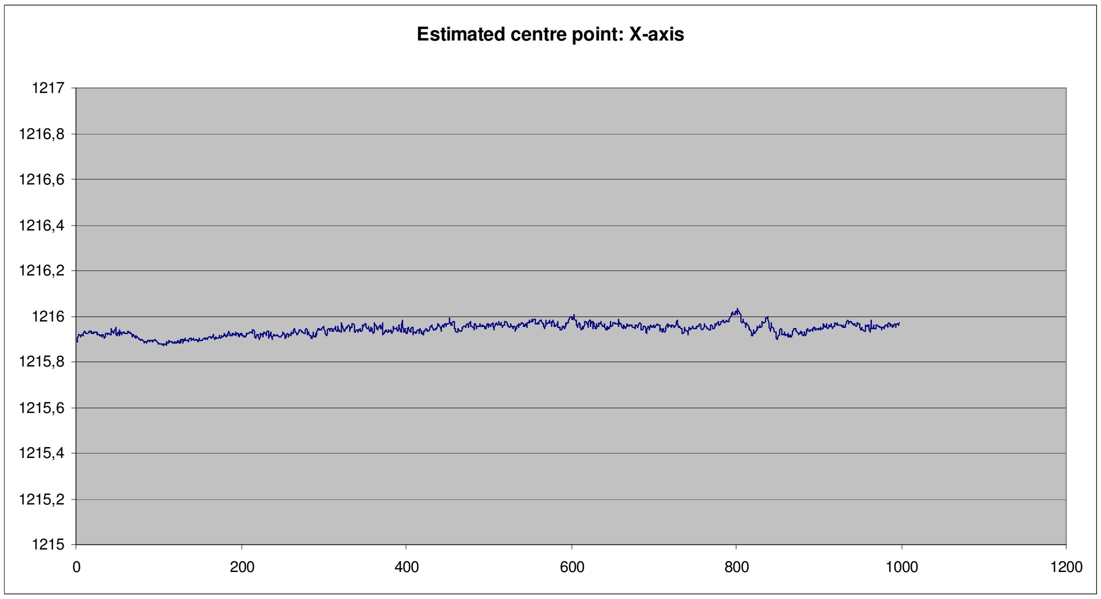


Chart 7.1
In this chart, the centre point X in axis of the 1000 repeats is shown.
The results show a stabile trend.

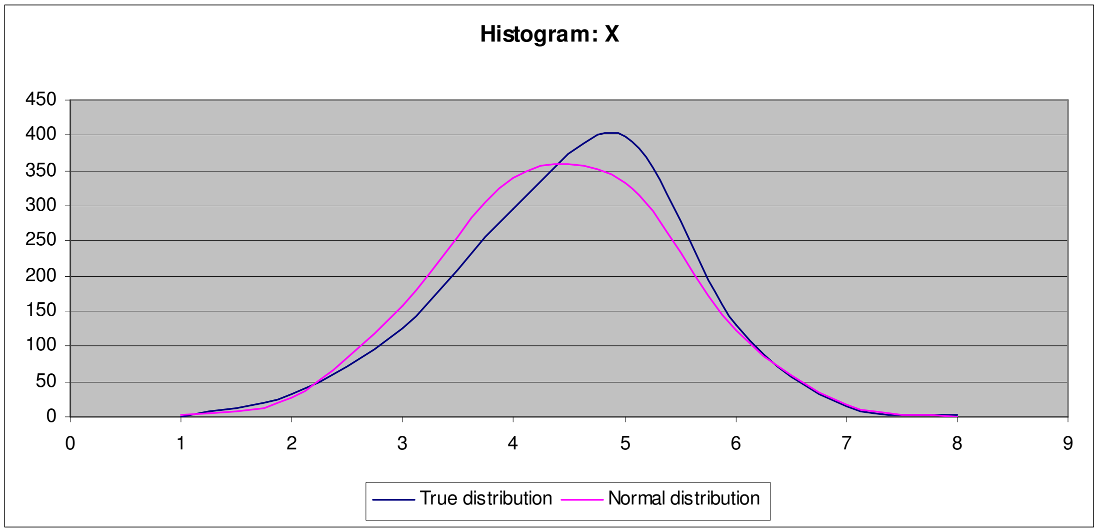


Chart 7.2
The shape of the histogram based on true values is nearly perfect normal distributed. Therefore, the evaluation has approximated the results as a normal distribution.

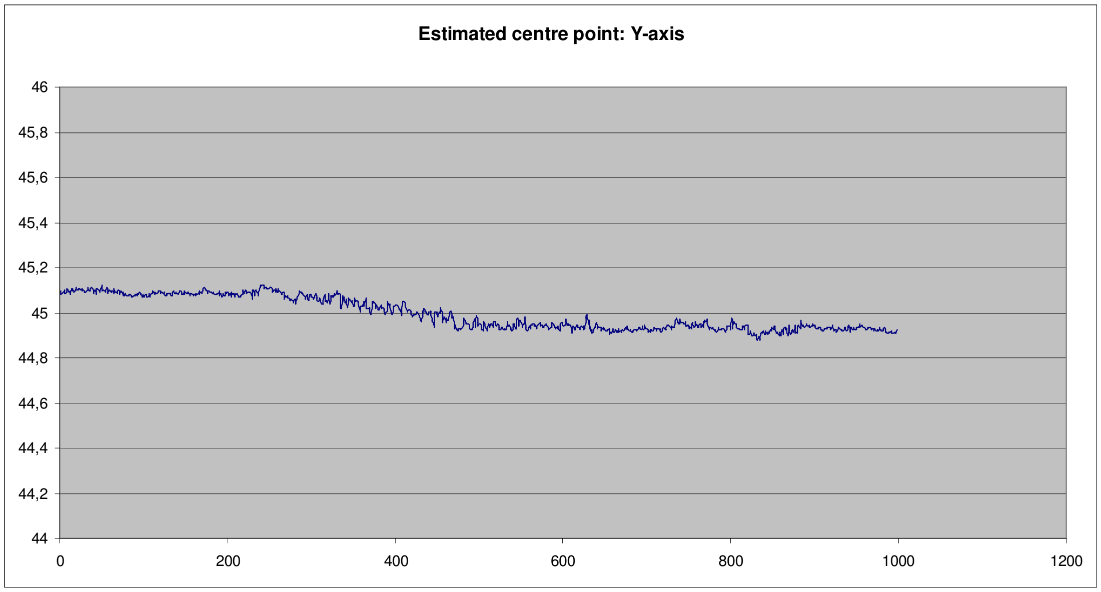


Chart 8.1
In this chart, the centre point of the Y axis value of the 1000 repeats is shown. At about 300 measurements the pattern starts to descend until about 500 measurements. A part from this deviation the trend is stabile.

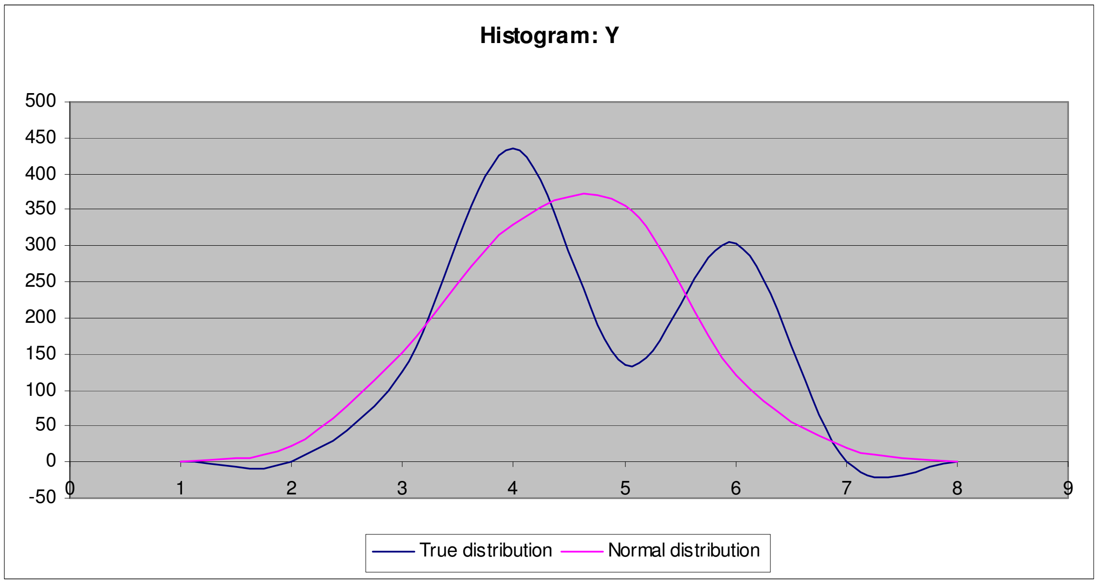


Chart 8.2
The shape of the histogram based on true values is not normal distributed. However at two sigma, the normal distribution will be a fairly good approximation.

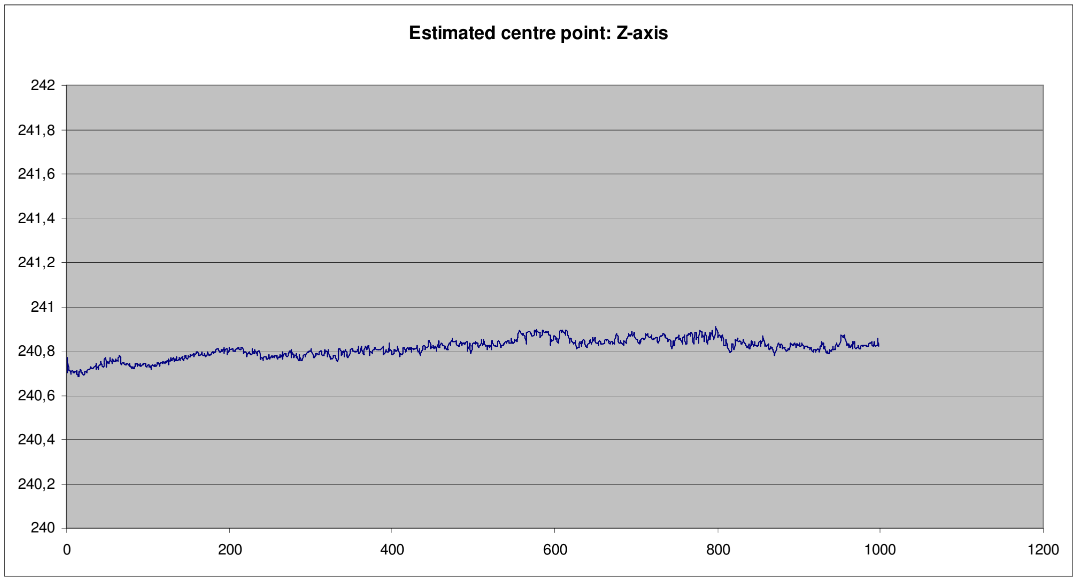


Chart 9.1
In this chart, the centre point Z value of the 1000 repeats is shown.
The results do show a stabile trend.

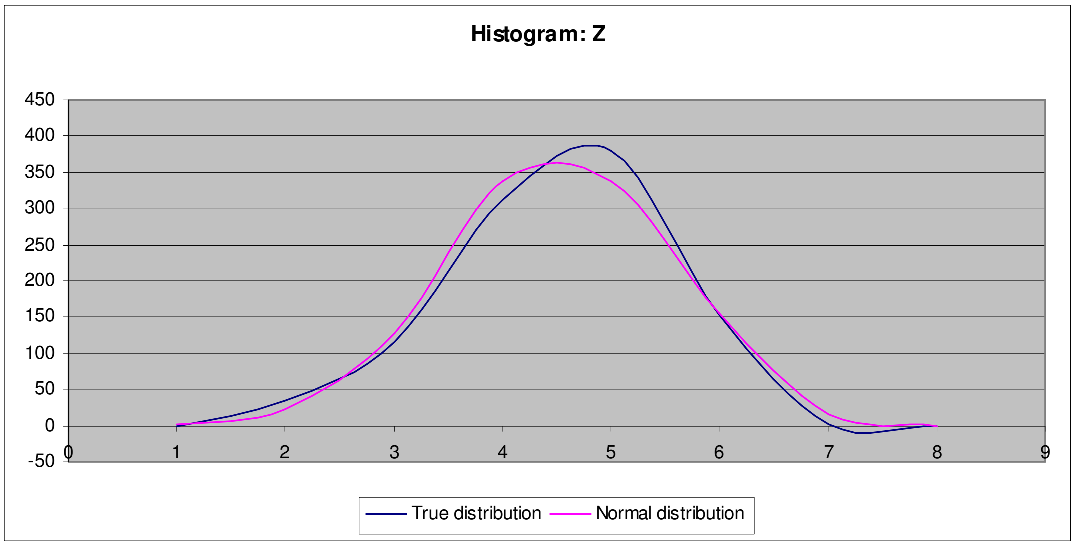


Chart 9.2
The shape of the histogram based on true values is nearly perfect normal distributed. Therefore, the evaluation has approximated the results as normal distributed

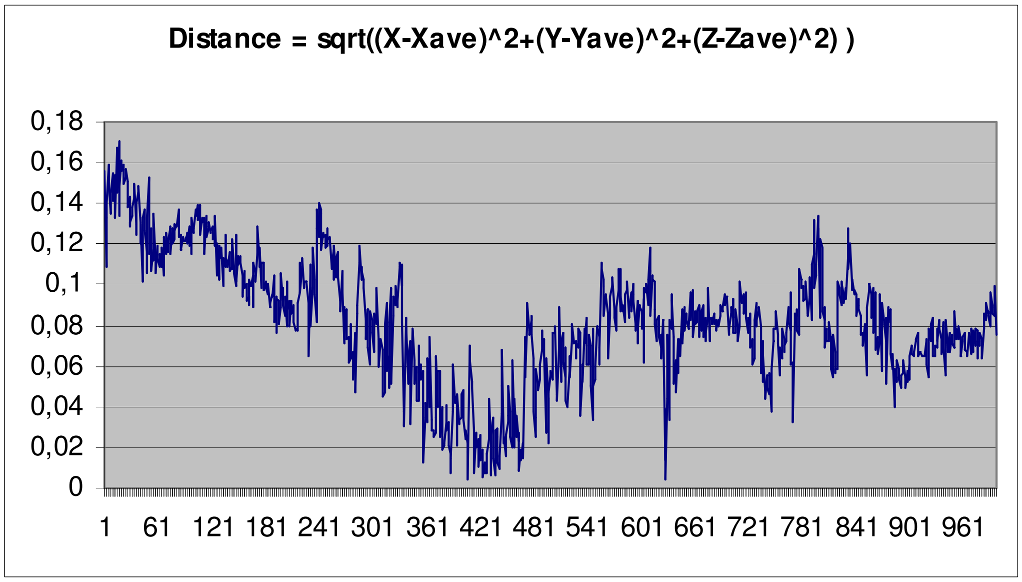


Chart 10.1
In this chart, the determined centre point is presented as the Euclidian distance based on 1000 repeats. The Euclidian distance is calculated from the average value of the X, Y and Z axis positions.

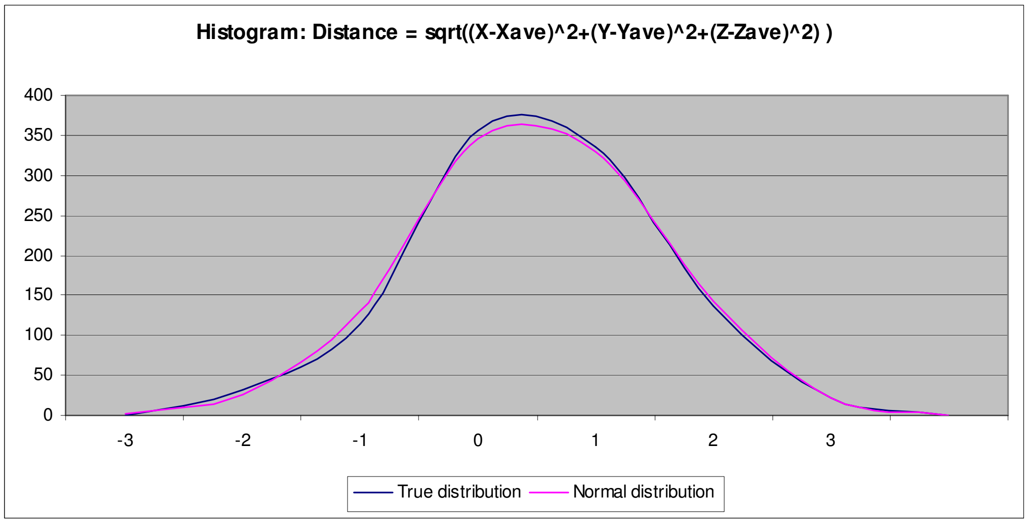


Chart 10.2
The shape of the histogram based on true values is nearly normal distributed. Therefore, the evaluation has approximated the results as normal distributed

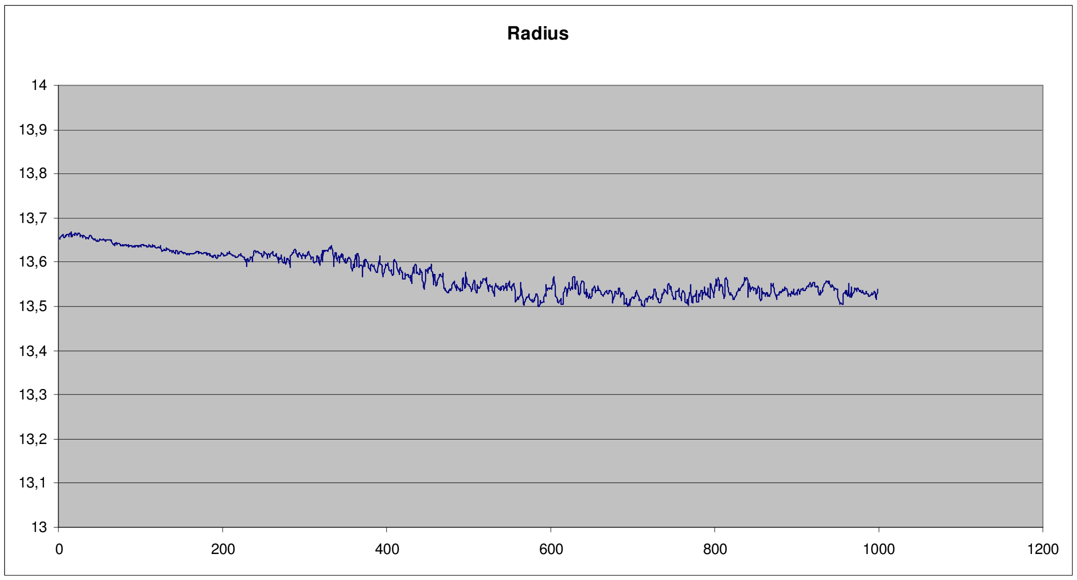


Chart 11.1
In this chart, the measured radius based on 1000 repeats is shown. The results decrease until 500 measurements. After 500 measurements the trend is stabile.

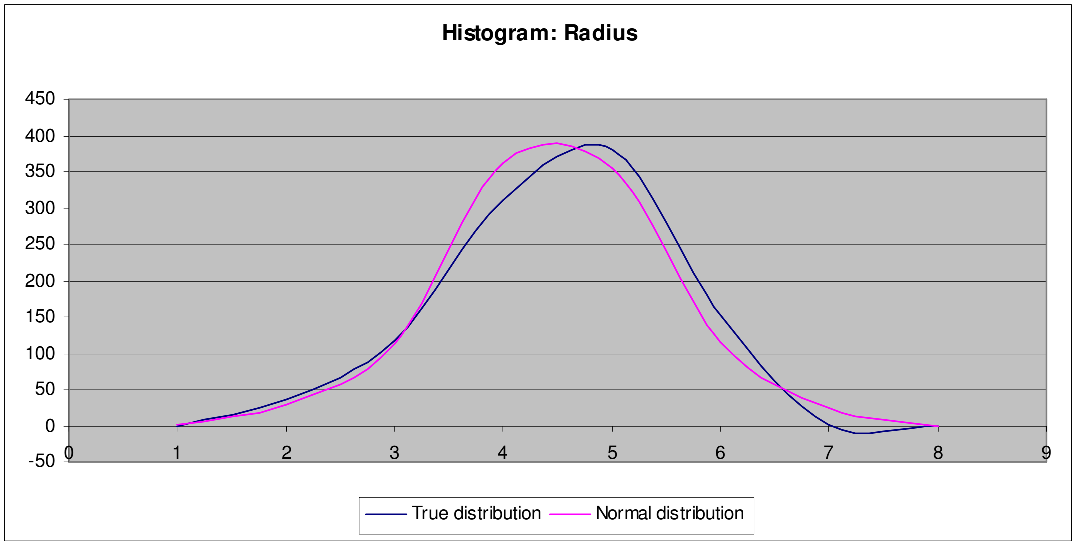


Chart 11.2
The shape of the histogram based on true values is nearly perfect normal distributed. Therefore, the evaluation has approximated the results as a normal distribution.

	1σ ≈ 68%	**2σ ≈ 95%**	**3σ ≈ 99%**
X-axis [mm]	±0.026	±0.053	±0.079
Y-axis [mm]	N/A	±0.144	±0.215
Z-axis [mm]	±0.043	±0.087	±0.130
Distance[1] [mm]	±0.031	±0.062	±0.093
Radius[2] [mm]	±0.046	±0.092	±0.139

1) Average euclidian distance measured to $0.082\ mm$
2) Average radius measured to $13.572\ mm$

Table 3
The table shows the measured deviation from the average values at the constant start angle repeatability test. The deviation is expressed in different confidence interval. As shown in the table the deviation is greatest at the Y-axis values.

As the average Euclidian distance from the centre point is 0,082mm. The figures in table 3 gives that the centre position is determined at a confidence interval of 84% within a radius of 0.113mm and at 98% within a radius of 0.144.

5.3.2 Centre of sphere - shifting start angle

In order to verify the influence of the play in the gear train of axis five and the potential impact of an inadequate definition of TCP, shifting start angles have been used. At this repeatability test, the start angle is increased by 45 degrees after each measurement. This means that after eight conducted measurements, the measurements will restart all over again at the initial start angle. In reality, this is not very likely that the calibration method will be used in this manner. The calibration will likely be used with constant start angle.

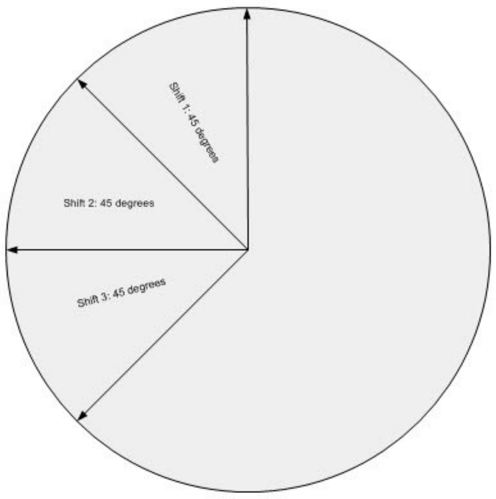


Figure 49
This figure shows the shifting start angles of the sphere probe.

In comparison to the repeatability test with the constant start angle test, inadequate definition of the TCP will have an impact in this test. The measurement positions will be different as the start angle shifts with 45º degrees.

This test will show how significant the play and a potential incorrect definition of the TCP will affect the estimated centre position. Depending on how the wrist is oriented against sphere, the play will have different influences on the estimated radius. However, the definition of the TCP will not have an impact on the measured radius. The results are based on 880 repeats. At each repeat, eight positions are obtained on the fixed sphere. The positions are obtained according to the description in 4.5.1.

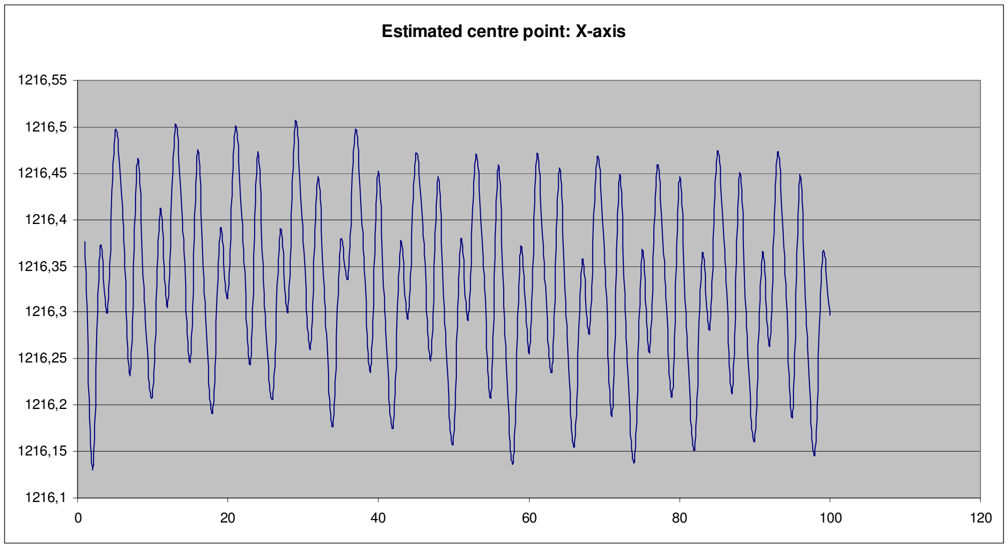


Chart 12.1
In this chart 100 of the 880 repeats are shown. The pattern is periodical. This is likely explained by the fact that the start angle increases by 45º after each measurement. This mean that the start angle returns to same angle after eight times. The results show a fairly stabile trend.

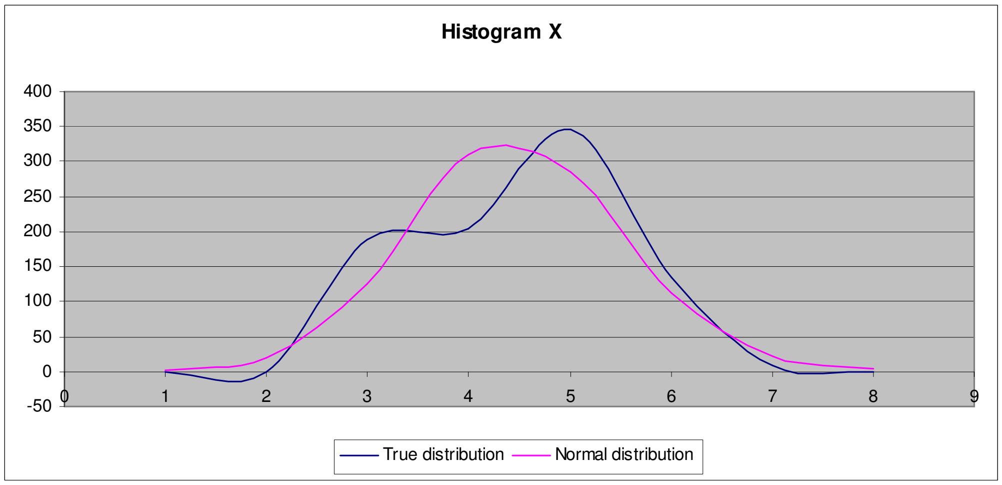


Chart 12.2
The shape of the histogram is based on the actual rue values. The shape is nearly normal distributed. In the evaluation, the results have been approximated the results as normal distributed.

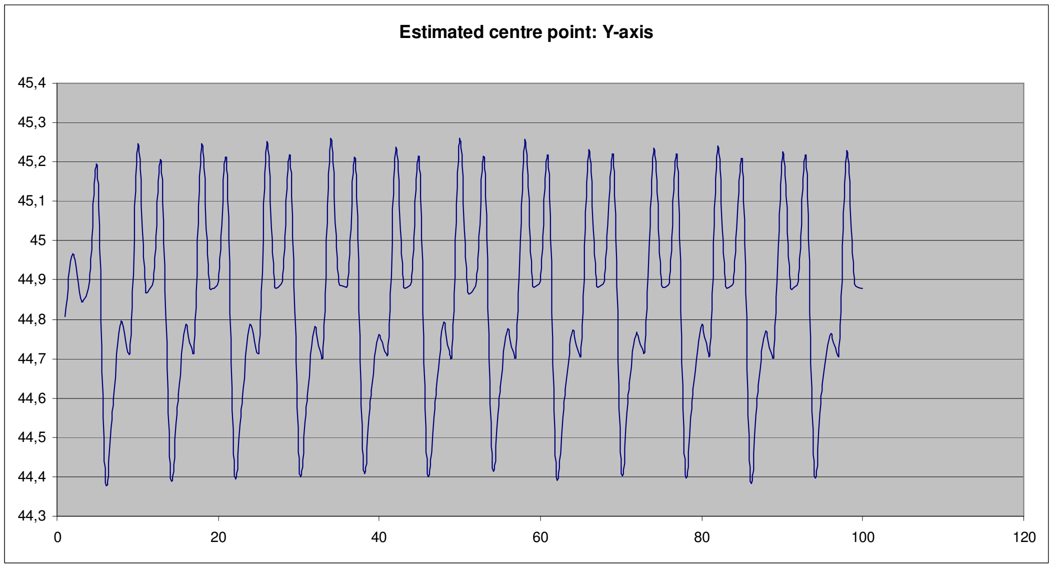


Chart 13.1
In this chart 100 of the 880 repeats are shown. The pattern is periodical. The periodical repeats are likely explained by the fact that the start angle increases by 45º after each measurement. This mean that the start angle returns to same angle after eight times. The results show a stabile trend.

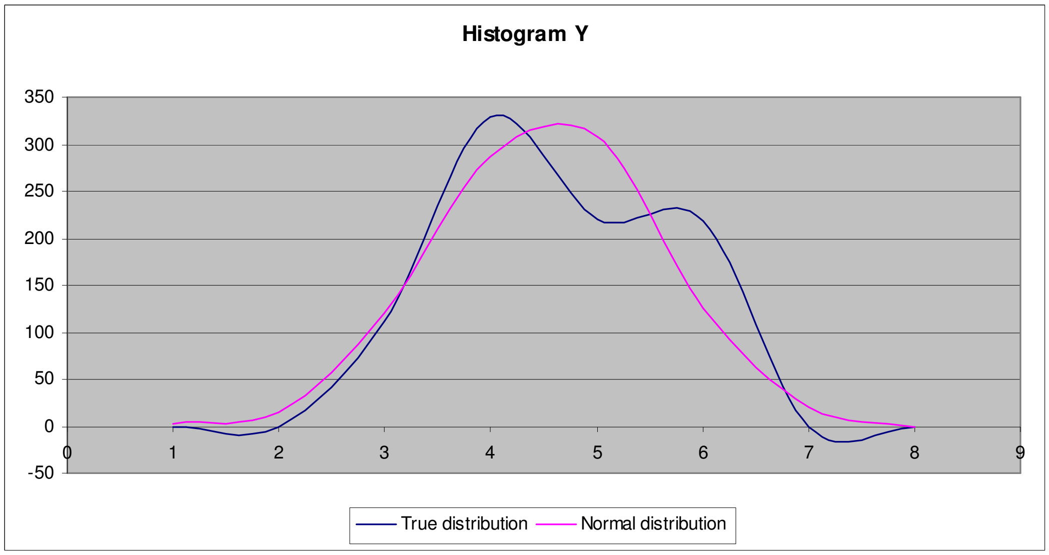


Chart 13.2
The shape of the histogram based on actual true values. The true values nearly form a normal distribution. The result has been evaluated the results as a normal distribution as an approximation.

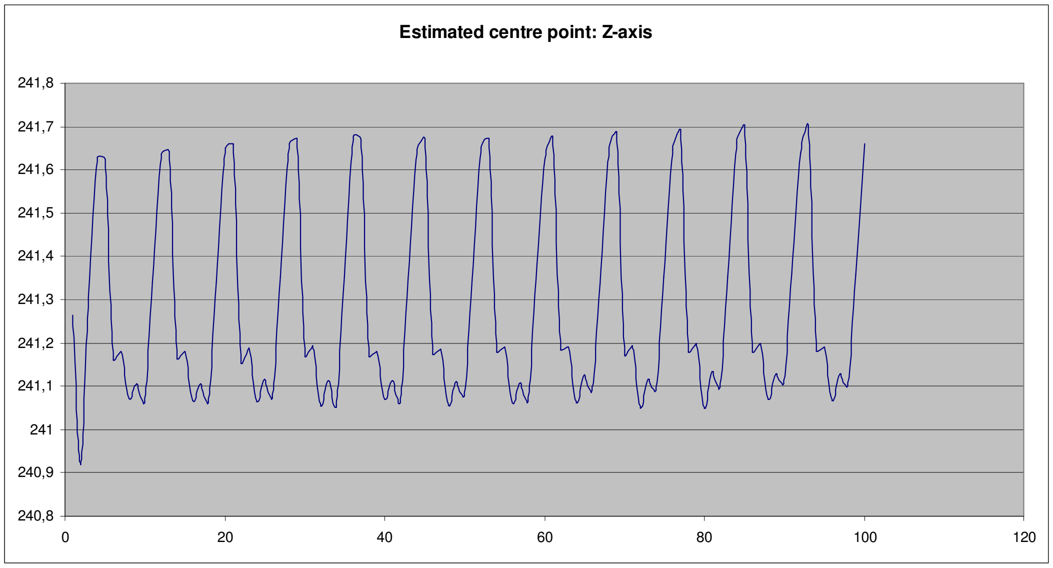


Chart 14.1
In this chart 100 of the 880 repeats are shown. The pattern is periodical, this is likely explained by the fact that the start angle increases by 45º after each measurement. This mean that the start angle returns to same angle after eight times. The results show slight increasing trend likely caused by thermal expansion

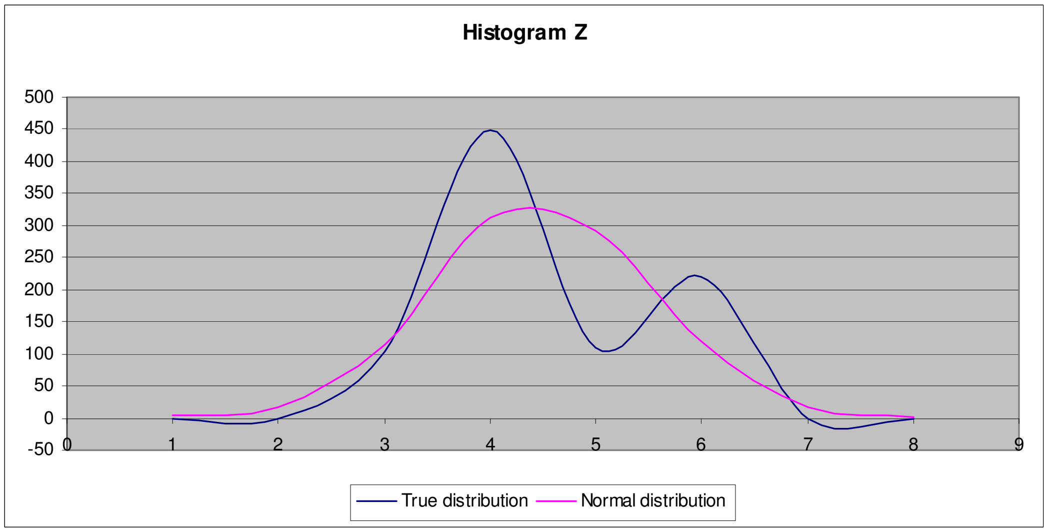


Chart 14.2
The shape of the histogram is based on true the values. The shape is not normal distributed. However at two sigma, the normal distribution will be fairly good approximation.

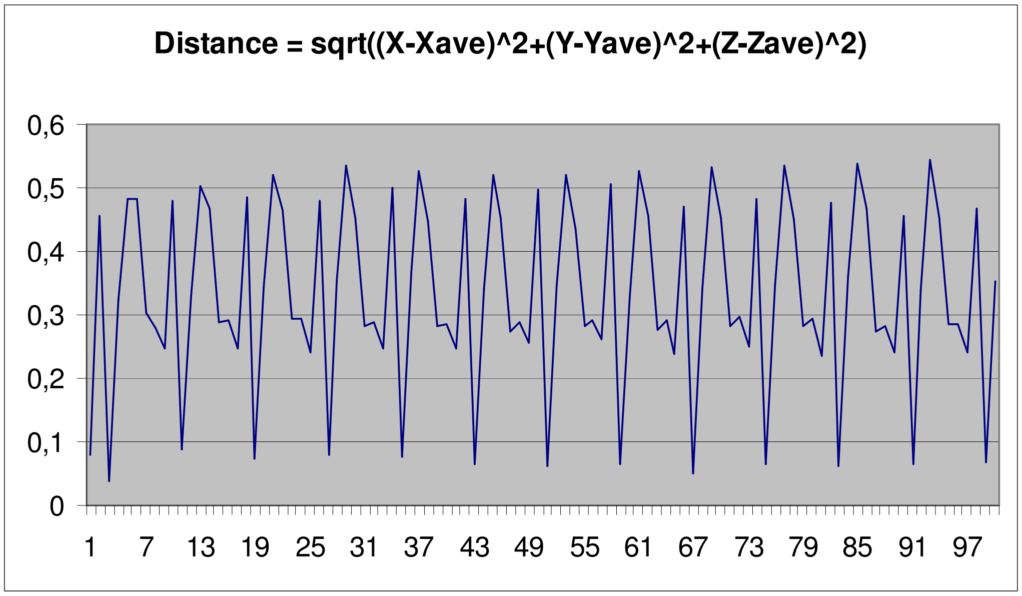


Chart 15.1
In this chart 100 of the 880 repeats are shown. The Euclidian distance is calculated from the average value of the X, Y and Z axis positions. The pattern is periodical, this is likely explained by the fact that the start angle increases by 45º after each measurement.

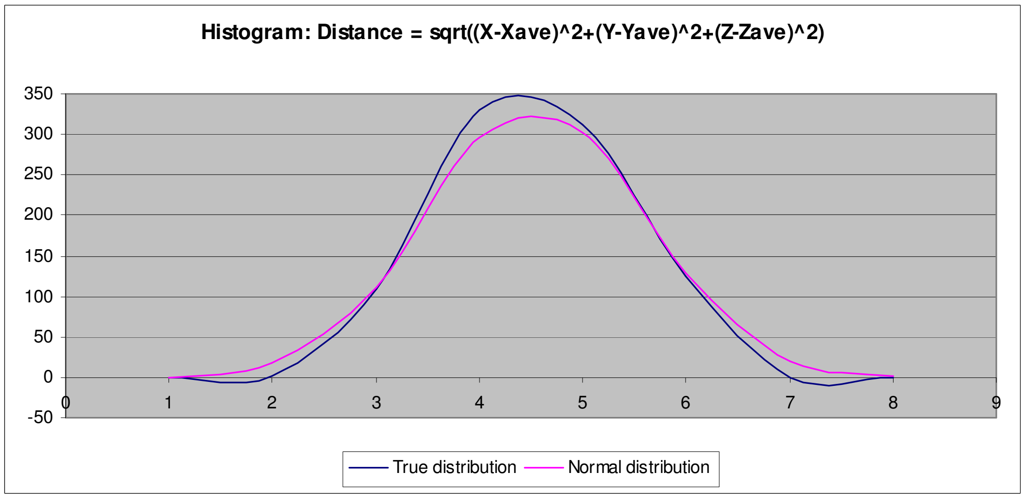


Chart 15.2
The shape of the histogram based on true values is nearly normal distributed. Therefore, the evaluation has approximated the results as normal distributed.

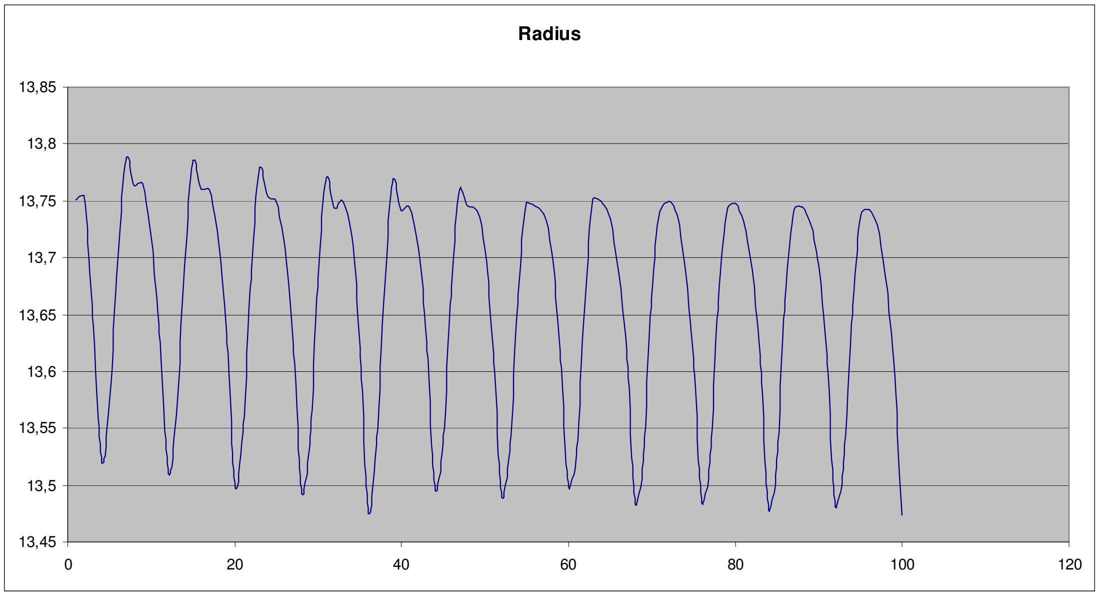


Chart 16.1
In this chart 100 of the 880 repeats are shown. The pattern is periodical, this is likely explained by the fact that the start angle increases by 45º after each measurement. This mean that the start angle returns to same angle after eight times. The results do shows a slightly decreasing trend.

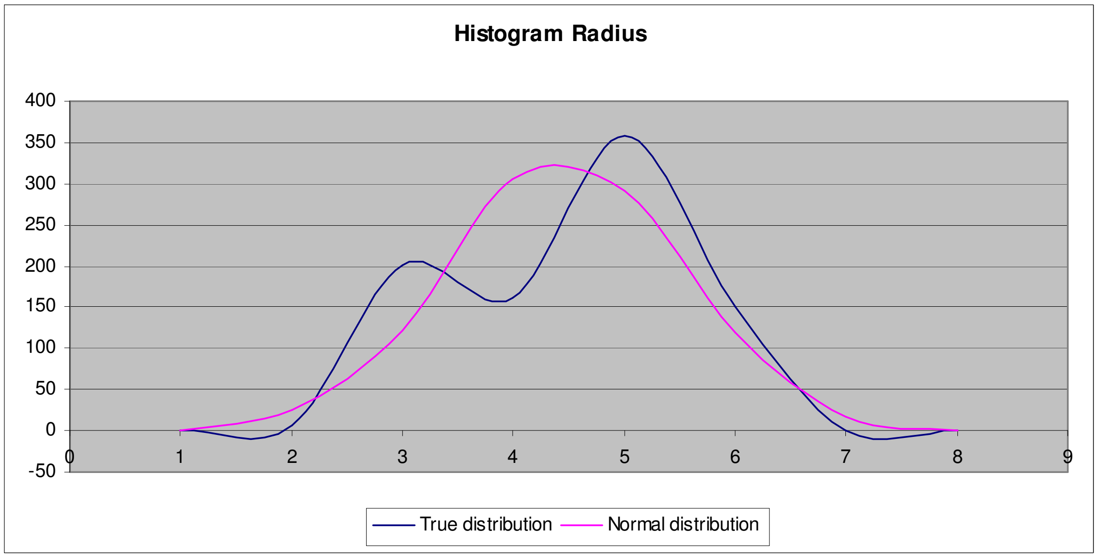


Chart 16.2
The shape of the histogram based on true values is not normal distributed. However, at two sigma or at 95%, the normal distribution will be fairly good approximation.

	1σ ≈ 68%	**2σ ≈ 95%**	**3σ ≈ 99%**
X-axis [mm]	±0.132	±0.270	±0.396
Y-axis [mm]	±0.245	±0.490	±0.735
Z-axis [mm]	N/A	±0.215	±0.323
Distance[1] [mm]	±0.136	±0.272	±0.409
Radius[2] [mm]	N/A	±0.107	±0.161

1) Average euclidian distance measured to 0.332 mm
2) Average radius measured to 13.623 mm

Table 4
This table shows the measured deviations with respect to the average values at shifted start angle. The deviations are expressed in different confidence interval. The figures show that the greatest deviations are in the Y-axis. This deviation is probably greatest due to the play in the gear train of axis five.

As the average Euclidian distance from the centre point is 0.332mm. The figures in table 4 gives that the centre position is determined at a confidence interval of 84% within a radius of 0.468mm.

Centre of sphere- comparison, constant vs. shifting start angle

At confidence interval 2σ ≈ 95%	Constant	Shifted	Difference
X-axis [mm]	+/- 0.053	+/- 0.270	I 0.217 I
Y-axis [mm]	+/- 0.144	+/- 0.490	I 0.346 I
Z-axis [mm]	+/- 0.087	+/- 0.215	I 0.128 I

Table 5

In this table, the deviations of the constant and the shifted start angle measurements are shown. The values are presented at confidence interval of 95%. As shown in this table, the deviation difference between the constant and shifted measurements is most significant for the X, Y and Z position values.

Among the determined centre position, the Y axis value has the largest deviation error. Compared to the X and Z values, the X value has a deviation which is more than 100% greater for the shifted measurements. Most likely this is explained by the orientation of the wrist during the measurements. At particular measurements the play in the gear trains will have more significant impact on the Y axis position values than compared to X and Y position values. It is also possible that the deviation has been caused by an incorrect definition of the TCP.

In charts 17.1-19.2, the constant and the shifted 45 degrees start angle is presented in three different planes X-Y, X-Z and Y-Z. The scale of the axes in the charts is more or less identical. In the three charts which represent the results from the 45 degrees shifting start angle measurements, clear isolated islands are shown. These islands represent the measurements at each periodic 45 degree start angle. The deviation within each isolated island is more or less equivalent to the deviation at the constant start angle. These results confirm that the method is robust.

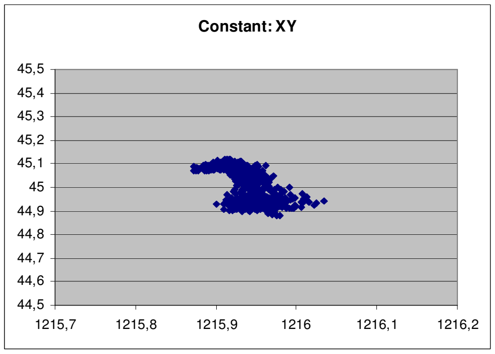


Chart 17.1
Deviation in X and Y axis at constant start angle.

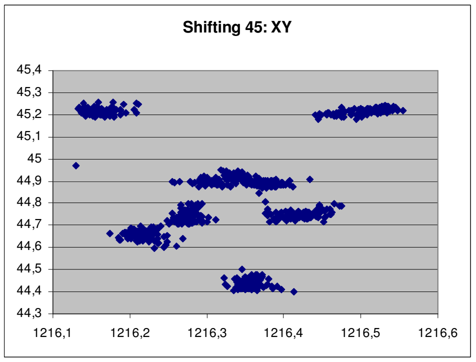


Chart 17.2
Deviation in X and Y axis at shifting 45 start angle.

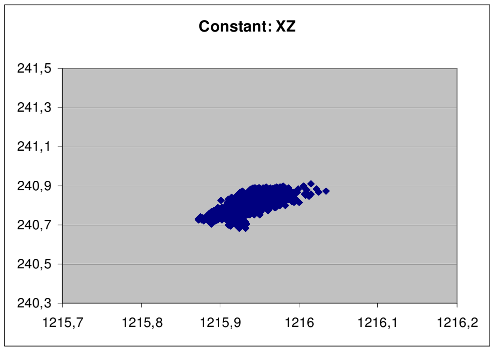


Chart 18.1
Deviation in X and Z axis at constant start angle

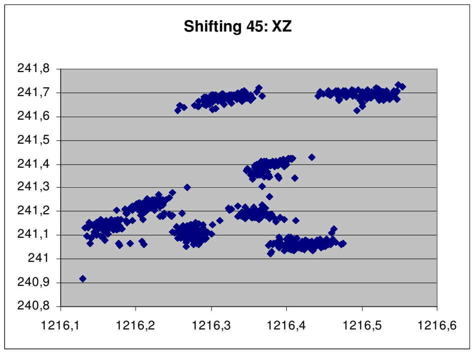


Chart 18.2
Deviation in X and Z axis at 45 degree shifting start angle.

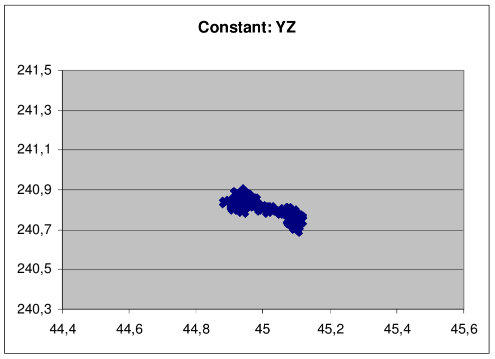


Chart 19.1
Deviation in Y, and Z axis at constant start angle

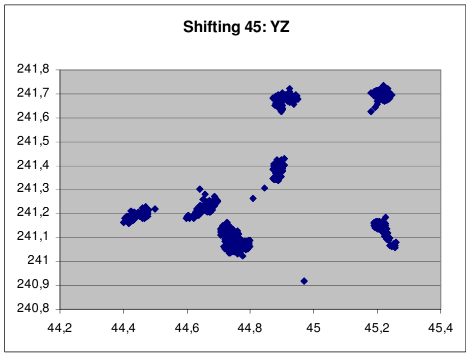


Chart 19.2
Deviation in Y and Z axis at shifting 45 start angle.

5.3.3 Evaluation of measured radius

Radius	**Average**	**1σ ≈ 68%**	**2σ ≈ 95%**	**3σ ≈ 99%**
Constant [mm]	13.572	±0.046	±0.092	±0.139
45º shifting [mm]	13.623	±0.054	±0.107	±0.161

Table 6
In this table a comparison of the radius between the constant and shifting start angle measurements are shown. These results confirm that the measured radius is more or less identical in the constant and shifted measurements.

All measured radius values are smaller than the true radius which is 15mm.This is the sum of the radius of the probe and the fixed sphere, 5mm and respectively 10mm. Most likely the deviation can be explained by the play in the axes gear trains. As shown in the table 6 the measured radius does never get close to the true radius.

During the conducted tests, the orientation of the probe is programmed to be identical relative the sphere at each position. This means that an inadequate defined TCP should not have an influence on the determined radius. On the other hand, an inadequate definition of the TCP will affect the position of the determined centre point.

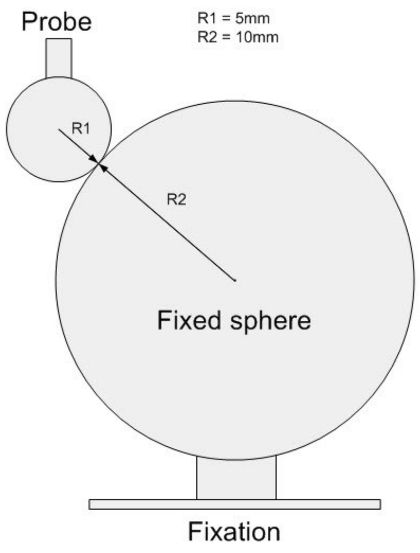


Figure 50
This figure shows the sphere probe and the fixed sphere in contact with each other. The sphere probe has radius 5mm and the fixed sphere has radius of 10mm.

Due to the fact that all measured values of the radius are smaller than 15mm. This indicates that the deviation has been induced by the play in the gear trains. If the deviation was caused by the definition of the TCP or nonlinearities of the axis movements, the measured radius could in fact become greater than 15mm.

Based on the finding that the measured radius is smaller than the actual true radius, the true location of the centre point can not be guarantied. As shown in figure 51, the true centre point may deviate in respect of the measured centre point.

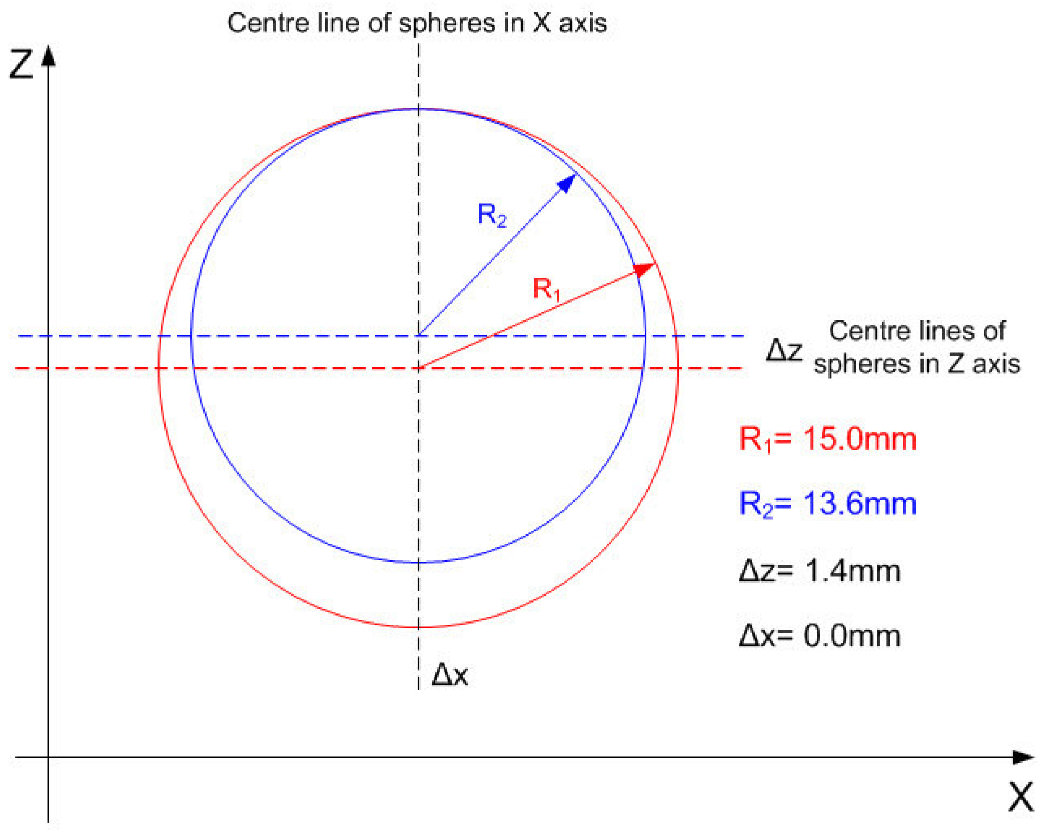

Centre line of spheres in X axis
Z
R2
R1
Δz
Centre lines of spheres in Z axis
R1= 15.0mm
R2= 13.6mm
Δz= 1.4mm
Δx= 0.0mm
Δx
X

Figure 51

In this figure, the cross section of two spheres is shown. The red circle represents the actual true sphere which has a radius of 15mm. The blue circle represents the measured sphere which has an average radius of 13.6mm. Under condition that the centre point of the two circles are perfect aligned in respect of the X-axis, the maximum deviations of the centre point in Z-axis will be the difference of the radius, in this case 1.4mm. This means that the true location can not be ensured. However, based on the deviations of the Euclidian distance to the measured centre point, the method will repeatedly provide this point at certain accuracy. This may not be the true centre point of the fixed sphere but the method will return the same position with very high accuracy. In other, words, the influence by the gear train play is more or less identical at each repeated measurement.

5.3.4 Calibration possibilities

Based on the fact that the method does not provide the true location of the centre point of the fixed sphere in the world coordinates system; the method can not directly be used to transfer an offline generated program to the physical installation under the condition that very high accuracy is required. However due to the fact that the same position repeated at high accuracy; the method can be used in following manner.

1. A number of reference positions are determined on the physical fixture by determine the centre of fixed spheres.
2. One position of the offline generated program is then fine tuned with respect to the determined reference positions of the fixture frame. The rest of the robot program path positions automatically will be modified in equivalent manner. This implies that the work object is not too wide. If too wide, nonlinear axis movement may cause to great path deviations.
3. If the installation has to be recalibrated, the reference positions on the work fixture only have to be determined.

This calibration method can also be used in the traditionally “teach in” manner.

1. A number of reference positions are defined with this method on a work fixture.
2. When these reference positions have been determined, adjustment of positions by teach in programming of the actual work object can be conducted.
3. If the installation has to be recalibrated, the reference positions on the work fixture are determined once again.

5.3.5 Centre of sphere - final evaluation

In the section where constant and shifted start angle are compared shows that the calibration method is less accurate if when the start angle is shifted. In reality, the calibration method will most likely be used with constant start angle. On top of this, a robot installation is most likely built up and programmed. The relative position fluctuations of the installation will then be a matter of millimetres and a couple of degrees. The configuration of the robot axes will therefore be more or less identical. Based on this fact, the values which are shown at the constant start angle will represent the repeated accuracy of this calibration method.

Application	Required accuracy [mm]	Conf int. 84% +/- 0.113 [mm]	Conf int. 98% +/- 0.144 [mm]
Arc welding	+/- 0.4	OK	OK
Spot welding	+/- 0.7	OK	OK
Material handling	+/- 2.0	OK	OK
Water cutting	+/- 0.2	OK	OK
Laser cutting	+/- 0.2	OK	OK
Machining	+/- 1.0	OK	OK
Palletising	+/- 5.0	OK	OK

Table 6
Requirements of accuracy presented as a function of the application.

An industrial robot can be used in many different applications. Depending on the application the requirements of the accuracy will vary. Unfortunately, it does not exist a common standard for the level of accuracy depending of the application. In table 6, estimations of the required accuracy for different applications are presented. These values have to be considered as a general rule of thumb. The values also complies the complete robot process. This means for example that the tolerances of the work piece itself have to be taken in to consideration. As shown, the calibration method based on the repeated accuracy will be suitable for most of the presented applications

6 Discussion

The goal of this thesis is to evaluate a simple calibration method that enables to transfer an offline generated robot program to the physical installation. Traditionally this is done by adjusting the robot path positions by “Teach in”. This is possible due to the fact that the robot has very good position repeatability. However, this method can be time consuming and therefore it is desired to find alternative solutions which replace the position fine tuning step.

The key for this task is to determine reference positions on the work object fixture at high accuracy. If this can be obtained, the offline program can be transferred to the reality by calibrating fixture coordinate system. The evaluated method is based on the fact that when two perfect spheres which are in contact with each other in theory will have identical distance between their centre points. Under the condition that at least four positions around the fixed sphere have been obtained, the centre position of it can be determined. In a robot installation this is obtained by mounting a spherical probe as robot tool and a spherical object on the work piece fixture.

One of the conditions for this method is that the centre position in the spherical probe is defined at high accuracy. In order to define this position, a method which does not add any external equipment was evaluated. In short, this method guides the spherical probe on the robot in to identical position at different orientations. Unfortunately the results from the repeatability measurements do not support this method for calibration. The min/ max deviation level was achieved in Z-axis but not in X-axis and Y-axis. Potentially this method would have been applicable if it was possible to more widely spread out the measurement range. Instead of this method, the bull’s eye method is suggested in order to find the centre of spherical probe on the robot. Unfortunately the Bulls eye method means an additional investment.

In order to evaluate the efficiency of the method which determines the centre of a fixed sphere, repetition tests were conducted. At these tests the centre of the spherical probe was defined with bull’s eye method. The result from these tests confirmed that the method is robust. The measured radius of the fixed sphere somewhat deviated from the actual true radius. This indicates that the true centre point can not be guaranteed. However, the precision of the repeated centre position is very accurate and good enough for many common robot applications.

The method has a limit under condition that high accuracy is required. The repositions of the fixture with respect to the robot can then not be too great. If the fixture position variations are too great, different kind of errors from the robot structure will have an influence on the result. However, this should not be a major issue due to the fact that reposition of fixtures in applications which demands high accuracy in most cases is a matter of millimetres.

Under condition that high accuracy is required; the conclusion based on the repeatability test results is that this calibration method can not be used to directly transfer an offline generated programming in to the physical installation. This is based on the fact that the method does not measure the radius of the sphere at sufficient accuracy.

Even if the calculated centre point of the fixed sphere is not absolute accurate it is still repeatable. This means that the method can be used to find reference positions on a work fixture. By fine tuning one position of the offline generated program in the calibrated fixture frame, the rest of the robot program path positions automatically will be calibrated. However, this implies that the work object is not too wide. If too wide, robot nonlinearities may cause too great path deviations.

The method can also be used in traditional manner based on the high repeatability. By first calibrate a fixture coordinate system and then perform “tech in” programming, the program is generated with respect to the calibrate fixture coordinate system. As the reference positions on the fixture are known, the method can therefore be run in order to calibrate the fixture if required.

References

- Convex optimization, Stephen P. Boyd, Lieven Vandenberghe.
  ISBN 0521833787

- Fitting of Circles and Ellipses, BIT 34 (1994), pp. 558-578. Walter Gander, Gene H. Golub, Rolf Strebel September 26, 1994

- Linear algebra and its application, Third Edition Gilbert Strang.
  ISBN 015551005-3

- Solving linear least squares problem by Gram-Schmidt orthogonalization. Åke Björck BIT 7 (1967)

- Numerical methods for least squares problems, Åke Björck
  ISBN 0898713609

- Accuracy and stability of numerical algorithms, Nicholas J. Higham
  ISBN-10:0898715210

- Kinematische Kalibrierung von Industrierobotern, Ulrich Wiest, Universität Karlsruhe 2001.

7 Appendix

7.1 TCP_Calibration_2.mod

```
MODULE TCP_calibration_2

!--------------- Variables -----------------

! Meas_Sphere "Binzel robot studio"
PERS tooldata Meas_Sphere1:=[TRUE,[[40.0001,3,320],[0.983132,-
0.00145,0.176582,0.0476255]],[1.5,[10,0,110],[1,0,0,0],0.01,0.01,0.01]];

! Bulls eye tool 2011-07-08
! PERS tooldata Meas_Sphere1:=[TRUE,[[43.0001,-
0.4,320.6002],[0.986,0,0.165,0]],[1.5,[10,0,110],[1,0,0,0],0.01,0.01,0.01]];

! Meas_Sphere "Binzel labbrobot"
PERS tooldata Meas_Sphere2:=[TRUE,[[43,-0.17,320.227],[0.983132,-
0.00145,0.176582,0.0476255]],[0.3,[0,0,100],[1,0,0,0],0,0,0]];

! Test tool
PERS tooldata T_tool:=[TRUE,[[49.8707,-
4.06615,315.983],[0.986,0,0.165,0]],[1.5,[10,0,110],[1,0,0,0],0.01,0.01,0.01]]
;

PERS robtarget Pref;

LOCAL VAR speeddata v_slow := [5, 5, 200, 15 ];       ! Set speeddata for
approching the hole
LOCAL VAR speeddata v_norm := [30, 5, 200, 15 ];      ! Set general speeddata

!LOCAL VAR speeddata v_slow := [2000, 500, 200, 15 ]; ! Set speeddata for
approching the hole
!LOCAL VAR speeddata v_norm := [2000, 500, 200, 15 ]; ! Set general speeddata

LOCAL VAR jointtarget joints{100};
LOCAL VAR robtarget Tool0_points{100};
LOCAL VAR robtarget Fake_points{100};
LOCAL VAR pose Cali_TCP{100};

LOCAL VAR string joints_pos_str;

LOCAL VAR iodev Pos_file;
LOCAL VAR string FilePath:="TCP_biglog";


!--------------- End of variables -----------------
```

```
PROC Bort_main()

      VAR num Counter:=1000;
      VAR num repeats;

      repeats:=counter;

      FOR i4 FROM 1 TO counter DO
            IF i4=1
            THEN
                  Pref:=CRobT(\Tool:=Meas_Sphere1);
            ENDIF
            Get_TCP_points 80;

            Calib_TCP Tool0_points, 80;

            !Calib_TCP_NEW joints;
            TEST_New_TCP;

            repeats:=repeats-1;
            TPWrite "Repeats remain= "\Num:=repeats;
      ENDFOR
      Stop;
ENDPROC


PROC Get_TCP_points(num meas)

      VAR robtarget Calib_pos;
      VAR robtarget Tool0_RobT;
      VAR num pos_index:=0;

      VAR string joints_pos_str;

      VAR num max_reor_angle_x:=25;
      VAR num max_reor_angle_y:=25;
      VAR num reor_angle;


      VAR num repos;

      ! Activate soft servo on all six axes
      SoftAct 1, 50\Ramp:=150;
      SoftAct 2, 30\Ramp:=100;
      SoftAct 3, 30\Ramp:=100;
      SoftAct 4, 30\Ramp:=150;
      SoftAct 5, 30\Ramp:=150;
      SoftAct 6, 50\Ramp:=150;

      !TPReadFK repos, "Do you need new reference position?", stEmpty,
stEmpty, stEmpty, "Yes", "No";
```

```
      !IF repos=4
      !     THEN
                  ! Jog to postion above caibration hole.
      !           Stop;
                  !Pref:=CRobT(\Tool:=Meas_Sphere1);
      !     ELSE
                  !MoveL Pref, v2000, fine, Meas_Sphere1;
      !ENDIF

!--- Get TCP points  -----

      joints_pos_str:=CDate()+" "+CTime()+" ";

      FOR i1 FROM 1 TO meas/4 DO

                  reor_angle:=max_reor_angle_x*cos(180*i1/(meas/3));

                  pos_index:=pos_index+1;
                  TPWrite "Measurement "+NumToStr(Pos_index,0)+" of
"+NumToStr(meas,0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=reor_angle,\Ry:=0,\Rz:=0);
                  MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

      Calib_pos:=RelTool(Pref,0,0,10,\Rx:=reor_angle,\Ry:=0,\Rz:=0);
                  MoveL Calib_pos, v_slow, fine, Meas_Sphere1;
                  Waittime(1);
                  Tool0_points{pos_index}:=CRobT(\Tool:=tool0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=reor_angle,\Ry:=0,\Rz:=0);
                  MoveL Calib_pos, v_norm, z1, Meas_Sphere1;

      ENDFOR

      Calib_pos:=RelTool(Pref,0,0,0);
      MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

      FOR i2 FROM meas/4+1 TO meas*2/4 DO

                  reor_angle:=max_reor_angle_y*cos(180*i2/(meas/4));

                  pos_index:=pos_index+1;
                  TPWrite "Measurement "+NumToStr(Pos_index,0)+" of
"+NumToStr(meas,0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=0,\Ry:=reor_angle,\Rz:=0);
                  MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

      Calib_pos:=RelTool(Pref,0,0,10,\Rx:=0,\Ry:=reor_angle,\Rz:=0);
                  MoveL Calib_pos, v_slow, fine, Meas_Sphere1;
                  Waittime(1);
```

```
            Tool0_points{pos_index}:=CRobT(\Tool:=tool0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=0,\Ry:=reor_angle,\Rz:=0);
            MoveL Calib_pos, v_norm, z1, Meas_Sphere1;

      ENDFOR

      Calib_pos:=RelTool(Pref,0,0,0);
      MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

      FOR i3 FROM meas*2/4+1 TO meas*3/4 DO

            reor_angle:=max_reor_angle_y*cos(180*i3/(meas/4));

            pos_index:=pos_index+1;
            TPWrite "Measurement "+NumToStr(Pos_index,0)+" of
"+NumToStr(meas,0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=reor_angle,\Ry:=0,\Rz:=45);
            MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

      Calib_pos:=RelTool(Pref,0,0,10,\Rx:=reor_angle,\Ry:=0,\Rz:=45);
            MoveL Calib_pos, v_slow, fine, Meas_Sphere1;
            Waittime(1);
            Tool0_points{pos_index}:=CRobT(\Tool:=tool0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=reor_angle,\Ry:=0,\Rz:=45);
            MoveL Calib_pos, v_norm, z1, Meas_Sphere1;
      ENDFOR

      Calib_pos:=RelTool(Pref,0,0,0);
      MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

      FOR i4 FROM meas*3/4+1 TO meas DO

            reor_angle:=max_reor_angle_y*cos(180*i4/(meas/4));

            pos_index:=pos_index+1;
            TPWrite "Measurement "+NumToStr(Pos_index,0)+" of
"+NumToStr(meas,0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=0,\Ry:=reor_angle,\Rz:=45);
            MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

      Calib_pos:=RelTool(Pref,0,0,10,\Rx:=0,\Ry:=reor_angle,\Rz:=45);
            MoveL Calib_pos, v_slow, fine, Meas_Sphere1;
            Waittime(1);
            Tool0_points{pos_index}:=CRobT(\Tool:=tool0);

      Calib_pos:=RelTool(Pref,0,0,0,\Rx:=0,\Ry:=reor_angle,\Rz:=45);
            MoveL Calib_pos, v_norm, z1, Meas_Sphere1;
      ENDFOR

      Calib_pos:=RelTool(Pref,0,0,0);
      MoveL Calib_pos, v_norm, fine, Meas_Sphere1;

!--- Return to start position -----

      MoveL Pref,v_norm,Z1,Meas_Sphere1;

      SoftDeact \Ramp:=150;

ENDPROC


PROC Calib_TCP_NEW(VAR jointtarget xJoints{*})

      VAR num Maxerr;
      VAR num Meanerr;

      MToolTCPCalib xJoints{1}, xJoints{3}, xJoints{2}, xJoints{4},
Meas_Sphere2, Maxerr, Meanerr;

      joints_pos_str:=" Meas_Sphere2
"+NumToStr(Meas_Sphere2.tframe.trans.x,3)+"
"+NumToStr(Meas_Sphere2.tframe.trans.y,3)+"
"+NumToStr(Meas_Sphere2.tframe.trans.z,3)+" Maxerr "+NumToStr(Maxerr,3)+"
Meanerr "+NumToStr(Meanerr,3);

      !  Open posistion file
      Open FilePath, Pos_file\Append;
      Write Pos_file, joints_pos_str  \NoNewLine;
      !  Close position file
      Close Pos_file;

      TPWrite "Meas_Sphere2="\Pos:=Meas_Sphere2.tframe.trans;
      TPWrite "Maxerr ="\Num:=Maxerr;
      TPWrite "Meanerr ="\Num:=Meanerr;

ENDPROC

! Calibrates the TCP. Argument one is the estimated expressend in World
coordinates. Argument two is the position of tool0.
PROC Calib_TCP(VAR robtarget Tool0_points{*}, num meas)

      VAR pose E_TCP;
      VAR pose Tool0_pose;
      VAR pose Cali_TCP_final;

      VAR num X_TCP_final:=0;
      VAR num Y_TCP_final:=0;
      VAR num Z_TCP_final:=0;
      VAR num Maxerr;
      VAR num Meanerr;

      VAR num Rerror_ref;
      VAR num Rerror_test;

      VAR string M_2;
      VAR string T_t;

      VAR pose Cali_TCP;

      VAR iodev Pos_file2;
      VAR string FilePath2:="TCP_poslog";
      VAR string TCP_str;

      FilePath2:=FilePath2+"_"+CDate()+".log";

      ! Least square estimation based on Tool0 points, part of a Sphere

      ECOS_QR Tool0_points, meas;
      !Stop;

      ! Find best tool0 point

      ! Rerror_ref:=abs(Radius-sqrt(pow(ECP_QR_RobT.trans.x-
Tool0_points{1}.trans.x,2)+pow(ECP_QR_RobT.trans.y-
Tool0_points{1}.trans.y,2)+pow(ECP_QR_RobT.trans.z-
Tool0_points{1}.trans.z,2)));
      ! Meas_Sphere2.tframe.trans:=[ECP_QR_RobT.trans.x-
Tool0_points{1}.trans.x,ECP_QR_RobT.trans.y-
Tool0_points{1}.trans.y,ECP_QR_RobT.trans.z-Tool0_points{1}.trans.z];
      ! Meas_Sphere2.tframe.rot:=E_TCP.rot;

      E_TCP.trans:=ECP_QR_RobT.trans;           ! Get estimated cenre point
based on QR factorixation
      E_TCP.rot:=Meas_Sphere1.tframe.rot;

      Tool0_pose.trans:=Tool0_points{1}.trans;
      Tool0_pose.rot:=Tool0_points{1}.rot;

      Cali_TCP:= PoseMult(PoseInv(Tool0_pose), E_TCP);

      Rerror_ref:=abs(Radius-
sqrt(pow(Cali_TCP.trans.x,2)+pow(Cali_TCP.trans.y,2)+pow(Cali_TCP.trans.z,2)))
;

      Meas_Sphere2.tframe.rot:=Meas_Sphere1.tframe.rot;
      Meas_Sphere2.tframe.trans:=Cali_TCP.trans;

      TPWrite "Start Meas_sphere2 = "\Pos:=Meas_Sphere2.tframe.trans;

      FOR index FROM 2 TO meas DO
```

```
            Tool0_pose.trans:=Tool0_points{index}.trans;
            Tool0_pose.rot:=Tool0_points{index}.rot;

            Cali_TCP:= PoseMult(PoseInv(Tool0_pose), E_TCP);

            Rerror_test:=abs(Radius-
sqrt(pow(Cali_TCP.trans.x,2)+pow(Cali_TCP.trans.y,2)+pow(Cali_TCP.trans.z,2)))
;

            !TPWrite "Rerror_test= "\Num:=Rerror_test;
            !TPWrite "Rerror_ref= "\Num:=Rerror_ref;


            IF     Rerror_ref > Rerror_test
            THEN

                  Meas_Sphere2.tframe.trans:=Cali_TCP.trans;

                  Rerror_ref := Rerror_test;

                  TPWrite "Radius error = "\Num:=Rerror_test;


            ENDIF
      ENDFOR

      TPWrite "Final Meas_sphere2 = "\Pos:=Meas_Sphere2.tframe.trans;

      Open FilePath2, Pos_file2\Append;

      TCP_str:="Radius "+NumToStr(Radius,6)+"Tool_TCP
"+NumToStr(Meas_Sphere2.tframe.trans.x,6)+"
"+NumToStr(Meas_Sphere2.tframe.trans.y,6)+"
"+NumToStr(Meas_Sphere2.tframe.trans.z,6);

      Write Pos_file2, TCP_str;

      Close Pos_file2;

ENDPROC

PROC TEST_New_TCP()

      VAR robtarget P1;
      VAR robtarget P2;
      VAR robtarget P3;

      TPWrite "Meas_sphere2";
      P1:=RelTool(Pref,0,0,-50);
      MoveJ P1, v_norm, fine, Meas_Sphere2;
      P2:=RelTool(Pref,0,0,-50,\Rx:=0,\Ry:=0,\Rz:=45);
      MoveL P2, v_norm, fine, Meas_Sphere2;

      P3:=RelTool(Pref,0,0,-50,\Rx:=0,\Ry:=30,\Rz:=0);
      MoveL P3, v_norm, fine, Meas_Sphere2;

      MoveL P1, v_norm, fine, Meas_Sphere2;

!     TPWrite "T_tool";
!     P1:=RelTool(Pref,0,0,-50);
!     MoveJ P1, v_norm, fine, T_tool;

!     P2:=RelTool(Pref,0,0,-50,\Rx:=0,\Ry:=0,\Rz:=45);
!     MoveL P2, v_norm, fine, T_tool;

!     P3:=RelTool(Pref,0,0,-50,\Rx:=0,\Ry:=30,\Rz:=0);
!     MoveL P3, v_norm, fine, T_tool;

!     MoveL P1, v_norm, fine, T_tool;

      MoveL Pref, v_norm, fine, Meas_Sphere1;

ENDPROC

ENDMODULE
```

7.2 Get_sphere_points2.mod

```
MODULE Get_sphere_points_2

!Drop down i Z-axis
LOCAL VAR num Cz:=15;

LOCAL VAR speeddata v_AS := [5, 10, 200, 15 ];        ! Set speeddata for
approching the sphere
LOCAL VAR speeddata v_GEN := [30, 10, 200, 15 ];      ! Set general speeddata

!LOCAL VAR speeddata v_AS := [2000, 10, 200, 15 ];          ! Set speeddata
for approching the sphere
!LOCAL VAR speeddata v_GEN := [2000, 10, 200, 15 ];   ! Set general speeddata

VAR robtarget PointXYZ{100};
LOCAL VAR jointtarget joints{100};

VAR robtarget measpos{10};

! Trap routin parameters
LOCAL VAR num Trigglevel:=110;                                    ! What is
the percentage of reduced distance to trigg search stop
```

```
LOCAL VAR num Itimer_frequency:=0.25;                       ! Itimer interupt
frequency
LOCAL VAR num Start_supervise:=4;! When shall the program start to supervise
the speed
LOCAL VAR intnum Speed_supervision;
LOCAL VAR intnum time_int;
LOCAL VAR dionum High:=1;
LOCAL VAR dionum Low:=0;
LOCAL VAR signaldi Signal_DI;
LOCAL VAR signaldo Signal_DO;
LOCAL VAR num Counter;
LOCAL VAR robtarget P1_supervision;
LOCAL VAR robtarget P2_supervision;

!--------------- Functions -----------------

!Calculate theata angle
FUNC num theata_angle(num order,num N)
      RETURN 110*(order)/N;
ENDFUNC

!Calculate phi angle
FUNC num phi_angle(num order)
      RETURN 80*(order-1);
ENDFUNC

!--------------- End of functions -----------------


PROC Fmain()

      VAR robtarget Pref_sphere;

      Pref_sphere:=CRobT(\Tool:=Meas_Sphere2);
      !Numer of measurements, Outer radius, Inner radius, Top_radius, Refpos
      MAS 20, 40, 20, Pref_sphere, 10;
      Stop;
ENDPROC


! Move Around Sphere
PROC MAS(num NOM, num OuterRad, num InnerRad, robtarget P_referens, num
start_phi)

      VAR robtarget P_centre_sphere;
      VAR robtarget P_OuterRad;
      VAR robtarget P_SemiRad;
      VAR robtarget P_InnerRad;
      VAR robtarget P_Start;
      VAR robtarget P_End;

      VAR num pos_index:=0;
      VAR num fk;

      VAR num rad_sphere:=10;

      ! Pos file variables
      VAR iodev Pos_file;
      !VAR string FilePath:="Sphere_positions";

      VAR string joints_str;
      VAR string positions_str;
      ! End of pos file variables



      !IDELETE Speed_supervision;
      !CONNECT Speed_supervision WITH Flag_speed_change;
      !ISleep Speed_supervision;
      AliasIO IO_Signal_DI, Signal_DI;
      AliasIO IO_Signal_DO, Signal_DO;

      ! Activate soft servo
      SoftAct 1, 40 \Ramp:=150;
      SoftAct 2, 40 \Ramp:=150;
      SoftAct 3, 40 \Ramp:=150;
      SoftAct 4, 40 \Ramp:=150;
      SoftAct 5, 40 \Ramp:=150;
      SoftAct 6, 40 \Ramp:=150;

      joints_str:=CDate()+" "+CTime()+" ";
      Open FilePath, Pos_file\Append;
      Write Pos_file, joints_str \NoNewLine;
      Close Pos_file;


      ! Find position on top of sphere

      P_OuterRad:=RelTool(P_referens,0,0,0);
      P_InnerRad:=RelTool(P_referens,0,0,5);

      MoveJ P_OuterRad, v_GEN, fine, Meas_Sphere2;
      !MoveL P_InnerRad,v_AS,fine,Meas_Sphere2;

      Counter:=0;
      SetDO Signal_DO, High;
!     WaitTime 2;
      SetDO Signal_DO, Low;
!     WaitTime 2;
      ! Connect trap routine
!     CONNECT Speed_supervision WITH Flag_speed_change;
!     ITimer Itimer_frequency, Speed_supervision;
!     SearchL \SStop, Signal_DI\Flanks, PointXYZ{pos_index}, P_InnerRad, v_AS,
Meas_Sphere2;
!     MoveL P_InnerRad,v_AS,fine,Meas_Sphere2;
!     IDELETE Speed_supervision;

      WaitTime(0.1);

      ! Get ref position on top of fixed sphere
      !P_centre_sphere:=CRobT(\Tool:=Meas_Sphere2);

      !Shift ref pos close to centre of sphere
      !P_centre_sphere:=RelTool(P_centre_sphere, 0, 0, rad_sphere+5);
      P_centre_sphere:=RelTool(P_referens, 0, 0, rad_sphere+5);

      !MoveJ P_OuterRad,v_GEN,fine,Meas_Sphere2;



      ! Get positions around the sphere
      FOR i1 FROM 1 TO NOM DO

            pos_index:=pos_index+1;

!           TPWrite "Theata " \Num:=theata_angle(i1, NOM);
!           TPWrite "10*cos(Theata) " \Num:=10*cos(theata_angle(i1, NOM));
!           TPWrite "Phi " \Num:=phi_angle(i1);

            IF i1=1
            THEN
                  P_Start:=RelTool(P_referens, OuterRad*sin(theata_angle(i1,
NOM))*sin(start_phi+phi_angle(i1)), OuterRad*sin(theata_angle(i1,
NOM))*cos(start_phi+phi_angle(i1)), 0);
                  MoveL P_Start,v_GEN,fine,Meas_Sphere2;
            ENDIF

!           P_OuterRad:=RelTool(P_centre_sphere, OuterRad*sin(theata_angle(i1,
NOM))*sin(phi_angle(i1)), OuterRad*sin(theata_angle(i1,
NOM))*cos(phi_angle(i1)), -OuterRad*cos(theata_angle(i1, NOM))
\Rx:=10*sin(phi_angle(i1)) \Ry:=10*cos(phi_angle(i1)));
!           P_SemiRad:=RelTool(P_centre_sphere,
0.3*OuterRad*sin(theata_angle(i1, NOM))*sin(phi_angle(i1)),
0.3*OuterRad*sin(theata_angle(i1, NOM))*cos(phi_angle(i1)), -
0.3*OuterRad*cos(theata_angle(i1, NOM)) \Rx:=10*sin(phi_angle(i1))
\Ry:=10*cos(phi_angle(i1)));
!           P_InnerRad:=RelTool(P_centre_sphere, InnerRad*sin(theata_angle(i1,
NOM))*sin(phi_angle(i1)), InnerRad*sin(theata_angle(i1,
NOM))*cos(phi_angle(i1)), -InnerRad*cos(theata_angle(i1, NOM))
\Rx:=10*sin(phi_angle(i1)) \Ry:=10*cos(phi_angle(i1)));

            P_OuterRad:=RelTool(P_centre_sphere, OuterRad*sin(theata_angle(i1,
NOM))*sin(start_phi+phi_angle(i1)), OuterRad*sin(theata_angle(i1,
NOM))*cos(start_phi+phi_angle(i1)), -OuterRad*cos(theata_angle(i1, NOM)));
```

```
            P_SemiRad:=RelTool(P_centre_sphere,
0.3*OuterRad*sin(theata_angle(i1, NOM))*sin(start_phi+phi_angle(i1)),
0.3*OuterRad*sin(theata_angle(i1, NOM))*cos(start_phi+phi_angle(i1)), -
0.3*OuterRad*cos(theata_angle(i1, NOM)));
            P_InnerRad:=RelTool(P_centre_sphere, InnerRad*sin(theata_angle(i1,
NOM))*sin(start_phi+phi_angle(i1)), InnerRad*sin(theata_angle(i1,
NOM))*cos(start_phi+phi_angle(i1)), -InnerRad*cos(theata_angle(i1, NOM)));

            MoveL P_OuterRad,v_GEN,fine,Meas_Sphere2;


            Counter:=0;

            ! Connect trap routine
            SetDO Signal_DO, Low;
!           MoveL P_SemiRad,v_AS,fine,Meas_Sphere2;
!           CONNECT Speed_supervision WITH Flag_speed_change;
!           ITimer Itimer_frequency, Speed_supervision;
!           SearchL \SStop, Signal_DI, PointXYZ{pos_index}, P_InnerRad, v_AS,
Meas_Sphere2;
            MoveL P_InnerRad,v_AS,fine,Meas_Sphere2;
!           IDELETE Speed_supervision;

            WaitTime(1);

            joints{pos_index} := CJointT();
            PointXYZ{pos_index}:=CRobT(\Tool:=Meas_Sphere2);

            !  Open posistion file
            Open FilePath, Pos_file\Append;
            joints_str:=" Joint"+NumToStr(pos_index,0)+"
"+NumToStr(joints{pos_index}.robax.rax_1,5)+"
"+NumToStr(joints{pos_index}.robax.rax_2,5)+"
"+NumToStr(joints{pos_index}.robax.rax_3,5)+"
"+NumToStr(joints{pos_index}.robax.rax_4,5)+"
"+NumToStr(joints{pos_index}.robax.rax_5,5)+"
"+NumToStr(joints{pos_index}.robax.rax_6,5);
            Write Pos_file, joints_str \NoNewLine;
            positions_str:=" Pos"+NumToStr(pos_index,0)+"
"+NumToStr(PointXYZ{pos_index}.trans.x,6)+"
"+NumToStr(PointXYZ{pos_index}.trans.y,6)+"
"+NumToStr(PointXYZ{pos_index}.trans.z,6);
            Write Pos_file, positions_str \NoNewLine;
            !  Close position file
            Close Pos_file;

            MoveJ P_OuterRad,v_GEN,fine,Meas_Sphere2;

            IF i1=NOM
            THEN
```

```
                  P_End:=RelTool(P_referens, OuterRad*sin(theata_angle(i1,
NOM))*sin(start_phi+phi_angle(i1)), OuterRad*sin(theata_angle(i1,
NOM))*cos(start_phi+phi_angle(i1)), -5);
                  MoveL P_End,v_GEN,fine,Meas_Sphere2;
            ENDIF

      ENDFOR

      MoveL RelTool(P_referens,0,0,0),v_GEN,fine,Meas_Sphere2;

      SoftDeact;

      !  Open posistion file
!     Open FilePath, Pos_file\Append;
!     Write Pos_file, " ";
      !  Close position file
!     Close Pos_file;

      ERROR
      IF ERRNO=ERR_WHLSEARCH
      THEN
      !StorePath;
      MoveJ P_OuterRad,v_GEN,fine,Meas_Sphere2;
      !RestoPath;
      RETRY;
      ELSEIF ERRNO=ERR_SIGSUPSEARCH
            THEN
            TPWrite "The signal of the SearchL instruction is already high!";
            TPReadFK fk,"Try again after manual reset of
signal?","YES","stEmpty","stEmpty","stEmpty","NO";
            IF fk = 1
                  THEN
                  MoveJ P_OuterRad,v_GEN,fine,Meas_Sphere2;
                  RETRY;
                  ELSE
                  Stop;
            ENDIF
      ENDIF

ENDPROC


LOCAL TRAP Flag_speed_change

      VAR num Dist_diff;

      Dist_diff:=v_AS.v_tcp*Itimer_frequency*Trigglevel/100;      !Set
distance diff
      !Dist_diff:=Trigglevel;

      P1_supervision:=CRobT(\Tool:=Meas_Sphere2);
      IF Counter > Start_supervise
            THEN
            IF Distance(P2_supervision.trans, P1_supervision.trans) <
Dist_diff
                  THEN
                        !TPReadFK repos, "Set DO 1", stEmpty, stEmpty,
stEmpty, "Yes", "No";
                        TPWrite "-----------------";
                        TPWrite "Contact!!!!!!!!!!";
                        SetDO Signal_DO, High;
            ENDIF
      ENDIF

      TPWrite "Distance " \Num:=Distance(P2_supervision.trans,
P1_supervision.trans);
      TPWrite "Dist diff " \Num:=Dist_diff;
      TPWrite "Counter " \Num:=Counter;
!     TPWrite "P1 " \Pos:=P1_supervision.trans;
!     TPWrite "P2 " \Pos:=P2_supervision.trans;
      P2_supervision:=P1_supervision;
      Counter:=Counter+1;

      RETURN;
ENDTRAP
ENDMODULE
```

7.3 Linear_Least_squares_QR.mod

```
MODULE Linear_Least_squares_QR

!--------------- Variables -----------------


! Declares dimension of matrices
LOCAL VAR num m;
LOCAL VAR num n;
LOCAL VAR num p;

LOCAL VAR robtarget PointsXYZ{5};

! Declaration of matrices

!LOCAL VAR num A_matrix{5,4}:=[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1],[1,1,-
1,1]];
!LOCAL VAR num B_matrix{5,1}:=[[2],[1],[3],[1],[1]];

LOCAL VAR num
V_matrix{100,4}:=[[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],
[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0
,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0
,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0
,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],
[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0
,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0
,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0
,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],
[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0
,0]];
LOCAL VAR num
Q_matrix{100,4}:=[[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],
[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0
,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0
,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0
,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],
[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0
,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0
,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0
,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],
[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0
,0]];
```

```
LOCAL VAR num R_matrix{4,4}:=[[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]];
LOCAL VAR num QTB_matrix{4,1};

! Linear least square solution
VAR num LLS_X{10};

!--------------- Procedures --------------------------

!PROC main()

!LLS_QR A_matrix, B_matrix;
!STOP;

!ENDPROC

! Main procedure for linear least square QR based on modified Gram Schmidt. xm
defines amount of rows. xn defines amount of columns.
PROC LLS_QR(VAR num xA_matrix{*,*}, VAR num xB_matrix{*,*}, num xm, num xn)

      !Set dimensions for m
      !m:=DIM(xA_matrix,1);
      m:=xm;

      !Set dimensions for n
      !n:=DIM(xA_matrix,2);
      n:=xn;

      !Set dimensions for p
      !p:=DIM(xB_matrix,2);
      p:=1;

      QR_MOD_GRAM_SCHMIDT xA_matrix, m, n;
      Build_QTB_matrix Q_matrix, xB_matrix;
      Back_solve R_matrix, QTB_matrix;

      ECP_QR_RobT:=PointsXYZ{1};
      ECP_QR_RobT.trans:=ECP_QR_pos;

ENDPROC


! FIND Q AND R MATRICES BY MODIFIED GRAM-SCHMIDT (STABLE)
LOCAL PROC QR_MOD_GRAM_SCHMIDT(num xA_matrix{*,*},num m, num n)

      VAR num temp;

      FOR index1 FROM 1 TO m DO
            FOR index2 FROM 1 TO n DO
                  V_matrix{index1,index2}:=xA_matrix{index1,index2};
            ENDFOR
      ENDFOR
      FOR index3 FROM 1 TO n DO

            temp:=0;
            FOR index4 FROM 1 TO m DO

                  temp:=temp+pow(V_matrix{index4,index3},2);

            ENDFOR

            R_matrix{index3,index3}:=sqrt(temp);

            FOR index5 FROM 1 TO m DO


      Q_matrix{index5,index3}:=V_matrix{index5,index3}/R_matrix{index3,index3}
;

            ENDFOR

            IF index3<n THEN

                  FOR index6 FROM index3+1 TO n DO

                        R_matrix{index3,index6}:=0;
                        FOR index7 FROM 1 TO m DO
                              !TPWrite "index3 " \Num:=index3;
                              !TPWrite "index6 " \Num:=index6;
                              !TPWrite "index7 " \Num:=index7;

                              !TPWrite "V_matrix "
+NumToStr(index7,0)+","+NumToStr(index6,0)+" = "
\Num:=V_matrix{index7,index6};


      R_matrix{index3,index6}:=R_matrix{index3,index6}+(Q_matrix{index7,index3
}*V_matrix{index7,index6});
                        ENDFOR
                        !TPWrite "R_matrix "
+NumToStr(index3,0)+","+NumToStr(index6,0)+" = " \Num:=temp;
                        !Stop;

                        FOR index8 FROM 1 TO m DO
                              temp:=0;

                              !TPWrite "R_matrix"
+NumToStr(index3,0)+","+NumToStr(index6,0)+" = "
\Num:=R_matrix{index3,index6};
                              !TPWrite "Q_matrix"
+NumToStr(index8,0)+","+NumToStr(index3,0)+" = "
\Num:=Q_matrix{index8,index3};

      temp:=R_matrix{index3,index6}*Q_matrix{index8,index3};
```

```
                              !TPWrite "V_matrix före "
+NumToStr(index8,0)+","+NumToStr(index6,0)+" = "
\Num:=V_matrix{index8,index6};
                              V_matrix{index8,index6}:=
V_matrix{index8,index6}-temp;
                              !TPWrite "Minus "
+NumToStr(index8,0)+","+NumToStr(index6,0)+" = " \Num:=temp;
                              !TPWrite "V_matrix efter "
+NumToStr(index8,0)+","+NumToStr(index6,0)+" = "
\Num:=V_matrix{index8,index6};
                              !Stop;

                        ENDFOR
                  ENDFOR

            ENDIF
      ENDFOR


      FOR index10 FROM 1 TO n DO
            FOR index11 FROM 1 TO n DO
                  !TPWrite "R_matrix "
+NumToStr(index10,0)+","+NumToStr(index11,0)+" = "
\Num:=R_matrix{index10,index11};
            ENDFOR
      ENDFOR
      !Stop;
      FOR index13 FROM 1 TO n DO
            FOR index12 FROM 1 TO m DO
                  !TPWrite "Q_matrix "
+NumToStr(index12,0)+","+NumToStr(index13,0)+" = "
\Num:=Q_matrix{index12,index13};
            ENDFOR
      ENDFOR
      !Stop;
ENDPROC


LOCAL PROC Build_QTB_matrix(VAR num xQ_matrix{*,*}, VAR num xB_matrix{*,*})

      VAR num temp;
      VAR num QT{100,100};

      !Transponate Q matrix
      FOR i1 FROM 1 TO m DO
            FOR i2 FROM 1 TO n DO
                  QT{i2,i1}:=xQ_matrix{i1,i2};
            ENDFOR
      ENDFOR

      !TPWrite "A_matrix m dim= " \Num:=m;
      !TPWrite "A_matrix n dim= " \Num:=n;
      !Stop;

      FOR index1 FROM 1 TO p DO
            FOR index2 FROM 1 TO n DO
                  temp:=0;
                  FOR index3 FROM 1 TO m DO

      temp:=temp+QT{index2,index3}*xB_matrix{index3,index1};
                  ENDFOR
                  QTB_matrix{index2,index1}:=temp;
            ENDFOR
      ENDFOR

ENDPROC


! Back solve
LOCAL PROC Back_solve(VAR num xUTR_matrix{*,*}, VAR num xB_vector{*,*})

      VAR string FilePath:="D:/Simulation";
      VAR iodev Pos_file;
      VAR string ECP_X;
      VAR string ECP_Y;
      VAR string ECP_Z;


      !VAR num m:=0;
      !VAR num n:=0;

      VAR num sum;
      VAR num INVi2;

      VAR num x{10};

      !m:=DIM(xUTR_matrix,1);
      !n:=DIM(xUTR_matrix,2);

      !TPWrite "m " \Num:=m;
      !TPWrite "n " \Num:=n;
      !Stop;

      FOR k FROM 1 TO n-1 DO
            !TPWrite "k " \Num:=k;
            FOR i1 FROM k+1 TO n DO
            !     TPWrite "i " \Num:=i1;
                  xb_vector{i1,1}:=xb_vector{i1,1}-
xUTR_matrix{i1,k}*xb_vector{i1,1};
            ENDFOR
      ENDFOR

      x{n}:=xb_vector{n,1}/xUTR_matrix{n,n};
      !TPWrite "x " \Num:=x{n};
      !Stop;

      FOR i2 FROM 1 TO n-1 DO

            INVi2:=n-i2;
      !     TPWrite "INVi2 " \Num:=INVi2;

            sum:=xb_vector{INVi2,1};

            FOR j FROM INVi2 TO n DO

                  sum:=sum-xUTR_matrix{INVi2,j}*x{j};

            ENDFOR

            x{INVi2}:=sum/xUTR_matrix{INVi2,INVi2};

      ENDFOR

      FOR index FROM 1 TO n DO
            LLS_X{index}:=x{index};
            !TPWrite "LLS_x:  " \Num:=LLS_X{index};
      ENDFOR



!     Open FilePath, Pos_file\Append;

!     Write Pos_file, ECP_X \NoNewLine;
!     Write Pos_file, ECP_Y \NoNewLine;
!     Write Pos_file, ECP_Z;
!     Close Pos_file;

      !Stop;
ENDPROC


!--------------- End of procedures -----------------
ENDMODULE
```

7.4 Estimate_centre_of_sphere_QR_2.mod

```
MODULE Estimate_centre_of_sphere_QR_2


!--------------- Variables -----------------

! Estimated centre point

VAR pos ECP_QR_pos;
VAR robtarget ECP_QR_RobT;
VAR num Radius;
VAR num converg{4};

! Declares dimension of matrices
LOCAL VAR num m;
LOCAL VAR num n;
LOCAL VAR num p;

LOCAL VAR robtarget tPointsXYZ{5};

! Declaration of matrices
LOCAL VAR num A_matrix{100,3};
!LOCAL VAR num A_matrix{4,3}:=[[9,0,26,3],[12,0,-7,],[0,4,4],[0,-3,-3]];
!LOCAL VAR num B_matrix{4,1}:=[[1],[2],[3],[5]];
LOCAL VAR num B_matrix{100,1};

LOCAL VAR num J_matrix{100,4};
LOCAL VAR num D_matrix{4,1}:=[[1],[2],[3],[5]];

LOCAL VAR num R0;
LOCAL VAR num Di_vector{100,1};

PERS tooldata
test_tool:=[TRUE,[[0,0,100],[1,0,0,0]],[0.3,[0,0,100],[1,0,0,0],0,0,0]];



!--------------- End of variables -----------------

!--------------- Functions -----------------

! Excecute calculation Xn(n+1)^2-Xn^2+Y(n+1)^2-Yn^2+Z(n+1)^2-Zn^2
LOCAL FUNC num B_row(Pos P1, Pos P2)
      RETURN Pow(P2.x,2)-Pow(P1.x,2)+Pow(P2.y,2)-Pow(P1.y,2)+Pow(P2.z,2)-
Pow(P1.z,2);
ENDFUNC

! Excecute calculation 2x(X(n+1)-Xn)
LOCAL FUNC num A_row(num P1, num P2)
```

```
      RETURN 2*(P2-P1);
ENDFUNC

! Calculate determinant of 2x2 matrix A.
LOCAL FUNC num DET2x2(num A11, num A22, num A21, num A12)
      RETURN A11*A22-(A12*A21);
ENDFUNC


! Excecute calculation
LOCAL FUNC num Calc_Ri(num Xi, num X0, num Yi, num Y0, num Zi, num Z0 )
      RETURN sqrt(pow(Xi-X0,2)+pow(Yi-Y0,2)+pow(Zi-Z0,2));
ENDFUNC

!--------------- End of functions -----------------

PROC gMain()

      VAR num Xi{4};

      !VAR num xA_matrix{5,4}:=[[1,0,0,0],[1.2,-0.9,0,0],[0,1,1,0],[0,1,-
1,1],[0,0,2,1]];
      !VAR num xB_matrix{5,1}:=[[1],[0],[2],[1.1],[3]];

      Xi{1}:=-1;
      Xi{2}:=1;
      Xi{3}:=5;
      Xi{4}:=1;


      tPointsXYZ{1}.trans:=[0,0,6];

      tPointsXYZ{2}.trans:=[0,0,2];

      tPointsXYZ{3}.trans:=[-0.1,2,4];

      tPointsXYZ{4}.trans:=[0.1,-2,4];

      tPointsXYZ{5}.trans:=[2.1,0,4];

      !ECOS_QR tPointsXYZ;

      !LLS_QR xA_matrix, xB_matrix;

      !Gauss_Newton tPointsXYZ, Xi, 3, 10;

      Stop;
ENDPROC

!--------------- Procedures -----------------------

! Main procedure for Estimate centre of sphere
```

```
PROC ECOS_QR(VAR robtarget PointsXYZ{*}, num meas)

      Build_A_matrix PointsXYZ, meas;
      Build_B_matrix PointsXYZ, meas;
!     TPWrite "DIM(PointsXYZ,1):  " \Num:=DIM(PointsXYZ,1)-1;
!     TPWrite "DIM(A_matrix,2):  " \Num:=DIM(A_matrix,2);
!     Stop;

      !Least square QR Ax=b , Rows , Columns
      LLS_QR A_matrix, B_matrix, meas-1, DIM(A_matrix,2);

!     TPWrite "LLS_x:  " \Num:=LLS_X{1};
!     TPWrite "LLS_y:  " \Num:=LLS_X{2};
!     TPWrite "LLS_z:  " \Num:=LLS_X{3};
!     TPWrite "LLS_r:  " \Num:=LLS_X{4};
      LLS_X{4}:=25;
      !Stop;

      !Gauss Newton
      Gauss_Newton PointsXYZ, LLS_x, 3, meas;

ENDPROC


! Builds B Matrix, Set number of decimals by the varibel digits
LOCAL PROC Build_B_matrix(VAR robtarget xPointsXYZ{*}, VAR num xMeas)
      FOR index FROM 1 TO xMeas-1 DO
            B_matrix{index,1}:=B_row(xPointsXYZ{index}.trans,
xPointsXYZ{index+1}.trans);
      ENDFOR

      !FOR i1 FROM 1 TO m DO
      !     FOR i2 FROM 1 TO 1 DO
                  !TPWrite "B_matrix " +NumToStr(i1,0)+","+NumToStr(i2,0)+" =
" \Num:=B_matrix{i1,i2};
      !     ENDFOR
      !ENDFOR
      !Stop;

ENDPROC


! Builds A matrix, Set number of decimals by the varibel digits
LOCAL PROC Build_A_matrix(VAR robtarget xPointsXYZ{*}, VAR num xMeas)
      FOR index FROM 1 TO xMeas-1 DO
            A_matrix{index,1}:=A_row(xPointsXYZ{index}.trans.x,
xPointsXYZ{index+1}.trans.x);
            A_matrix{index,2}:=A_row(xPointsXYZ{index}.trans.y,
xPointsXYZ{index+1}.trans.y);
            A_matrix{index,3}:=A_row(xPointsXYZ{index}.trans.z,
xPointsXYZ{index+1}.trans.z);
      ENDFOR
      !FOR i1 FROM 1 TO m DO
      !     FOR i2 FROM 1 TO n DO
            !     TPWrite "A_matrix " +NumToStr(i1,0)+","+NumToStr(i2,0)+" = "
\Num:=A_matrix{i1,i2};
      !     ENDFOR
      !ENDFOR
      !Stop;

ENDPROC




!Build Jacobian matrix
LOCAL PROC Build_J_Di(VAR robtarget xPointsXYZ{*},VAR num X0{*}, var num
xMeas)

      VAR num Xi;
      VAR num Yi;
      VAR num Zi;

      FOR index FROM 1 TO xMeas DO

            Xi:=xPointsXYZ{index}.trans.x;
            Yi:=xPointsXYZ{index}.trans.y;
            Zi:=xPointsXYZ{index}.trans.z;

            ! Create Jacobian matrix
            J_matrix{index,1}:=-(Xi-X0{1})/Calc_Ri( Xi, X0{1}, Yi, X0{2}, Zi,
X0{3});
            ! TPWrite "J_matrix{index,1} = " \Num:=J_matrix{index,1};
            J_matrix{index,2}:=-(Yi-X0{2})/Calc_Ri( Xi, X0{1}, Yi, X0{2}, Zi,
X0{3});
            ! TPWrite "J_matrix{index,2} = " \Num:=J_matrix{index,2};
            J_matrix{index,3}:=-(Zi-X0{3})/Calc_Ri( Xi, X0{1}, Yi, X0{2}, Zi,
X0{3});
            ! TPWrite "J_matrix{index,3} = " \Num:=J_matrix{index,3};
            J_matrix{index,4}:=-1;
            ! TPWrite "J_matrix{index,4} = " \Num:=J_matrix{index,4};
            !Stop;
            ! Create -Di matrix
            Di_vector{index,1}:=-((Calc_Ri( Xi, X0{1}, Yi, X0{2},Zi, X0{3})-
X0{4}));

      ENDFOR

ENDPROC


! Gauss Newton
```

```
PROC Gauss_Newton(VAR robtarget xPointsXYZ{*}, VAR num xLLS_x{*}, num
iterations, num xMeas)


            VAR num X_old{4};
            VAR num X_new{4};

            ! Declare Ro startvalue
            !R0:=sqrt((pow(xLLS_x{1},2)+pow(xLLS_x{2},2)+pow(xLLS_x{3},2)-
xLLS_x{4}));
            !TPWrite "R0:  " \Num:=R0;
            R0:=25;

            ! Declare start values
            X_new{1}:=xLLS_x{1};
            TPWrite "Start value X:  " \Num:=X_new{1};
            X_new{2}:=xLLS_x{2};
            TPWrite "Start value Y:  " \Num:=X_new{2};
            X_new{3}:=xLLS_x{3};
            TPWrite "Start value Z:  " \Num:=X_new{3};

      X_new{4}:=Distance(xPointsXYZ{1}.trans,[xLLS_x{1},xLLS_x{2},xLLS_x{3}]);
            TPWrite "Start value R:  "
\Num:=Distance(xPointsXYZ{1}.trans,[xLLS_x{1},xLLS_x{2},xLLS_x{3}]);


            FOR index FROM 1 TO iterations DO

                  ! Create Jacobian matrices J, Di
                  Build_J_Di xPointsXYZ, X_new, xMeas;

                  ! Conduct linear least square
                  LLS_QR J_matrix, Di_vector, xMeas, 4;
                  ! Update with new value guess

                  X_old{1}:=X_new{1};
                  X_old{2}:=X_new{2};
                  X_old{3}:=X_new{3};
                  X_old{4}:=X_new{4};

                  X_new{1}:=X_new{1}+LLS_X{1};
                  X_new{2}:=X_new{2}+LLS_X{2};
                  X_new{3}:=X_new{3}+LLS_X{3};
                  X_new{4}:=X_new{4}+LLS_X{4};

                  !TPWrite "Grad X = " \Num:=LLS_X{1};
                  !TPWrite "Grad Y = " \Num:=LLS_X{2};
                  !TPWrite "Grad Z = " \Num:=LLS_X{3};
                  !TPWrite "Grad rad = " \Num:=LLS_X{4};
                  !Stop;
                  !TPWrite "ABS(X_new{1})-ABS(X_old{1}) = "
\Num:=(ABS(X_new{1})-ABS(X_old{1}));
                  !TPWrite "ABS(X_new{2})-ABS(X_old{2}) = "
\Num:=(ABS(X_new{2})-ABS(X_old{2}));
                  !TPWrite "ABS(X_new{3})-ABS(X_old{3}) = "
\Num:=(ABS(X_new{3})-ABS(X_old{3}));
                  !TPWrite "ABS(X_new{4})-ABS(X_old{4}) = "
\Num:=(ABS(X_new{4})-ABS(X_old{4}));

            ENDFOR

            TPWrite "Convergence X = " \Num:=LLS_X{1};
            IF ABS(LLS_X{1}) < 0.001
                  THEN
                        TPWrite "Covergence X OK";
                  ELSE
                        TPWrite "Covergence error X";
                        Stop;
            ENDIF

            TPWrite " Convergence Y = " \Num:=LLS_X{2};
            IF ABS(LLS_X{2}) < 0.001
                  THEN
                        TPWrite "Covergence Y OK";
                  ELSE
                        TPWrite "Covergence error Y";
                        Stop;
            ENDIF

            TPWrite " Convergence Z = " \Num:=LLS_X{3};
            IF ABS(LLS_X{3}) < 0.001
                  THEN
                        TPWrite "Covergence Z OK";
                  ELSE
                        TPWrite "Covergence error Z";
                        Stop;
            ENDIF

            TPWrite " Convergence Radius = " \Num:=LLS_X{4};
            IF ABS(LLS_X{4}) < 0.001
                  THEN
                        TPWrite "Covergence Radius OK";
                  ELSE
                        TPWrite "Covergence error radius";
                        Stop;
            ENDIF

            ECP_QR_pos.x:=X_new{1};
            ECP_QR_pos.y:=X_new{2};
            ECP_QR_pos.z:=X_new{3};

            ECP_QR_RobT.trans.x:=ECP_QR_pos.x;
            ECP_QR_RobT.trans.y:=ECP_QR_pos.y;
            ECP_QR_RobT.trans.z:=ECP_QR_pos.z;

            TPWrite "Final X = " \Num:=X_new{1};
            TPWrite "Final Y = " \Num:=X_new{2};
            TPWrite "Final Z = " \Num:=X_new{3};
            TPWrite "Final radius = " \Num:=X_new{4};

            Radius:=X_new{4};
            converg{1}:=LLS_X{1};
            converg{1}:=LLS_X{1};
            converg{1}:=LLS_X{1};
            converg{1}:=LLS_X{1};

ENDPROC

!--------------- End of procedures -----------------
ENDMODULE
```