Robot Tool Calibration of an Active Pen with Python using an Enabled Surface from Anoto Technology

Johnny Sjöberg

Department of Automatic Control

MSc Thesis
TFRT-6051
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

This master thesis has shown multiple uses of the Anoto dot pattern and Anoto pen for different applications. One application is the calibration of the robot tool and work object. Another is the use of an Anoto pen and paper as replacement of the ABB Flexpendant to create a lightweight and intuitive control device to control an industrial robot, experimentally demonstrated for the IRB140 robot arm. The Python programming language and interpreter were used together with ABB's RAPID programming language to create a framework to handle the communication between the user and the robot.

The inverse and forward kinematics were derived for the robot and used for simulations and for determining if a trajectory could be traversed. This helped in the theoretical verification in the development of the calibration algorithm. The calibration details from the results section show that a naive approach to calibrating the tool tip, i.e., using all of the measurement values results in a deviation of 1mm or more with slow convergence compared to the manual calibration of 0.5mm deviation. By using a greedy optimization strategy of successively adding measurements that improve the calibration, and removing measurements that worsen it, we get a calibration comparable to a manual calibration with fast convergence.

The work object calibration, i.e., the orientation estimation of the Anoto pattern surface shows promising results with few measurements and faster convergence, but can also be improved with the same optimization strategy. By performing manual measurements, and using measurements of the error in the robot flange position, a realistic lower bound on the precision of the calibration algorithm was decided to be no lower than 0.3 - 0.4mm, and depended on the performance of the pen. The flexpendant control board replacement was developed through a user study with volunteers from Anoto.

Acknowledgements

I would like to thank both my supervisors Anders Robertsson and Per Lidström for being the most supportive supervisors. They have both in different ways inspired me and supported me through my work and have both furthered my interest in a PhD degree from stimulating discussions. Their patience, support and encouraging words have been invaluable in finishing this master thesis.

I would also like to thank Claus Führer for introducing me to the concept of the hand-eye calibration problem.

Anoto has been generous in giving me access to their robotics lab in Lund and letting me work with their technology and I want to thank them for the opportunity to do my master thesis for them.

A special thank you to Terez Lundman and William Sjöberg for their support, patience and for filling my life with laughter.

Contents

1

Introduction

1.1 Problem Description

This master thesis has aimed to solve three specific problems that the Anoto company has been having when developing their digital pens.

During development of new surfaces or new pens the company needs to measure the pixel-, image- and light quality of different combinations of pen models, surface types and pattern printing methods. These quantities depend on the relative orientation between the pen and the pattern surface being measured, geometry of the pen model, quality of print and type of surface. For this reason a 6 degree-of-freedom (DOF) industrial robot arm from ABB is used for measuring these quantities with their digital pens.

Problem I. Before measurements are done with the robot, the robot needs to be calibrated for the tool center point (TCP)[1]. This process can take, in worst case, up to 2 hours to get a good calibration, and 15-30 minutes on average for someone experienced. This is without counting the time it takes to calibrate the position and orientation of the surface the samples are placed on, refered to as the work object.

Since the company usually measures multiple pens and multiple surface samples during development, this quickly becomes a very tedious process that puts a lot of strain (and stress) on the person doing the calibration, and a big time sink for the company in the long run.

Problem II. After tool calibration, the robot measures by collecting data from the pen while moving the pen tip in a straight line[2] on a surface sample multiple times with different constant relative orientations[3]. During this motion the robot can

[1] The pen used for the measurement is fastened in a cradle and mounted on the robot flange.
[2] This is refered to as a "stroke".
[3] This is done with a constant velocity.

reach one or more motor joint limits, causing the robot to stop in its measurements. While performing a stroke, two joints can also line up and become parallel, forcing the robot to dramatically increase its joint velocities to keep a constant pen tip velocity. When two joints are completely parallel, then the robot can not continue with the desired motion due to not being able to calculate the next interpolation step for its joint velocities and joint values—the robot has reached a singularity configuration that again causes the robot to stop in its measurements.

This causes unexpected, unplanned delays in the development process for the company since if the robot stops in its measurements, it may not be detected until several hours later.

This also means that one person must be assigned to oversee the robot during measurements. This adds to the stress in the working environment and wastes valuable time from other duties at the company.

Problem III. When working with the robot it is very useful to be able to control it manually, either to move it out of the way, or to program a path, or mark points of interests[4]. This is usually done with the ABB Flexpendant, a large, heavy control device equipped with a dead man's switch and a joystick with 3 degrees of freedom. This device requires training to use correctly and is connected to the robot controller by a long heavy cable that very easily gets caught in corners and other equipment.

It takes a long time to get good at controlling the robot with the flexpendant, and since it is a heavy device with a heavy cable in the way it is an energy consuming task for the operator to control the robot.

Solutions. In this thesis the first and second problems are solved by deriving and making use of the theory of the forward and inverse kinematics of the robot.

A calibration algorithm is derived with the forward kinematics to solve the first problem.

The second problem is solved by deriving the inverse kinematics and implementing it in Python. This is used together with a path-finding algorithm to find a smooth motion, without sudden stops, over a curve.

The third problem is solved by creating a framework in Python to parse and interpret the Bluetooth data sent from the pen to control the robot arm.

[4] These are usually positions of the mounted tool tip paired with the orientation of the tool, i.e., "poses".

2

Background

2.1 Anoto

Anoto Group AB[1] has over 150 employees with offices in Lund and Norrköping in Sweden as well as in the UK, US and Japan. The company sprung from C-Technologies (known by their text-digitizer product: the C-pen) and is one of the leading experts in active digital pens and digital writing and drawing solutions, see Figure 2.1.

Its technology platform and branded products enable high-precision pen or stylus input on nearly any surface, from capturing and digitizing handwritten notes and business forms on paper to designing, creating and collaborating directly on large interactive displays, whiteboards, and walls up to 24 feet.

ANOTO

THE POWER TO CAPTURE IT ALL

Figure 2.1 The Anoto logo and motto [Anoto AB, 2017b].

Source: Anoto AB.

As of this writing the Anoto family consists of 53 solution providers, among them Destiny Wireless, We-inspire, Hitachi, Vodafone and Penvision, together with Livescribe which is a wholly-owned subsidiary of Anoto that designs and manufactures the Livescribe brand of smartpens [Anoto AB, 2017a], [Livescribe, Inc, 2017]. Current and past partners in the United Kingdom healthcare system are The Perinatal Institute, that helps to collect perinatal information across the UK using Anoto digitizing forms, the Aneurin Bevan Health Board as well as the Welsh Ambulance Service that utilize the Anoto digitizing writing solution [Anoto Group AB,

[1] In interviews and in the sequel of this report the company is just referred to as "Anoto".

2016b]. Other partners that benefit from the Anoto digital forms are companies in the production and maintenance industry related to fields such as electric and diesel locomotives, railcars, process cooling, lifts, escalators and cradle maintenance [Anoto Group AB, 2016c]. The company has also extended its services to retail and logistics, financial sector as well as the public sector [Anoto Group AB, 2016a].

2.2 Anoto Dots Technology

The strength behind Anoto's digital writing and drawing solutions is its unique non-repeating dot-pattern which encodes absolute position by applying distorted dots relative to an invisible raster grid. The dots' positions relative to a raster intersection encodes the position. The raster grid and encoding dots make up a grid of 2×2-millimetre squares, with 36 dots in such a square. This creates a unique pattern in each square and a total of 4,722,366,482,869,645,213,696 unique squares can be generated[2] [Silberman, 2001]. This allows for sub-millimetre precision in decoding and interpolating the coordinate being decoded [Pettersson and Elsö, 1999], see Figure 2.2.

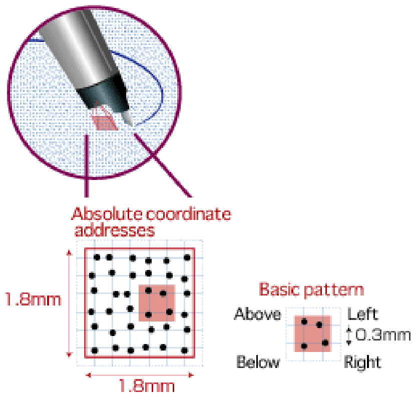


Figure 2.2 Absolute pen tip position is decoded from distorted dots relative a fictional raster grid from a 2×2-millimetre square on the Anoto surface. The filled area shows four intersections with distorted dots [Destiny Wireless, 2017].

Source: Destiny Wireless Ltd.

[2] This pattern makes up an area that is 4.7 million square kilometres – half the area of the US [Silberman, 2001].

2.3 Anoto Digital Pen Technology

Combining the dot-pattern technology with a small camera system in the form of a digital pen, with a Bluetooth-chip, pressure sensor enabled tip and infrared light emitting diode, that uses computer vision and image analysis, it is possible to decode the absolute position of the pen tip on the pattern-printed surface, as well as the orientation of the pen relative to the surface, see Figure 2.3.

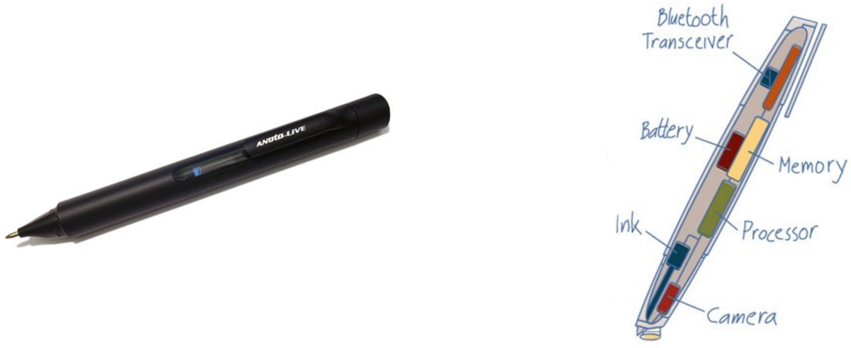


Figure 2.3 To the left: The Anoto *Live* pen with active display [Xms Penvision, 2017]. To the right: A generalized drawing of the components in the pen [NECS, inc, 2017]. The infrared emitting diode and force sensor are not annotated in the figure.

Sources: Xms Penvision AB (Left), Necs, inc. (Right).

This allows for capturing pen strokes as pen tip trajectories, enabling for on-the-fly, wireless digitization of handwritten text in real-time. Some uses for this technology involve signature verification, fraud detection, digitization of handwritten forms and digital whiteboards.

2.4 Robot ABB IRB140-6/0.8

This multi-purpose industrial robot model with a handling capacity of 6kg, seen in Figure 2.4, was designed for packing products, handling different materials, cleaning, spraying and for assembly production among other uses [ABB Ltd, 2016c]. Its relatively small size, with dimensions of $950\times800\times620\text{mm}^3$ and a weight of 98kg, and the fact it can be deployed in any angle on floors, walls and ceilings allows for deployment in a multitude of environments where space limitation is a factor.

A high-performance version of this manipulator exists under the name of the IRB140T which is 25% faster on the first two joints. There also exists a high-speed upgrade kit for the IRB140 that transforms it into the 140T—showing the flexible design of this robot manipulator [ABB Ltd, 2016d]. With the upgrade it is possible to obtain up to 15-20% decrease in operation cycle time if only the first two joints are utilized.

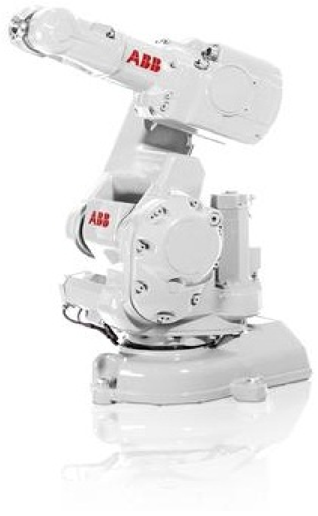


Figure 2.4 The 6 DOF IRB140 robot manipulator from ABB [ABB Ltd, 2016c].

Source: ABB.

Both a Foundry-version and a Wash-version[3] exist that make this robot suitable for use even in harsh environments.

The robot consists of six so-called revolute joints that allow for pure rotational movement along one given axis per joint. How this axis is related to the joint depends on that specific design as there are different variations of the revolution joints. This allows for 6 degrees of freedom (DOF) in the movement and orientation in the room for an attached tool, see Figure 2.5. Three degrees of freedom are needed to yield the position[4] and three more are required to yield the orientation of the end-effector. This means the robot by design resembles the freedom and flexibility you would obtain from an ordinary human arm, despite the robot arm having one less degree of freedom compared to the human arm's 7 DOF.

[3] High pressure steam washable version.
[4] The wrist center point.

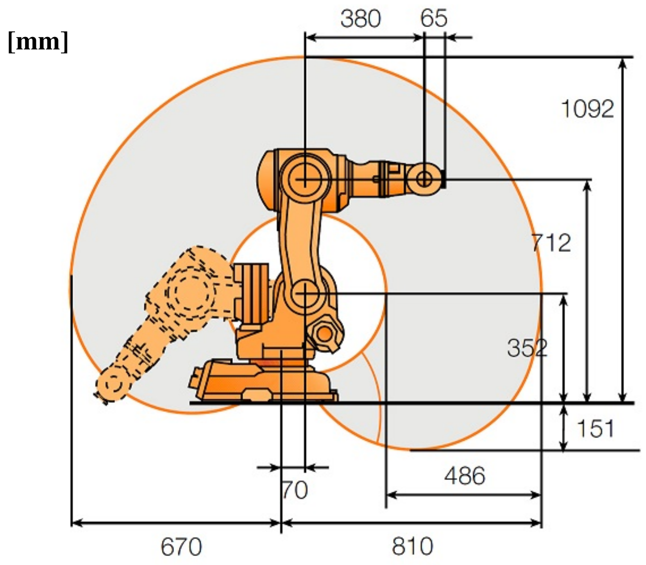


Figure 2.5 The full range of the robot workspace [ABB Ltd, 2016c].

Source: ABB.

ABB’s advancements in the field in motion control of robotic systems, together with complete dynamic models of the robots in the robot controllers, is what allows for the high standard of quality, accuracy and repeatability of following motion paths [ABB Ltd, 2016a]. The position repeatability[5] of the IRB140 model is 0.03mm [ABB Ltd, 2016c]. With the robots’ high accuracy and control it is possible to coordinate up to 4 different robots (36 different axis) at the same time [ABB Ltd, 2016a].

This has been demonstrated by ABB with the "Fanta Challenge" on their homepage [ABB Ltd, 2016a] with an embedded YouTube video [ABB Ltd, 2016b].

A summary of the technical specifications, e.g., the joint-limits of the IRB140 can be found in Appendix A.1.

[5] This precision is not obtained during motion or in absolute accuracy.

3

Literature Research

The need to relate measurements made by a camera to a different known coordinate system arises in many engineering applications. Historically, it appeared for the first time in connection with cameras mounted on robotic systems. This problem is commonly known as hand-eye calibration [Heller et al., 2014].

In many robotic applications it is necessary to determine the position and orientation of the tool and its tip for use in, e.g., robot assisted welding, machine loading, drilling, packaging and medical applications to name a few. Today in medical applications a camera (referred to as the "eye" in the robotics community) is placed on the robot hand to assist in precisely positioning the tool [Ernst et al., 2012], [Horaud and Dornaika, 1995] while in tracking-and-grasping applications it is used to grasp moving objects, e.g., products on a converyor belt [Allen et al., 1993].

The Anoto digital pen uses a camera to determine where its tip is located on the surface that makes use of the Anoto pattern, but in order to perform measurements with the IRB140, the robot needs to know where the surface and the pen tip is located relative to the robot base frame.

The ground work for solving these problems were laid out independently by Shiu and Ahmad [Shiu and Ahmad, 1989] and Tsai and Lenz[1], [Tsai and Lenz, 1989]. Both solutions make use of matrix algebra and the special properties of homogeneous matrices, separately determining the orientation of the tool being used and the position of its tip [Ernst et al., 2012]. The position of the tip and the orientation of the tool were solved as two decoupled problems, where the orientation was solved using either rotation matrices for rotations around an axis by an angle [Tsai and Lenz, 1989], [Shiu and Ahmad, 1989] or quaternions [Chou and Kamel, 1991].

[1] The problem is formulated as solving a set of equations on the form $\underline{\mathbf{AX}} = \underline{\mathbf{XB}}$, where $\underline{\mathbf{A}}$ is the relative transformation of the camera, $\underline{\mathbf{B}}$ the relative transformation of the robot flange and $\underline{\mathbf{X}}$ the relative transformation from the robot flange to the attached camera frame [Heller et al., 2014].

Horaud and Dornaika were the first to treat the orientation and position as coupled problems and solved for them simultaneously using a non-linear minimization approach [Horaud and Dornaika, 1995]. Chen introduced a new approach using screw-motion theory and performed a geometric analysis of the original problem and proved that the position and orientation problems become ill-conditioned when decoupled and should be solved for simultaneously [Chen, 1991]. Daniilidis introduced dual quaternions as an equivalent algebraic representation of screw-motion theory [Daniilidis, 1999]. A related problem is that of simultaneously solving for both the position and orientation of the tool and its tip as well as the position and orientation of the work object[2]. Zhuang derived a method of solving this problem using quaternions[3] [Zhuang et al., 1994].

These methods were derived on the assumption that orthogonal homogeneous transformations can be found that optimally solve for the tool and the work object[4]. Ernst [Ernst et al., 2012] proposed a method of calibration by extending earlier methods to allow non-orthogonal transformations to correct for system inaccuracies.

More recent and modern approaches include the attempt of Heller to solve hand-eye calibration as well as simultaneous robot-world calibration by restating them as multivariate polynomial optimization problems and use convex linear matrix inequality relaxations to obtain globally optimal solutions [Heller et al., 2014].

Strobl [Strobl and Hirzinger, 2006] presented a physically-based metric together with an experimentally validated error model for optimally estimating the solutions for the problems mentioned using the Maximum Likelihood method.

Ruland [Ruland et al., 2012] presented a globally optimal hand-eye self-calibration method and contributed new feasibility tests to integrate the hand-eye calibration problem into a branch-and-bound parameter space search. A guaranteed globally optimal estimator for simultaneous optimization of both tool tip position and orientation, with respect to a cost function based on reprojection errors, was presented and a benchmark dataset was published online to create a common point of reference for evaluation of hand-eye self-calibration algorithms.

[2] This related problem has the form $\underline{\mathbf{AX}} = \underline{\mathbf{YB}}$, where $\underline{\mathbf{Y}}$ is the relative transformation from the robot base-frame to the global world-frame.

[3] In most of these papers the work object is a calibration pattern used for calibrating the camera.

[4] This is not necessarily the case due to system inaccuracies, e.g., the robot is not perfectly constructed.

4

Method

4.1 Traversing a Path

When determining if the robot can move a given tool tip along a specific geometric path[1] under time-constraint[2,3] there are a number of obstacles that need to be considered. First, we need to know the robot can traverse the curve with the tool without having to change configuration from, e.g., forward-facing elbow-up configuration to backward-facing elbow-up in order to continue on the given path (joint limit problem) or being forced to change a joint 90 or 180 degrees between two neighbouring points on the curve (continuity problem). Second, we need to determine if the robot can traverse the path without exceeding joint-velocities (joint velocity-limit problem) and without any of the joints lock-up (singularity configuration problem). Last we need to determine if each point on the path is reachable. See summary below.

Problems when traversing a path:

1. One or multiple joints reach their joint limits.
2. One or multiple joints have to take very large motor angle steps between neighbouring curve points.
3. One or multiple joints reach their joint-velocity limits.
4. Two joints become parallel—the robot has reached a singularity configuration.
5. A part of the curve is unreachable.

[1] A geometric curve where each point point on the curve has an assigned tool-pose.
[2] If a time-constraint is exists then the path will also have velocity- and angular velocity vectors applied to each point on the curve.
[3] A geometric curve with time constraint is a trajectory.

In solving these problems we first solve the inverse kinematics problem for each pose $\mathbf{p}_i$ on the curve and assign the solutions $\mathbf{Q}_i = \{\mathbf{q}_{ik}\}_{k=\{1,2,\ldots,m_i\}}$ to the corresponding point on the curve, see Figure 4.1.

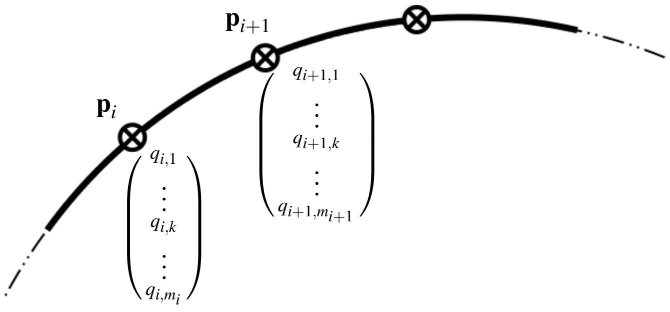


Figure 4.1 A curve with assigned poses and inverse kinematics solutions to specific points on the curve.

The Jacobians $\underline{\mathbf{J}}_{i,k}$ are calculated from the joint values $\mathbf{q}_{i,k}$ using forward kinematics and are used to solve for the joint velocities $\dot{\mathbf{q}}_{i,k}$ from the corresponding velocity vectors $\mathbf{v}_i$ and angular velocity vectors ω_i, see Figure 4.2.

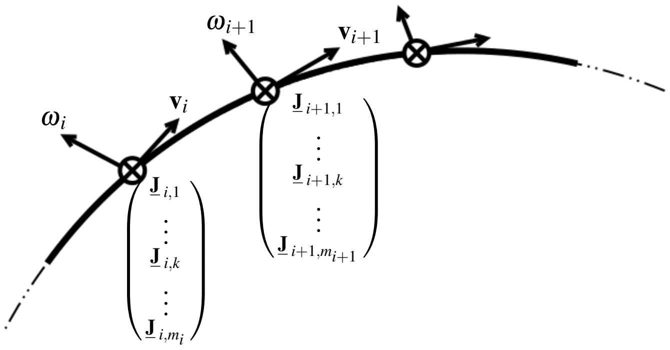

ω_{i+1}
$\mathbf{v}_{i+1}$
ω_i
$\mathbf{v}_i$
$\underline{\mathbf{J}}_{i+1,1}$
$\underline{\mathbf{J}}_{i+1,k}$
$\underline{\mathbf{J}}_{i+1,m_{i+1}}$
$\underline{\mathbf{J}}_{i,1}$
$\underline{\mathbf{J}}_{i,k}$
$\underline{\mathbf{J}}_{i,m_i}$

Figure 4.2 A curve with assigned velocity vectors and angular velocity vectors and Jacobian solutions to specific points on the curve.

A forward-directed graph is constructed from the solutions where each solution node from $\mathbf{p}_i$ is connected to every solution node for $\mathbf{p}_{i+1}$. A recursive path-finding algorithm is used to find the path of least L_2-norm angle-deviation from $\mathbf{q}_{1,k} \in \mathbf{Q}_1$ for $\mathbf{p}_1$ to $\mathbf{q}_{n,l} \in \mathbf{Q}_n$ for $\mathbf{p}_n$ with a tolerance constraint on the maximum allowed angle-deviation, see Figure 4.3. For each point the solutions are filtered for those where the joint values are all within their limits. Depending on the application the solutions can then be filtered a second time for problem specific constraints, e.g., only robot configurations of "backward-facing elbow-down" could be allowed or certain joints are only allowed to move within small angle ranges. Unreachable points will in Python result in invalid solutions as vectors of NaN[4] values and are automatically discarded during the filtering process.

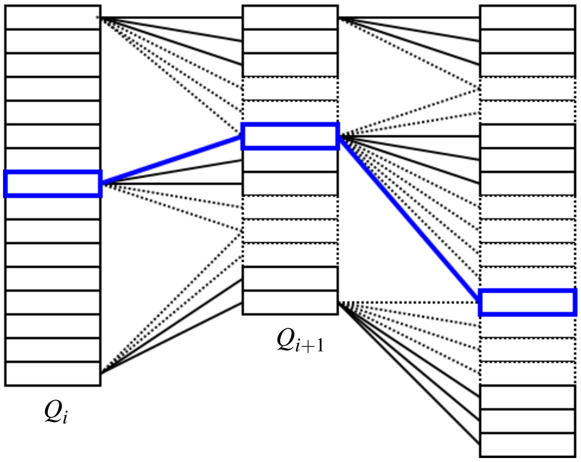


Figure 4.3 A path of least angle-deviation (blue) is found between each pose on the curve.

We should now have a collection of inverse kinematics solution paths that form one or more continuous paths in joint space. Each solution path is traversed and the corresponding Jacobians and joint velocities are calculated. If any of the joint velocities are outside their limits or any of the determinants of the Jacobians are close to zero then the whole solution path is discarded. The solution paths that are left will form continuous paths in joint space, see Figure 4.4. The trajectory and the resulting paths in joint space as well as the joint velocities can be further improved with spline interpolation.

4.2 Python/RAPID Framework

The robot is connected to a high-level controller, which interprets and executes instructions in the RAPID programming language and does not natively support real-time control nor does it support out-of-the box external control by Python scripts. It does, however, have good support for string-operations reminiscent of C# and Java and also supports late binding[5]. From here it is possible to construct a RAPID-Python interface using string communication through a serial port connec-

[4] Not a number—introduced in the IEEE 754 floating-point standard from 1985.
[5] Dynamic code execution from strings.

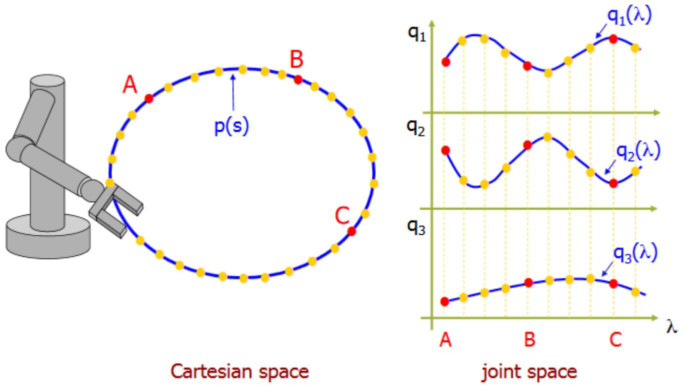


Figure 4.4 A trajectory in Cartesian space and its corresponding continuous path in joint space [Luca, 2015].

Source: A. De Luca, Sapienta University of Rome.

tion between an external computer[6] and the robot.

Such an interface would consist of a RAPID-program on the robot that listens to commands from the computer via the serial port, a RAPID module with procedures[7] that implement custom commands for the robot and updates the internal state of the robot, a Python class proxy implementation of the robot that keeps itself updated to the current state of the robot, and a Python serial interface implementation that is used behind the scenes of this virtual robot that glues the communication together between Python and RAPID, see Appendix A.7.

Similar to the robot-interface we need to implement a proxy pen-interface that connects to the pen via Bluetooth and keeps itself updated to the state-fields of the Anoto pen. The pen interface will differ from the robot interface in that it will implement a thread that keeps the state variables updated which is then fed to the robot interface in the main thread, which in turn will feed RAPID move commands to the robot depending on what the pen-interface is reporting, see Figure 4.5.

During this master thesis the pen interface (without a thread), robot serial interface and RAPID scripts were already implemented by Anoto. The RAPID scripts were improved on and extended in functionality and the robot proxy interface was added as an extension to this framework, the pen interface was extended by keeping track of the expected state of the pen and a thread was added to improve respon-

[6] This can be replaced by any device that can execute Python code and supports serial communication.
[7] Functions that have no return value.

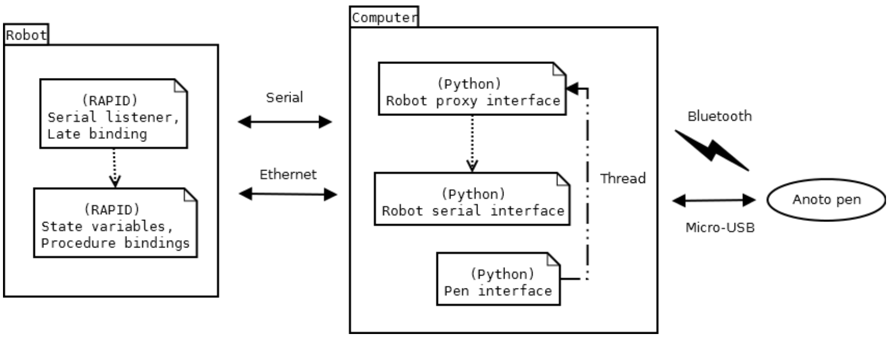


Figure 4.5 Overview of the components that make up the RAPID/Python binding which allows for communication between a robot and a digital pen.

siveness of the pen/robot communication by the author.

The serial port listener code pattern is located in arap.prg on the robot-controller and handles incoming data from the connected computer in the form of binary strings, see Listing 4.1.

```
    WHILE continue DO
      cmd_string := ReadStrBin2(comport \Time:=300000);
      TPWrite "Recv = " + cmd_string;
      ArapDoCommand( cmd_string );
      ! TODO: ArapHandler calls report func
      !       OR return string?
      WriteStrBin comport,"OK 200"+"\0a";
    ENDWHILE
```

Listing 4.1 RAPID (arap.prg): serial port listening.

The binary strings are parsed and converted to RAPID strings and sent to be executed as source code via late-binding in `ArapExecStr`, see Listing 4.2.

```
  WHILE readmore DO
      IF Present(Time) THEN
          character := ReadBin(dev \Time:=Time);
      ELSE
          character := ReadBin(dev);
      ENDIF

      TEST character
          CASE 10:
              readmore := FALSE;
          CASE 13:
              readmore := FALSE;
          DEFAULT:
              result := result + ByteToStr(character\Char);
```

Listing 4.2 RAPID (arap.prg): implementation of a string-builder from binary serial data.

In `ArapDoCommand` the RAPID strings are parsed for what RAPID procedures to call and what parameters to pass to them through the late-binding syntax, see Listing 4.3.

```
     CASE "ExecStr":
       get_next_arg callname, cmd;
       get_next_arg arg, cmd;
       ArapExecStr callname, arg;
```

Listing 4.3 RAPID (arap.prg): ArapDoCommand extracts function name and call parameters from input and passes them to ArapExecStr.

The late-binding procedures must have been loaded into memory when the strings are executed as code with the expression: `% call % arg`, see Listing 4.4.

```
! Executes the given procedure with string as parameter
LOCAL PROC ArapExecStr( string call, string arg )
    % call % arg;
  ERROR
    % a_rep_func % ERRNO, "ExecStr Failed";
ENDPROC
```

Listing 4.4 RAPID (arap.prg): input string is executed dynamically via late binding.

From here we implement the late-binding procedures, one such is initHAPTOR that initializes the the internal state variables, and is the first command sent from the Python robot interface during initializaton of the Python robot object, see Listing 4.5.

```
  PROC initHAPTOR()
      currSpeed := c_speed_parrot;
      setConfig( c_confdata );
      currConf := c_confdata;

      selToolAndWObj;
      currTool := sel_tool;
      currWobj := sel_wobj;
      c_oldPos := CRobT(\Tool:=currTool \Wobj:=wobj0);
      ConfL \Off;
      Open "HOME:" \File:="HAPTOR.LOG", logfile \Write;
      Close logfile;
  ENDPROC
```

Listing 4.5 RAPID (hptr_control.mod): binding procedure that initializes the state variables for the robot.

With the following binding-procedure it is possible to request for the current joint-values of the robot from the Python computer. The result is sent back as binary string data to the robot computer for processing, see Listing 4.6.

```
  PROC getJ()
      VAR jointtarget jtarget;
      VAR robjoint jvals;
      VAR iodev comport;
```

```

        jtarget := CJointT();
        jvals :=  jtarget.robax;

        Open "com2:", comport \Bin;
        WriteStrBin comport, ValToStr(jvals) + "\0a";
        Close comport;
    ENDPROC
```

Listing 4.6 RAPID (hptr_control.mod): binding procedure that send RAPID strings over serial port. Used to obtain current joint values.

Finally another procedure for controlling the robot with relative movement of the tool-center point in the global coordinate system, which allows for real-time control of the robot from Python, see Listing 4.7.

```
    PROC RelBasePos(string xyz)
        VAR pos rel_pos;
        VAR robtarget c_target;
        VAR bool ok;

        ok := StrToVal(xyz, rel_pos);
        TPWrite("OK: "+ValToStr(ok));
        TPWrite("new_pos: "+ValToStr(rel_pos));

        c_target := CRobT(\Tool:=currTool \Wobj:=wobj0);
        c_target := Offs(c_target, rel_pos.x, rel_pos.y, rel_pos.z);
        MoveL c_target, currSpeed, fine, currTool \WObj:=wobj0;
    ENDPROC
```

Listing 4.7 RAPID (hptr_control.mod): binding procedure that controls the robot. Used to move TCP in base-frame (global) coordinate system.

The information returned from the robot-controller to Python is converted from string to a Python expression with `ast.literal_eval`, in the case of retrieving the joint values from the robot the result is a Python list with six floating-point values, see Listing 4.8.

```
    def get_joints(self):
        while True:
            try:
                res = self.arap.command('getJ')
                i0 = res.find('[')
                i1 = res.find(']')+1
                res = res[i0 : i1]
                res = ast.literal_eval(res)
                if not(len(res) == 6):
                    raise Exception('Not valid joints format (length: {}'.format(len(res)))
                return mat(res)
            except Exception as e:
                print 'Failed to obtain joint-values!'
                print str(e)
```

Listing 4.8 String is converted to Python list type expression.

The computer connected to the robot controller via the first serial port uses a `RobotSerial` Python interface that implements `send` and `receive` methods for converting binary strings from the controller to Python strings and vice-versa, see Listing 4.9.

```
    def command(self, arapCommand='', argument=None, sync=True):
        """ Send command to Arap running on robot controller.
            if sync is True, call receive, otherwise continue (
    asynchronous call)
            TODO exception handling
        """
        if type(argument) in [tuple, numpy.ndarray]:
            argument = list(argument)
        _str = str(argument).replace(' ','');

        string_value = arapCommand + ' ' + _str

        if type(argument) == type(None):
            arapString = 'Exec ' + arapCommand
        elif type(argument)==int or type(argument)==float:
            arapString = 'ExecNum ' + string_value
        else:
            arapString = 'ExecStr ' + string_value

        log.info( 'arapString: {0}'.format(arapString) )
        result   = self.send(arapString)

        if (result == 0) & sync:
            result = self.receive()
            log.info( 'arapResult(sync): {0}'.format(result) )
        else:
            result = 'SYNC OFF'
            log.info( 'arapResult: {0}'.format(result) )

        if not result:
            if not self._debug:
                raise ArapException('can not execute command: %s'
 % str(arapCommand))
        return result
```

Listing 4.9 The command method makes use of send and receive to give formatted commands to the robot, and waits for a response to return until issuing the next command if sync is True, otherwise it will send and receive asynchronously. An exception is raised if the robot could not execute the command.

`Robot` is a wrapper class in Python that implements methods that the user will be using to send commands to the robot via `RobotSerial`. This makes it easier for the user to control the robot by encapsulating multiple corresponding RAPID commands into single Python methods like `move_to_door`, `move_to_ready`, `get_flange` and `get_joints`.

It also makes use of `ast.literal.eval` mentioned earlier to convert the Python strings from `RobotSerial` to Python expressions so that the user can receive a list of floats instead of a string representation of the result that the user will have to parse. This class also maintains an internal thread connection to a Python pen interface of an Anoto pen and keeps a copy of the `alive` state of the pen and will cease the control of the robot if this state should become false, e.g., the pen looses connection with the computer or stops responding, see Listing 4.10.

```python
class Robot

    def __init__(self,lock = None, angle=45, pen_interface=None,
    num_data=16):
        self._upload_files()
        self._init_serial()

        self.lock = lock
        self.all_data = []

        self.start_time = time.strftime('%H%M')

        self.alive = False
        self._finished = False
        self.num_data_points = num_data
        if pen_interface:
            self.pen = pen_interface
            self.pen_hit_thread = Thread(target=pen.check_hit)
            self.pen_hit_thread.start()
            time.sleep(1)
            with self.lock:
                self.alive = self.pen.alive
            if pen.alive:
                self.move_to_ready(angle)
                self.save_tool_pos()
```

Listing 4.10 A pen object is passed to the constructor of Robot where a thread runs its check_hit method.

4.3 Flexpendant Control Board Design

A simple design of a control board can be seen in Figure A.6 in Appendix A.8, where the Anoto dot-pattern has been omitted. The purpose of the design was to give an intuitive feeling while using it together with a digital pen. A dead-zone was chosen to allow the user to rest their hand on the board while holding the pen. This can be changed to become a re-orientation zone if the user wished. The arrows in the dead-zone indicate the positive x- and y-direction of the robot. A scrollbar type control was added on the side to allow the user to change to vertical movement by placing the pen in the red zone. The distance to the center of the scrollbar dictates the vertical speed and the distance to the resting-zone center depicts the speed parallel to the ground. This set-up does not allow for both vertical and planar movement at the same time and was chosen by design due to safety concerns from the author.

4.4 Measurements

All figures in this section can be found in Appendix A.9 and A.8.

Two pens were used for collecting data for the results in this master thesis: the Anoto *Live* pen and a prototype pen. The Anoto *Live* pen is an official product that has quality assurance associated with its design making it a sturdy and reliable digital pen, it also has a magnetic cradle used for charging the pen, see Figure A.8. The prototype pen is not as sturdy and will bend depending on how force is applied during use, see Figure A.9.

As reference a calibration tip was created with machine workshop tools by manual labour from a threaded steel rod and is firmly mounted on the robot using a wingnut, see Figures A.11 and A.12. It is much easier to get a precise tool calibration with a rigid steel tip and allows for a good calibration of the work object surface.

The work object surface is a piece of sturdy cardboard cut from a moving box approximately $841 \times 594\text{mm}^2$ in size with thirteen adhesive tape pieces $20 \times 20\text{mm}^2$ in size placed homogeneously over the underside of the surface of the card board, see Figure A.13. These are needed to keep the cardboard flat to the measuring surface, taping the sides and keeping it stretched will not be enough to prevent it from bending from the surface it is placed on. Anoto paper with dimensions $594 \times 210\text{mm}^2$ in size is stretched and taped onto the work object and is called the "target" which we will be measuring, see Figure A.14.

There are 5 very thin cross markers on the target placed at known Anoto coordinates with a precision of approximately 0.1mm, where the precision is limited by the printer quality. The relative distances of the markers to each other in millimetres are well known. By calibrating the work object with the calibration tip using these markers we get a good estimate on where the target, and the work object, are located relative the robot base coordinate system, see Figure A.7.

The robot has a metal plate and cover with screws attached to its flange where the pens are mounted by fastening them under the cover called a "pen holder", see Figure A.10. Because the pens did not fit, the device had to be modified by removing the cover and adding adhesive tape that the pens could be attached to, see Figures A.15 and A.16. Duct tape was stretched and used to fasten the pens very firmly to the pen holder for extra stability, see Figures A.17, A.18, A.19 and A.20.

Using the RAPID/Python framework and bindings a grid pattern of coordinates was generated by interpolation between the cross markers in millimetres and in Anoto distance[8] which could then be transformed to the robot base coordinate system from the work object calibration. The starting position for each measurement was calculated by translating from corresponding grid point using the work object normal and a starting distance of 3mm.

The pen tip tool-center point is roughly estimated relative the robot flange and is used to move the pen and pen holder to the starting positions and re-orient them relative this point. The relative estimated pen-tip direction is used as translation direction. When the pen registered that its tip was pressed down all information that was streamed from the pen was saved, as well as joint information and flange information from the robot, and returned the pen and holder to the starting position and orientation before continuing with the next measurement.

The robot was made to move 0.02mm/s which means that one measurement would take around 2.5 minutes to complete and measuring 240 points would take 10 hours, see Figures A.21 and A.22. As reference a manual calibration was performed for each pen and was compared against a marker location, see Figures A.23 and A.24. The measurements were then used with the tool tip calibration algorithm to generate the results in this master thesis.

[8] $1ad \approx 0.3mm.$

5 Theory

5.1 Forward Kinematics

Since the aim of this report is to solve a problem with a given design, without the option to alter it in any way—including control parameters, we will focus primarily on the kinematics of the robot geometry and how this relates to the kinematics of a particle following a curve, i.e., the tip of the pen.

No consideration is taken to the dynamics of the robot other than we trust the robot manipulator to do the best it can to conform to our wishes, i.e., following a trajectory with a pen tip. We also trust the robot to not perform over its abilities like over-heating the motors.

In general the forward kinematics problem is defined as, given a robot joint geometry configuration, find the mapping $\mathrm{F} : \mathbf{q}_i \rightarrow \mathbf{p}_i$, where a subset of known joints $\mathbf{q}_{i\in\mathbb{N}} = \{q_{1,i},\ q_{2,i},\ \ldots,\ q_{n,i}\}$ returns the corresponding pose of the end-effector $\mathbf{p}_i = \{\underline{\mathbf{R}}_p, \mathbf{t}_p\}_{i\in\mathbb{N}}$. The reason we say subset here is because multiple joint configurations will result in the same pose of the end-effector which in turn will yield the same corresponding pose for the tool.

By attaching frames to the joints of the robot that make up the orientation and position of each individual joint it is possible to define such a mapping with one (or more) so-called "kinematic chains" [Craig, 1986], see Appendix A.2. A kinematic chain is a series of relative homogenous transformations of translations and rotations, starting from a base-frame[1] to a desired destination frame (in robotics this is typically the end-effector), i.e.,

$$\underline{\mathbf{T}}_0^n = \underline{\mathbf{T}}_0^1\, \underline{\mathbf{T}}_1^2 \cdots \underline{\mathbf{T}}_{n-1}^n = \underline{\mathbf{A}}_1\, \underline{\mathbf{A}}_2 \cdots \underline{\mathbf{A}}_n \tag{5.1}$$

where each relative-transformation $\underline{\mathbf{T}}_{i-1}^i$ can, in theory, be defined by any convention the reader sees fit, but is ill-advised since arbitrary definitions spawned from

[1] The base frame does not have to be an inertial frame, but this report only deals with a robot that is firmly mounted.

personal preference has a tendency to quickly confuse fellow colleagues and it helps to have an agreed upon language that robotics researchers can use to quickly understand each other or papers in the same field—from here we use the Denavit-Hartenberg Convention[2] [Spong and Vidyasagar, 1989], [Hartenberg and Scheunemann, 1955] introduced in 1955.

Denavit-Hartenberg (DH) Convention

Apart from allowing researchers and engineers to communicate more easily, this conventions allows for one more benefit that must not be overlooked: by assigning frames and transforming between them in a clever way it is possible to describe a relative joint-transformation with only 4 degrees of freedom (two rotations, two translations) instead of the typical 6 degrees of freedom (three rotations, three translations) when dealing with rigid motion in Euclidean space in three dimensions. This is due to the frame-origins are allowed to be placed outside the physical joints and can exist in empty space, the proof of this is outside the scope of this report, but the interested reader is recommended to read pages 65 - 72 in [Spong and Vidyasagar, 1989].

The relative joint-transformation is defined as seen in Eq. (5.2) as a homogeneous matrix mapping from frame $\{i-1\}$ to frame $\{i\}$ where the end-result will be in frame $\{i-1\}$[3],

$$\underline{\mathbf{A}}_i = \underbrace{\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{rot_{z_{i-1}}(\theta_i)} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{trans_{z_{i-1}}(d_i)} \underbrace{\begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{trans_{x_i}(a_i)} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{rot_{x_i}(\alpha_i)} = \cdots$$

$$\cdots = \begin{pmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.2}$$

this mapping relation between joint-frames by homogeneous matrices is valid **iff** the following two conditions are upheld:

$$\text{The axis } x_i \text{ is perpendicular to axis } z_{i-1}\text{, i.e., } x_i \perp z_{i-1} \tag{DH1}$$

$$\text{The axis } x_i \text{ intersects axis } z_{i-1}\text{, i.e., } x_i \cap z_{i-1} \tag{DH2}$$

both conditions are met by applying the DH-convention, see Appendix A.6, thereby correctly defining the forward kinematics mapping as Eq. (5.1).

[2] This convention, alhough principally the same, varies between literatures—we are using the convention listed by the references.

[3] This means that if the we start from a frame in the base-coordinate system of the robot, then the result will be in the robot-base coordinates. If we start from joint 3, then the result will be in the frame of joint 3, and if that frame happens to be a pose in the base-frame—then the result will also be in the base frame. This is a very important detail that must be understood when reading the inverse kinematics section.

Table of DH Parameters

Using the DH-convention we end up with the following summarized table of the DH parameters for the IRB140, see Appendix A.5:

Link	a_i [mm]	α_i [rad]	d_i [mm]	θ_i [rad]
1	-70	$\frac{\pi}{2}$	352	$q_1+\pi$
2	360	0	0	$q_2+\frac{\pi}{2}$
3	0	$\frac{\pi}{2}$	0	$q_3+\pi$
4	0	$\frac{\pi}{2}$	380	$q_4+\pi$
5	0	$\frac{\pi}{2}$	0	$q_5+\pi$
6	0	0	65	q_6

Jacobian

As we have seen it is possible to find the position and orientation of an attached tool via forward kinematics by supplying a set of joint-angles to the Denavit-Hartenberg algorithm. it is also possible to find both the positional- and angular-velocity vectors for the tool end-effector in the robot base frame by derivation of the kinematic chain [Bruyninckx, 2010], [Freidovich, 2013], [Spong and Vidyasagar, 1989] which rigidly connects a point in the base frame $\mathbf{p}_0$ and a point in the last frame in the chain $\mathbf{p}_n$ with respect to time:

$$\mathbf{p}_0 = \underbrace{\begin{pmatrix} \mathbf{R}_0^n & \mathbf{t}_0^n \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}}_{\underline{\mathbf{T}}} \mathbf{p}_n \qquad (5.3) \qquad\qquad \frac{\mathrm{d}}{\mathrm{dt}}\left(\mathbf{p}_0\right) = \underbrace{\begin{pmatrix} [\omega] & \mathbf{v} \\ \mathbf{0}_{1\times 3} & 0 \end{pmatrix}}_{\frac{\mathrm{d}}{\mathrm{dt}}\left(\underline{\mathbf{T}}\right)\underline{\mathbf{T}}^{-1}} \mathbf{p}_0 \qquad (5.4)$$

$\left[\omega\right] = \omega \times \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, is a skew-symmetric matrix representation of the vector cross-product operation with the angular velocity vector ω[4]. From the derivation we obtain the following non-linear kinematic-relations between the

[4] $\omega \times \mathbf{I} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}.$

end-effector velocities and the robot-joints:

$$\begin{cases} \omega = \underline{\mathbf{J}}_{\omega}\big(\,\mathbf{q}(t)\,\big)\dfrac{\mathrm{d}}{\mathrm{dt}}\big(\,\mathbf{q}(t)\,\big) \\ \mathbf{v} \;\;= \underline{\mathbf{J}}_{v}(\,\mathbf{q}(t)\,)\dfrac{\mathrm{d}}{\mathrm{dt}}\big(\,\mathbf{q}(t)\,\big) \end{cases} \quad\Leftrightarrow\quad \begin{pmatrix}\omega\\ \mathbf{v}\end{pmatrix} = \underbrace{\begin{pmatrix}\underline{\mathbf{J}}_{\omega}\\ \underline{\mathbf{J}}_{v}\end{pmatrix}}_{\underline{\mathbf{J}}_{6\times n}}\dot{\mathbf{q}} \tag{5.5}$$

these non-linear relations are represented by linear mappings $\underline{\mathbf{J}}_{\omega}$ and $\underline{\mathbf{J}}_{v}$ that make up the **manipulator Jacobian** [Spong and Vidyasagar, 1989], [Craig, 1986], which after simplification [Freidovich, 2013] takes on the following form:

$$\underline{\mathbf{J}} = \begin{pmatrix}\underline{\mathbf{J}}_{v}\\ \underline{\mathbf{J}}_{\omega}\end{pmatrix} = \begin{pmatrix} \mathbf{z}_0^0\times(\mathbf{o}_0^n-\mathbf{o}_0^0) & \mathbf{z}_0^1\times(\mathbf{o}_0^n-\mathbf{o}_0^1) & \dots & \mathbf{z}_0^{n-1}\times(\mathbf{o}_0^n-\mathbf{o}_0^{n-1}) \\ \mathbf{z}_0^0 & \mathbf{z}_0^1 & \dots & \mathbf{z}_0^{n-1} \end{pmatrix} \tag{5.6}$$

this is a general manipulator Jacobian for a robot consisting of n revolute joints, where $\mathbf{z}_0^i$ and $\mathbf{o}_0^i$ are the rotational axis and corresponding origin obtained from the forward kinematic chain $\mathbf{T}_0^i$ of the $i:th$ frame represented in the base-frame of the robot[5].

5.2 Inverse Kinematics

For a robot-configuration with joint-dimension $n \in \mathbb{N}$ the inverse kinematics problem is defined as finding the mapping $M : \mathbf{p}_i \rightarrow \mathbf{Q}_i$ between a pose with position and orientation $\mathbf{p}_i = \{\underline{\mathbf{R}}_p, \mathbf{t}_p\}_{i\in\mathbb{N}}$, and the set $\mathbf{Q}_i = \{\mathbf{q}_{ik}\}_{k=\{1,\,2,\,\dots,\,m_i\}}$ of corresponding joint-values of size $m_i \in \mathbb{N}$ needed for the end-effector to attain this position and orientation, where $\mathbf{q}_{ik} = \{q_{1,i,k},\ q_{2,i,k},\ \dots,\ q_{n,i,k}\}$.

The number of solutions and the nature of the solutions of this type of problem vary depending on the geometry of the manipulator as well as the number of joints and type of joints the robot consists of. In the case with the six-jointed IRB140 of pure revolution-joints from ABB, the robot has two parallel joints that makes up an elbow, because of this the set of solutions can be separated into **elbow-up** and **elbow-down**, see Figure 5.1. The robot also has the ability to obtain a position and orientation by first turning itself around and then flipping over, i.e., rotating the first joint q_1 $\pm 180°$ and then rotating second or third (q_2 or q_3) joints until its wrist has passed the base-normal, separating the solutions into a second sub-set of combinations called **backward-facing** and **forward-facing** resulting in the set of four analytical solutions consisting of **forward-facing elbow-up**, **forward-facing elbow-down**, **backward-facing elbow-up** and **backward-facing elbow-down**. By its design

[5] This means, for instance, that $\mathbf{z}_0^0 = \begin{pmatrix}0\\0\\1\end{pmatrix}$ and $\mathbf{z}_0^1$ is defined by the first iteration in the kinematic-chain, see Section 5.1.

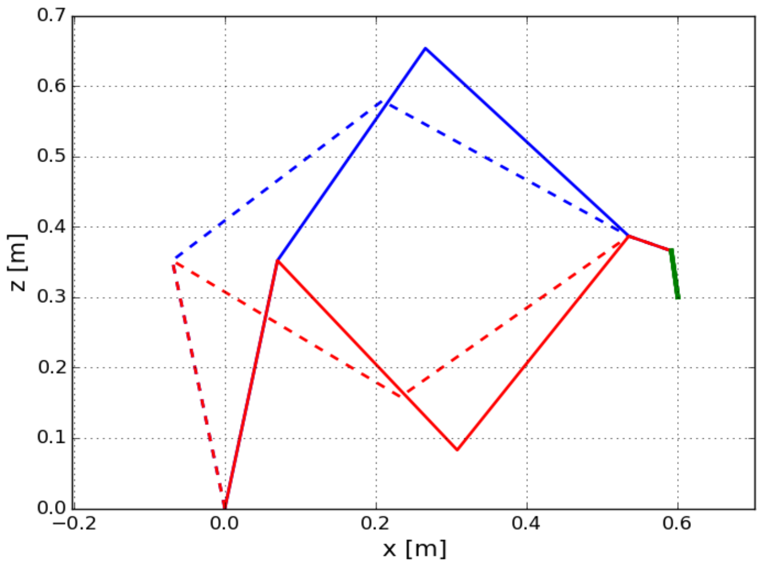


Figure 5.1 The IRB140 (links only) positioning itself so that the tool (green) tip is located at $(0.6, 0, 0.3)$. Shown here are the forward-facing (solid) and backward-facing (dashed) elbow-up (blue) and elbow-down (red) solutions.

the first three joint-angles q_1, q_2, q_3 describe the position of the so-called wrist center point $\mathbf{X}_{wcp}$ of the robot, while the last three joint-angles q_4, q_5, q_6 make up the spherical wrist, a design which will rotate the tool-center point $\mathbf{X}_{tcp}$ around the wrist center without displacement of the wrist—accurately describing the relative orientation and position of the tool-frame to the wrist-frame with the following relations:

$$\mathbf{X}_{wcp} = \mathbf{X}_{tcp} - d_6 \underline{\mathbf{R}}_0^6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.7}$$

$$\underline{\mathbf{R}}_{wrist} = \underline{\mathbf{R}}_4^6, \quad \underline{\mathbf{R}}_{tcp} = \underline{\mathbf{R}}_0^6 \tag{5.8}$$

in other words—the problem is divided by design into first solving for position (the first three joint-angles) of $\mathbf{X}_{wcp}$ and then orientation (last three joint-angles) for $\mathbf{X}_{tcp}$ respectively. Most of the difficulties arise when solving the first sub-problem of position, this is largely due to a large freedom in design decisions and angle definitions, while the second sub-problem of orientation becomes trivial in comparison when the first sub-problem is solved.

Because of the reasons stated the results for the forward-facing elbow-up solution configuration of IRB140 will be presented in full, with figures and code implementation and explanation of special cases. The other cases will have their results presented for q_2, q_3 together with the differing code from the elbow-up case only. The orientation sub-problem is presented last.

Wrist Center Position

For this type of problem we are given the pose of the end-effector from which we can calculate $\mathbf{X}_{wcp}$ with the known DH parameters by slightly modifying Eq. (5.7),

$$\mathbf{X}_{wcp} = \mathbf{t}_p - d_6 \underline{\mathbf{R}}_p \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.9}$$

from here the rest follows for all four cases of analytical solutions where the first joint-value is found from projecting $\mathbf{X}_{wcp}$ down to the XY-plane and calculating the angle to the global x-axis,

$$q_1 = \underbrace{\arctan 2(X_{wcp_x}, X_{wcp_y})}_{\theta_1}, \text{ forward-facing} \tag{5.10}$$

$$q_1 = \begin{cases} 180 \ + \ \theta_1, \ \ \theta_1 < 0 \\ -\ 180 \ + \ \theta_1, \ \ \theta_1 \geq 0 \end{cases}, \text{ backward-facing} \tag{5.11}$$

the solutions for q_2 and q_3 are connected where the solutions for q_3 determines the solutions for q_2 since these two angles make up the elbow-configuration for the robot. For all analytical cases the following relations are derived,

$$\theta_3 = \arccos\left(\frac{d_3^2 \ - \ x_1^2 \ - \ d_2^2}{2 \ \cdot \ x_1 \ \cdot \ d_2}\right) \tag{5.12}$$

$$q_3^{up} = \begin{cases} -90 + \theta_3, \text{forward-facing} \\ -90 - \theta_3, \text{backward-facing} \end{cases} \tag{5.13}$$

$$q_3^{down} = \begin{cases} -90 - \theta_3, \text{forward-facing} \\ -90 + \theta_3, \text{backward-facing} \end{cases} \tag{5.14}$$

$$q_3 = q_3^{\star}, \text{ where} \star \ = \ \{\text{up, down}\} \tag{5.15}$$

$$\theta_1' = \arctan 2(s, \ x_0) \tag{5.16}$$

$$\theta_2' = \arctan 2\Big(d_3 \ sin(\theta_3), \ d_2 \ + \ d_3 \ cos(\theta_3)\Big) \tag{5.17}$$

$$q_2^{up} = 90 - \left(\theta_1' + \theta_2'\right) \tag{5.18}$$

$$q_2^{down} = 90 - \left(\theta_1' - \theta_2'\right) \tag{5.19}$$

$$q_2 = \begin{cases} q_2^{\star}, & forward-facing \\ -\,q_2^{\star}, & backward-facing \end{cases}, \text{ where } \star = \{\text{up, down}\} \tag{5.20}$$

with these expressions we have to add exceptions to the forward-facing cases of Eq. (5.20) as follows,

$$\left.\begin{aligned} q_2^{up} &= -90 + \left(\theta_1' - \theta_2'\right) \\ q_2^{down} &= -90 + \left(\theta_1' + \theta_2'\right) \end{aligned}\right\} ||\mathbf{X}_{wcp_{x,y}}|| - ||\mathbf{p}_{0_{x,y}}|| < 0 \tag{5.21}$$

compare these with Eq. (5.18) and (5.19). The new parameter, $\mathbf{p}_0$, is the position of the axis that the second link in the kinematic chain rotates around, and is defined as

$$\mathbf{p}_0 = rot_{z_0}(\theta_1) \cdot \begin{pmatrix} 70 \\ 0 \\ 352 \end{pmatrix} \text{ [mm].} \tag{5.22}$$

The geometrical definitions can be seen in Figure 5.2 for a forward-facing elbow-up configuration.

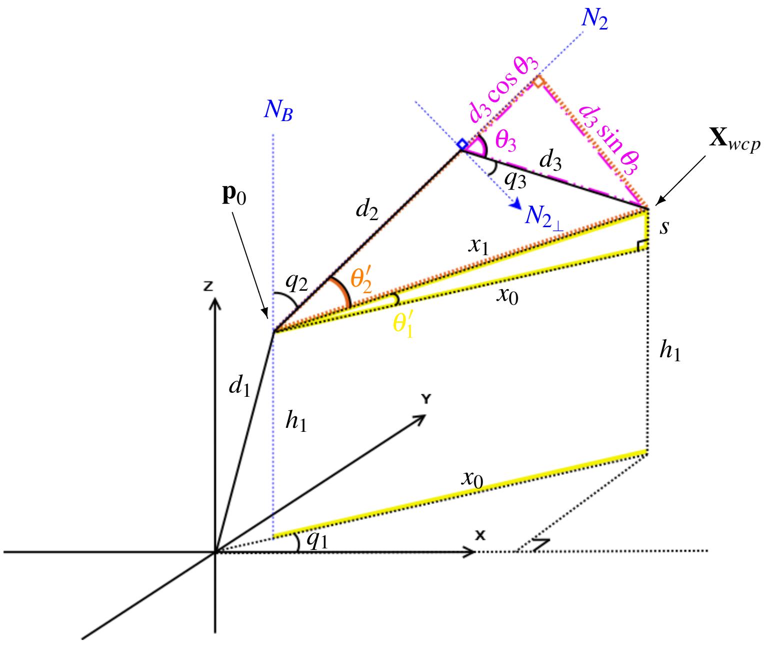


Figure 5.2 Forward-facing elbow-up configuration for IRB140, where only the first three links can be seen marked d_1, d_2 and d_3.

Wrist Center Orientation

Since we have defined a Denavit-Hartenberg representation for the IRB140, and we have solved the first part-problem of finding the joint-angles q_1, q_2, q_3 for the wrist center position, we can find the tool-flange $\mathbf{X}_{tcp}$ position and orientation in the local tool-center coordinate frame with the help of forward kinematics

$$\underline{\mathbf{T}} = \underline{\mathbf{T}}_0^1(q_1)\,\underline{\mathbf{T}}_1^2(q_2)\,\dots\,\underline{\mathbf{T}}_5^6(q_6) \tag{5.23}$$

$$\underline{\mathbf{T}}_0^3 = \underline{\mathbf{T}}_0^1(q_1)\,\underline{\mathbf{T}}_1^2(q_2)\,\underline{\mathbf{T}}_2^3(q_3) \tag{5.24}$$

$$\underline{\mathbf{T}}^{flange\star} = (\underline{\mathbf{T}}_0^3)^{-1}\underline{\mathbf{T}} = \underline{\mathbf{T}}_3^4(q_4)\,\underline{\mathbf{T}}_4^5(q_5)\,\underline{\mathbf{T}}_5^6(q_6), \tag{5.25}$$

$$\text{where } \underline{\mathbf{T}}^{flange\star} = \begin{pmatrix} \underline{\mathbf{R}}_3^6 & \mathbf{t}_3^6 \\ \mathbf{0} & 1 \end{pmatrix} \tag{5.26}$$

and from the DH parameters we can see that the orientation information, defined by each frames' rotation about the joint rotation-axis $\underline{\mathbf{Z}}_i$, from joint-frame 3 to frame 6 is by Euler-angle definitions a $\underline{\mathbf{R}}_3 = \underline{\mathbf{R}}_3^6 = \underline{\mathbf{ZYZ}}$ orientation, defined in base-frame for joint 3:

$$\underline{\mathbf{R}}_3^6(q_4,\, q_5,\, q_6) = \begin{pmatrix} c_4c_5c_6 - s_4s_6 & -c_6s_4 - c_4c_5s_6 & c_4s_5 \\ c_5c_6s_4 + c_4s_6 & c_4c_6 - c_5s_4s_6 & s_4s_5 \\ \underbrace{-c_6s_5}_{\mathbf{X}} & \underbrace{s_5s_6}_{\mathbf{Y}} & \underbrace{c_5}_{\mathbf{Z}} \end{pmatrix} \tag{5.27}$$

$$\text{where } c_i = \cos(q_i) \text{ and } s_i = \sin(q_i) \tag{5.28}$$

for which the solutions for the angle-values that make up this type of matrix are well known and can be derived from Eq. (5.27) to be:

$$q_4' = \arctan 2(Z_2,\, Z_1) \tag{5.29}$$

$$q_5' = \arctan 2(\sqrt{Z_1^2 + Z_2^2},\, Z_3) \tag{5.30}$$

$$q_6' = \arctan 2(Y_3,\, -X_3) \tag{5.31}$$

$$q_4'' = q_4' \pm 180° \tag{5.32}$$

$$q_5'' = -q_5' \tag{5.33}$$

$$q_6'' = q_6' \pm 180° \tag{5.34}$$

where all of the analytical solutions for the last three joint-values of the inverse kinematics problem span the range $[-180,\, 180]$, but we need to add solutions with multiples of $\pm$ 360 ° for two or more joint-values for two reasons:

The first is that for a general 6 DOF robot it is still possible we will miss solutions if we skip this step, but that depends also on the joint-restrictions. For instance, when solving for q_2 and q_3 for IRB140 we will not need to add any multiplicity due to the joint-restrictions of the joint-configurations for that manipulator design. If we on the other hand modify this manipulator-design so that the base is removed and joint 2 is wall-mounted and we loosen the restriction on joint 2 from $[-90,\ 110]$ to $[-180,\ 180]$ then we will not be able to find the equivalent valid solutions of $q_2^\star - 360°$ for different q_3 with the current mapping. This will hold true for other designs that for a given mapping of the inverse kinematics we need to add solutions of multiplicity $\pm 360°$ for some, if not all joints, to be sure to find all solutions.

The second reason is that $q_4 \in [-200,\ 200]$ and $q_6 \in [-400,\ 400]$ have values that are larger than $\pm$ 180 ° and since the analytical solutions have a span of 360° and these joint-values have spans of 400° and 800° we need to add solutions with multiplicity $n\ =\ 1$ for q_4 and $n\ =\ \{1,\ 2\}$ for q_6.

The 5 solutions for the last three joint-values are as follows:

$$q_4 = \begin{cases} q_4' \pm 360n\,° \\ \\ q_4'' \pm 360n\,° \end{cases} ,\ n\ =\ \{0,\ 1\} \tag{5.35}$$

$$q_5 = \begin{cases} q_5' \\ \\ q_5'' \end{cases} \tag{5.36}$$

$$q_6 = \begin{cases} q_6' \pm 360n\,° \\ \\ q_6'' \pm 360n\,° \end{cases} ,\ n\ =\ \{0,\ 1,\ 2\} \tag{5.37}$$

giving us a total number of solution configurations to: 5 wrist-configurations × 4 elbow-configurations = 20 total configurations + 2 × 3 solution multiplicities × 4 elbow-configurations yielding a total of 44 valid solutions for the IRB140 for one single pose.

5.3 Calibration with Anoto Pen

When working with an automated tool manipulator such as a robotic arm assembly it is important to make the calibration process as user-friendly as possible. Currently the IRB140 arm has a built-in calibration mode which involves manually operating the arm holding a pen and moving it so that the pen tip touches the Anoto Surface$^{\mathrm{TM}}$

at a predefined point from different orientations of the pen relative this calibration point. By then doing a least-square solution the **tool-center point**, $\mathbf{X}_{tcp}$ is obtained.

Besides the fact that this can be a time consuming task it is also a very sensitive method of calibration since the user performing the calibration will induce errors and each calibration operation can yield very different results which, in turn, might force the user to perform the calibration task multiple times until a consistent calibration is obtained. If also multiple pens are to be grip-calibrated before tests can be performed it becomes obvious that this part is crucial to automate.

First off it is important that the user is removed from the grip-calibration procedure as much as possible and allow the robot to do most of the work: from figuring out the location and orientation of the surface it is going to operate on with the pen to also figuring out how the robot is gripping the pen.

Due to the theory behind the Anoto Surface and the Anoto Pen it is possible to minimize the error of the surface-placement to sub-millimetre precision by removing the constraint of calibrating against a single well-defined point. The user is asked to move the arm so that the end-effector holding the pen is placed close to the Anoto Surface. The user then initiates a script which will continue to operate the arm and start moving it in the tool-frame Z-direction until we get a force-response from the pen. The arm will then raise the pen, move in an arbitrary tool-frame basis direction first in X-direction then in Y-direction and again move the pen closer to the Anoto Surface until we get a force response. Once three points are obtained it is possible to solve for the orientation and the placement of the Anoto Surface.

First a robot-coordinate system is defined with the origin $\mathbf{O}^{\omega}$ located in the symmetric center of the base-piece of the robot and three orthonormal Euclidean-space basis vectors $\hat{\mathbf{x}}^{\omega}$, $\hat{\mathbf{y}}^{\omega}$, $\hat{\mathbf{z}}^{\omega}$ are introduced with arbitrary orientation thus spanning an ordinary Euclidean space in $\mathbb{R}^3$, see Figure 5.3.

The pen coordinate system is the typical Euclidean-space in $\mathbb{R}^2$ where a coordinate is denoted as:

$$\mathbf{x} = \hat{\mathbf{x}}\,x + \hat{\mathbf{y}}\,y = \underbrace{\begin{pmatrix} \hat{\mathbf{x}}_x & \hat{\mathbf{y}}_x \\ \hat{\mathbf{x}}_y & \hat{\mathbf{y}}_y \end{pmatrix}}_{\underline{\mathbf{e}}} \begin{pmatrix} x \\ y \end{pmatrix} \tag{5.38}$$

and is related to the pen tip $\mathbf{x}'$ in the local Anoto Surface coordinate system by a submatrix transformation $[\mathbf{R}_1\ \mathbf{R}_2]$: $\mathbb{R}^2 \to \mathbb{R}^3$ rotation from the rotation tensor $\underline{\underline{\mathbf{R}}}_0^6$

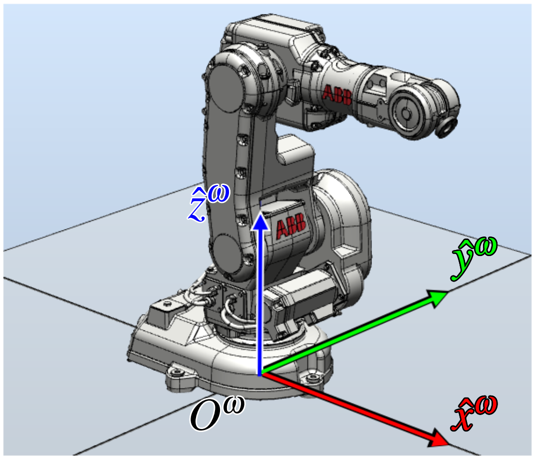


Figure 5.3 The robot-coordinate system.

Source: ABB.

and a translation $\mathbf{r}_{\mathbf{O}^{\omega}\mathbf{O}_{an}}$ in the global robot coordinate system in $\mathbb{R}^3$:

$$\mathbf{x}' = \underbrace{\mathbf{O}_{an} - \mathbf{O}^{\omega}}_{\mathbf{r}_{O^{\omega}\mathbf{O}_{an}}} + \underbrace{[\mathbf{R}_1\ \mathbf{R}_2]\,\underline{\mathbf{e}}}_{\underline{\mathbf{e}}_{an}} \begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{r}_{O^{\omega}O_{an}} + x\,\hat{\mathbf{x}}_{an} + y\,\hat{\mathbf{y}}_{an} \tag{5.39}$$

where $\{\underline{\mathbf{e}}_{an}, \mathbf{O}_{an}\}$ together defines the transformed local coordinate system of a plane in the global robot coordinate system, see Figure 5.4, with the transformed local orthnormal basis defined as:

$$\underline{\mathbf{e}}_{an} = \begin{pmatrix} \hat{\mathbf{x}}_{an_x} & \hat{\mathbf{y}}_{an_x} \\ \hat{\mathbf{x}}_{an_y} & \hat{\mathbf{y}}_{an_y} \\ \hat{\mathbf{x}}_{an_z} & \hat{\mathbf{y}}_{an_z} \end{pmatrix} = [\hat{\mathbf{x}}_{an}\ \hat{\mathbf{y}}_{an}] \tag{5.40}$$

it is possible to relate the pen tip $\mathbf{x}'$ to the robot tool-center point $\mathbf{X}_{tcp}$ in the global coordinate system by introducing a rotated deviation vector from local to global space δ' from the tool-center point to the pen tip:

$$\mathbf{x}' = \mathbf{X}^{o}_{tcp} + \underline{\mathbf{R}}^{6}_{0}\delta^{o}, \;\; \underline{\mathbf{T}} = \begin{pmatrix} \underline{\mathbf{R}}^{6}_{0} & \mathbf{X}^{o}_{tcp} \\ 0 & 1 \end{pmatrix} \Leftrightarrow \mathbf{x}' = \underline{\mathbf{T}}\delta' \tag{5.41}$$

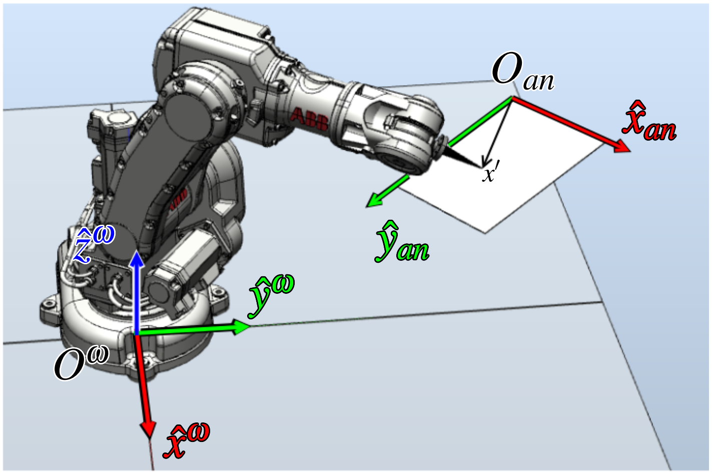


Figure 5.4 Global robot coordinate system and local rotated Anoto Surface plane coordinate system. The robot is seen here holding a black pen.

Source: ABB.

since we control the orientation of the robot then $\mathbf{X}_{tcp}$ is known, but both the pen tip $\mathbf{x}'$ and the deviation vector δ' are not since we do not yet know the orientation of the Anoto Surface plane-coordinate system, but we do know where on the Anoto surface the pen tip is intersecting in local plane coordinates.

By substituting in (5.41) with the definition for the pen tip in the global coordinate system from (5.39) we obtain the final relation for the tool-center point position in the global space and the pen tip:

$$\mathbf{X}^{o}_{tcp} + \underline{\mathbf{R}}^{6}_{0}\delta^{o} = \mathbf{r}_{O^{\omega}O_{an}} + x\,\hat{\mathbf{x}}_{an} + y\,\hat{\mathbf{y}}_{an} \tag{5.42}$$

this system of equations consists of 3 equations and 12 unknowns, the unknowns being $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\mathbf{r}_{O^{\omega}O_{an}}$ and δ^{o} - 4 variables of 3 dimension.

This relation holds for one configuration of the flange end-effector manipulating a pen on the Anoto Surface. By lifting the pen, reorienting the end-effector and moving it to another location above the Anoto paper and again lowering the pen to contact with the surface we will by taking the difference on all changing vectors find an equivalent relation for moving from one configuration to another resulting in another system of equations with 9 unknowns:

$$\Delta\mathbf{X}^{o}_{tcp} + \Delta\underline{\mathbf{R}}\delta^{o} = \Delta x\,\hat{\mathbf{x}}_{an} + \Delta y\,\hat{\mathbf{y}}_{an} \tag{5.43}$$

in order to solve either under-determined systems we will need at minimum 4 points, but generally more points are needed to obtain a good tool calibration result—

resulting in the following overdetermined system of equations, which are solved in the least-square sense, on the form:

$$\mathscr{A}\chi = \mathscr{b}\,, \tag{5.44}$$

where the system of equations is:

$$\mathscr{A} = [\mathscr{A}_1\ \mathscr{A}_2\ \mathscr{A}_3] \tag{5.45}$$

$$\mathscr{A}_1 = \begin{bmatrix} \Delta x_{01} & \Delta y_{01} & -\Delta r11_{01} \\ 0 & 0 & -\Delta r21_{01} \\ 0 & 0 & -\Delta r31_{01} \\ \vdots & \vdots & \vdots \\ \Delta x_{N(N-1)} & \Delta y_{N(N-1)} & -\Delta r11_{N(N-1)} \\ 0 & 0 & -\Delta r21_{N(N-1)} \\ 0 & 0 & -\Delta r31_{N(N-1)} \end{bmatrix} \tag{5.46}$$

$$\mathscr{A}_2 = \begin{bmatrix} 0 & 0 & -\Delta r12_{01} \\ \Delta x_{01} & \Delta y_{01} & -\Delta r22_{01} \\ 0 & 0 & -\Delta r32_{01} \\ \vdots & \vdots & \vdots \\ 0 & 0 & -\Delta r12_{N(N-1)} \\ \Delta x_{N(N-1)} & \Delta y_{N(N-1)} & -\Delta r22_{N(N-1)} \\ 0 & 0 & -\Delta r32_{N(N-1)} \end{bmatrix} \tag{5.47}$$

$$\mathscr{A}_3 = \begin{bmatrix} 0 & 0 & -\Delta r13_{01} \\ 0 & 0 & -\Delta r23_{01} \\ \Delta x_{01} & \Delta y_{01} & -\Delta r33_{01} \\ \vdots & \vdots & \vdots \\ 0 & 0 & -\Delta r13_{N(N-1)} \\ 0 & 0 & -\Delta r23_{N(N-1)} \\ \Delta x_{N(N-1)} & \Delta y_{N(N-1)} & -\Delta r33_{N(N-1)} \end{bmatrix} \tag{5.48}$$

and the unknowns we're solving for are:

$$\chi = \begin{bmatrix} \hat{x}_{an_x} \\ \hat{y}_{an_x} \\ \delta_x^o \\ \hat{x}_{an_y} \\ \hat{y}_{an_y} \\ \delta_y^o \\ \hat{x}_{an_z} \\ \hat{y}_{an_z} \\ \delta_z^o \end{bmatrix} \tag{5.49}$$

while the right-hand side consists of:

$$\underline{b} = \begin{bmatrix} \Delta X_{tcpx_{01}} \\ \Delta X_{tcpy_{01}} \\ \Delta X_{tcpz_{01}} \\ \vdots \\ \Delta X_{tcpx_{N(N-1)}} \\ \Delta X_{tcpy_{N(N-1)}} \\ \Delta X_{tcpz_{N(N-1)}} \end{bmatrix}. \tag{5.50}$$

5.4 Calibration of Pen Orientation

From rigid body kinematics we know that since the tool is attached rigidly to the flange of the robot arm there exists a constant relative rotation transformation from the flange orientation to the orientation of the tool. From the calibration algorithm we obtain the relative translation from the flange to the tool tip, and the global orientation and position of the Anoto Surface work object, in the robot base frame—but we do not obtain the tool-orientation.

The Anoto Pen is capable of calculating the relative **ZXZ**-Euler angles to the Anoto Surface in the Anoto Surface frame, and the flange orientation is obtained by forward kinematics and is described in the robot base frame. By using the information of the orientation of the work object from the previous calibration we obtain the pose orientation of the tool. This means that each time the robot touches the Anoto Surface with an Anoto Pen we obtain rotation pairs $\{\mathbf{R}_{\mathrm{P},i},\ \mathbf{R}_{\mathrm{F},i}\}$ that are rigidly connected by a constant rotation transformation $\mathbf{R}_{\mathrm{T}}$:

$$\underline{\mathbf{R}}_{\mathrm{P},i}\underline{\mathbf{R}}_{\mathrm{T}} = \underline{\mathbf{R}}_{\mathrm{F},i} \text{ where } i \in \mathbb{N}. \tag{5.51}$$

By performing N measurements we end up with the vast $3N \times 3$ system of equations:

$$\begin{cases} \underline{\mathbf{R}}_{\mathrm{P},1}\underline{\mathbf{R}}_{\mathrm{T}} = \underline{\mathbf{R}}_{\mathrm{F},1} \\ \vdots \\ \underline{\mathbf{R}}_{\mathrm{P},N}\underline{\mathbf{R}}_{\mathrm{T}} = \underline{\mathbf{R}}_{\mathrm{F},N} \end{cases} \tag{5.52}$$

This system of equations can be written as:

$$\underline{\underline{\mathbf{AC}}} = \underline{\mathbf{B}} \tag{5.53}$$

and is solved for $\underline{\mathbf{C}}$ in the least-square sense by the normal equations:

$$\underline{\mathbf{C}} = (\underline{\mathbf{A}}^T\underline{\mathbf{A}})^{-1}\underline{\mathbf{A}}^T\underline{\mathbf{B}} \text{ where } \underline{\mathbf{R}}_{\mathrm{T}} = \underline{\mathbf{C}}. \tag{5.54}$$

If the measurements are affected by noise (which they are) then the solution $\underline{\mathbf{R}}_{\mathrm{T}}$ is not a valid rotation matrix, as its basis vectors will not be orthonormal. This is handled by orthogonalizing the solution by finding the closest rotation matrix $\underline{\tilde{\mathbf{R}}}$ by Singular Value Decomposition (SVD) [Myronenko and Song, 2009], [Umeyama, 1991]:

$$\underline{\mathbf{R}}_{\mathrm{T}} = \underline{\mathbf{U}\Sigma\mathbf{V}}^T \qquad \underline{\tilde{\mathbf{R}}} = \underline{\mathbf{U}\mathbf{V}}^T \tag{5.55}$$

if the determinant of $\underline{\tilde{\mathbf{R}}}$ is not equal to 1, but instead -1, then the solution is not a rotation transformation but a reflection transformation and the orthogonalization has to be modified:

$$\underline{\tilde{\mathbf{R}}} = \underline{\mathbf{U}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \operatorname{sign} det(\underline{\mathbf{U}\mathbf{V}}^T) \end{pmatrix} \underline{\mathbf{V}}^T \tag{5.56}$$

5.5 Replacing the Flexpendant

Bluetooth

Bluetooth is a standard for short range, low power, low cost wireless communication that uses radio technology. Although originally envisioned as a cable-replacement technology by Ericsson (Sweden) in 1994 [McDermott-Wells, December 2004 / January 2005] the technology was conceived as a wireless alternative to data cables by exchanging data using radio transmissions and was created as an open standard to allow connectivity and collaboration between disparate products and industries [Bluetooth Special Interest Group, 2016b]. The name is inspired by Harald Bluetooth, a viking king who united the warring factions of Sweden, Denmark and Norway[6].

Human Interface Device (HID)

A Bluetooth HID device is a device providing the service of human or other data input and output to and from a Bluetooth HID Host. Examples of Bluetooth HID devices are keyboards, mice, joysticks, gamepads, remote controls, and also voltmeters and temperature sensors. A Bluetooth HID Host is a device using or requesting the services of a Bluetooth HID device. Examples would be a personal computer, handheld computer, gaming console, industrial machine, or data-recording device[7] [Bluetooth Special Interest Group, 2016a].

[6] In the same manner the Bluetooth technology lets different devices unite and communicate together.

[7] This specification incorporates significant portions of the USB (Universal Serial Bus) Device Class Definition for Human Interface Devices [Compaq et al., 2016].

Bluetooth HID Descriptor

The HID descriptor describes how the data in the HID reports being sent from a Bluetooth-device is structured, and follows the same convention as that from the USB HID descriptors. Besides telling the application how the data is structured it is also informing the receiving device, e.g., a computer or smart device on its properties. For instance, one type of digitizer pen could only report position coordinates, while another reports both position coordinates and pressure values. The application on its end chooses what to do with this information. A simple drawing program would ignore the fact that the digitizer pen can report angles or pressure while Adobe Photoshop and Gimp do have support for these features. When it comes to smart devices both Microsoft [Microsoft, 2015] and the Android Open-Source Project led by Google [Google Inc., 2016] have specifications on standard descriptors that specific devices must adhere to[8]. The HID descriptor for Android 6.0 devices can be seen in Listings 5.1 and 5.2.

```
unsigned char HID_DESC[] = {
     0x05, 0x0D, // UsagePage(Digitizer)
     0x09, 0x02, // Usage(Pen)
     0xA1, 0x01, // Collection(Application)
     0x09, 0x20, // Usage(Stylus)
     0xA1, 0x02, // Collection(Logical)
     0x09, 0x30, // Usage(Tip Pressure)
     0x15, 0x00, // Logical Minimum(0)
     0x26, 0xFF, 0x03, // Logical Maximum(1023)
     0x95, 0x01, // Report Count(1)
     0x75, 0x0A, // Report Size(10)
     0x81, 0x02, // Input(Data, Variable, Absolute, No Null)
     0x09, 0x44, // Usage(Barrel Switch)
     0x09, 0x5A, // Usage(Secondary Barrel Switch)
     0x09, 0x42, // Usage(Tip Switch)
     0x09, 0x3C, // Usage(Invert)
     0x25, 0x01, // Logical Maximum(1)
     0x95, 0x04, // Report Count(4)
     0x75, 0x01, // Report Size(1)
     0x81, 0x02, // Input(Data, Variable, Absolute, No Null)
     0x09, 0x5B, // Usage(Transducer Serial Number)
     0x95, 0x01, // Report Count(1)
     0x75, 0x80, // Report Size(128)
     0xB1, 0x03, // Feature(Constant, Variable)
     0xC0, // End Collection
     0xC0, // End Collection
 }
```

Listing 5.1 Raw Android 6.0 digitizer pen descriptor. Here implemented in C. [Microsoft et al., 2016]

[8] Relevant for this report are their HID descriptor specifications for digital pens.

```
UsagePage(Digitizer)
Usage(Pen)
Collection(Application)
    Usage(Stylus)
    Collection(Logical)
        Usage(Tip Pressure)
        Logical Minimum(0)
        Logical Maximum(1023)
        Report Count(1)
        Report Size(10)
        Input(Data, Variable, Absolute, No Null)
        Usage(Barrel Switch)
        Usage(Secondary Barrel Switch)
        Usage(Tip Switch)
        Usage(Invert)
        Logical Maximum(1)
        Report Count(4)
        Report Size(1)
        Input(Data, Variable, Absolute, No Null)
        Usage(Transducer Serial Number)
        Report Count(1)
        Report Size(128)
        Feature(Constant, Variable)
    EndCollection
EndCollection
```

Listing 5.2 Parsed Android 6.0 digitizer pen descriptor. [Microsoft et al., 2016]

The Anoto digital pens follow the specifications from Microsoft for digital pens and are very similar to the Android 6.0 specification, the exception being added HID usage fields for tilt-x, tilt-y and twist. The HID descriptor for the Anoto pen used in this report can be seen in Appendix A.4.

Bluetooth HID Reports

The data from the Anoto pens' HID reports is best represented as a collection of 20 pairs of hexadecimal digits. These will be resolved to their corresponding values of pen-tip coordinates, pressure, etc. in a parsing process as we receive the reports from the Anoto pens. From the Anoto HID descriptor[9] we see that the data is structured according to Figure 5.5, with 8 fields of varying sizes in bytes for each property of the Anoto pen digitizer. The byte-order of the data in each field is in big-endian format. In a word of bytes where the order of bytes follow {`most significant byte, higher order byte, lower order byte, least-significant byte`} it is referred to as "big-endian" format. If the bytes are sorted in reverse order by significance then it is referred to as "little-endian" format. Depending on which processor Python is running on we might need to sort the hexadecimal byte pairs in reverse order[10] before we can

[9] see Appendix A.4.

[10] An alternative solution is to use struct.unpack with the correct endian-decoder.

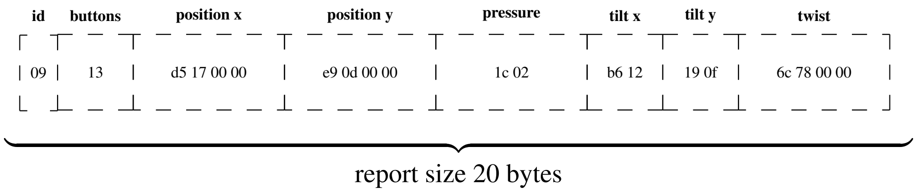


Figure 5.5 Bluetooth pen digitizer report from an Anoto pen.

convert the field values into their integer-representation counterpart[11].

In the Ubuntu distribution of Linux you can obtain both the parsed HID reports and the HID descriptor of a Bluetooth device by using the terminal command "cat" on two files named "events" and "rdesc" in /sys/kernal/debug/hid/... . Using this method as reference it is easier to figure out whether your system, and in turn Python, is using the same endianess as the Bluetooth device and if you need to reverse the data byte-wise or not.

The corresponding parsed values of Figure 5.5 can be seen in Listing 5.3.

```
Digitizers.TipSwitch = 1
Digitizers.BarrelSwitch = 1
Digitizers.Eraser = 0
Digitizers.Invert = 0
Digitizers.InRange = 1
GenericDesktop.X = 6101
GenericDesktop.Y = 3561
Digitizers.TipPressure = 540
Digitizers.003d = 4790    # tilt_x
Digitizers.003e = 3865    # tilt_y
Digitizers.0041 = 30828   # twist
```

Listing 5.3 Parsed Anoto raw HID report. (Linux/Ubuntu)

[11] For an Intel i3-2367M Python integers follow little-endian format by default.

6

Results

Here we present the final results of this master thesis with the results of the calibration algorithm presented first, with the inverse kinematics presented after and finally the remote control application by parsing Bluetooth HID data presented last.

6.1 Calibration Algorithm Measurements

The following figures show the calibration algorithm performance using 240 measurements collected for 10 hours.

Each plot represents the pen tip error against both tilt and skew angles in degrees of the pen[1].

The pen tip errors were calculated by using forward kinematics to calculate the expected pen tips, and then compared these with the pen tip positions reported from the Anoto pen on a reference Anoto Surface with known position and orientation.

Figure 6.1 shows the errors from a manual calibration as reference.

Figure 6.2 shows the errors for a naive calibration where all the measurement points were used.

Figure 6.3 shows the worst errors over the angles and Figure 6.4 shows the mean errors.

[1] see Appendix A.3 for definitions.

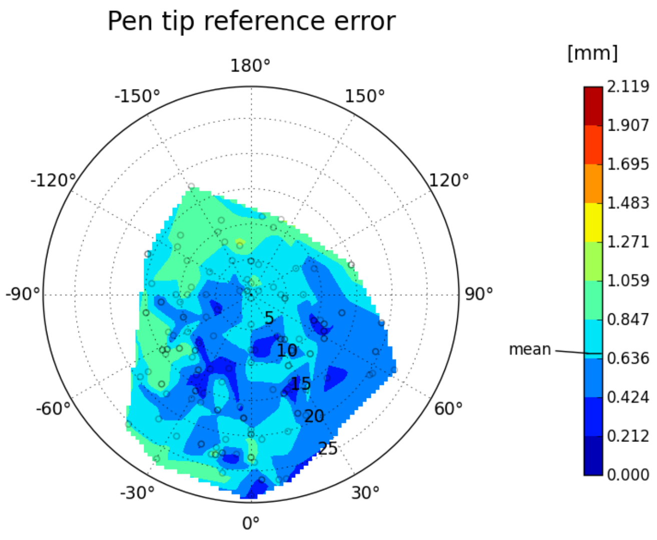


Figure 6.1 L_2-norm of the error of the pen-tip with respect to tilt and skew for a manually calibrated tool. These measurements were collected for 11 hours automatically by the IRB140 robot using Python.

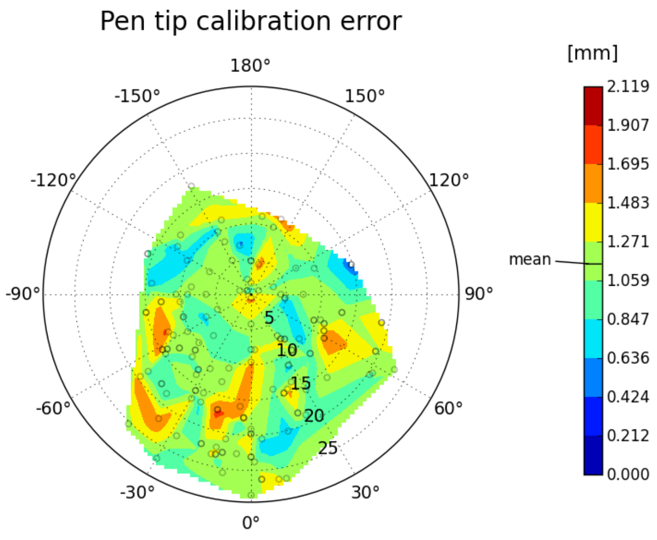


Figure 6.2 L_2-norm of the error of the pen-tip with respect to tilt and skew for the corresponding calibrated tool using the calibration algorithm.

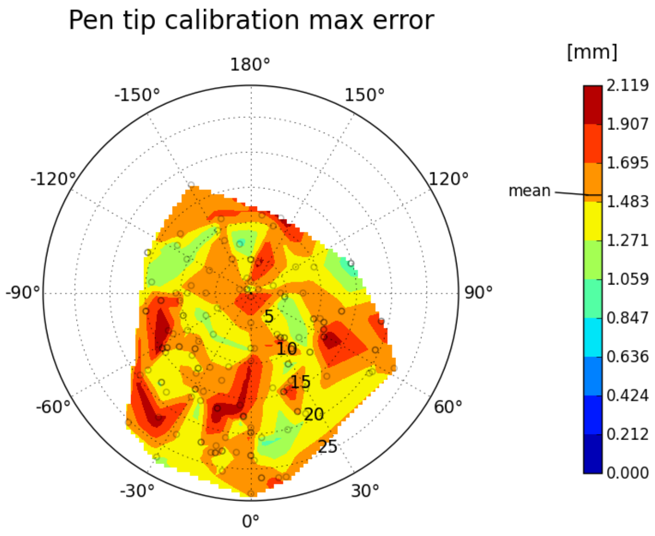


Figure 6.3 Maximum error of 100 different calibration runs where the calibrations were done with 10 different unique poses per run from the collected measurements.

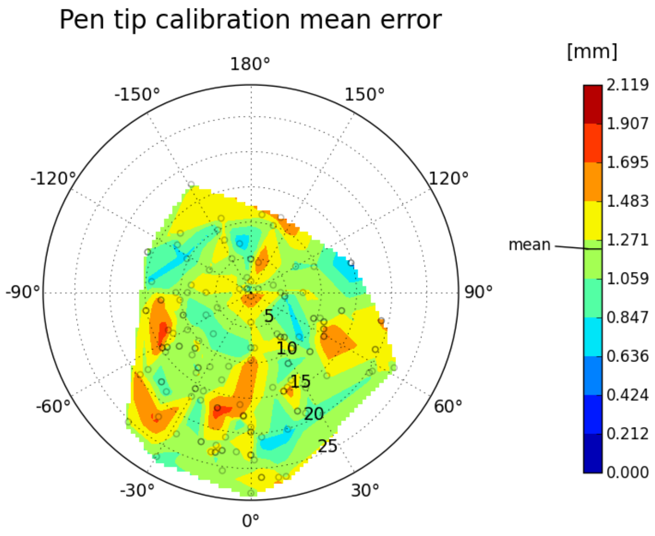


Figure 6.4 Mean error of 100 different calibration runs where the calibrations were done with 10 different unique poses per run from the collected measurements.

6.2 Calibration Algorithm Optimization

The following figures show the calibration algorithm performance on a smaller subset of 25 measurements where a greedy[2] optimization strategy is used to improve the result.

The optimization strategy we used involved performing the calibration algorithm, starting with 3 randomly selected measurement points, and then repeating the calibration after selecting another measurement point from the measurements, if this improves the calibration we keep it, otherwise we throw it away. This is iterated a number of times over the measurements, throwing away measurements until this strategy stops improving the calibration result.

Figure 6.5 shows the measured pen tip positions relative the first measured position in the reference work object coordinate system.

Figure 6.6 shows the result after optimization. As can be seen, the resulting pen tip positions form clusters that are far apart.

Figure 6.7 shows the maximum L_2-norm deviation error of the calibrated pen tip positions and how this changes with increasing number of measurements used in the calibration algorithm.

Figure 6.8 is the result after the greedy optimization where only 7 measurements are kept from the original 25. The figure shows how the calibration result improves for each measurement due to the optimization strategy being used. The results here can also be seen in Table 6.1 on page 55.

Figure 6.9 shows the calibrated work object relative orientation error to the reference work object in degrees of each basis vector.

As the calibration algorithm calibrates the tool, i.e., the pen tip position we also obtain a calibration of the work object surface orientation, in the robot base coordinate system, as an added result.

Figure 6.10 shows the result after optimization.

[2] A greedy optimization strategy follows the problem solving mind-set of making the locally optimal choice at each optimization iteration with the hope of finding a global optimum. A greedy strategy does not in general produce an optimal solution, but may find locally optimal solutions that approximate a global one in a reasonable time

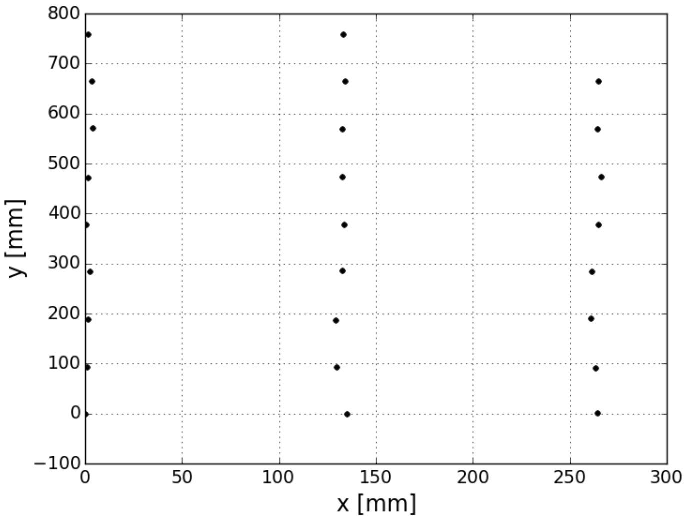


Figure 6.5 Pen coordinates in the work object coodinate system.

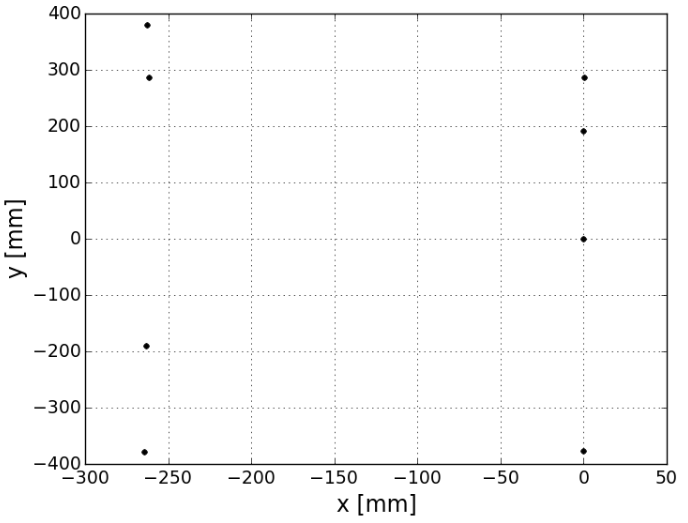


Figure 6.6 Pen coordinates after greedy optimization.

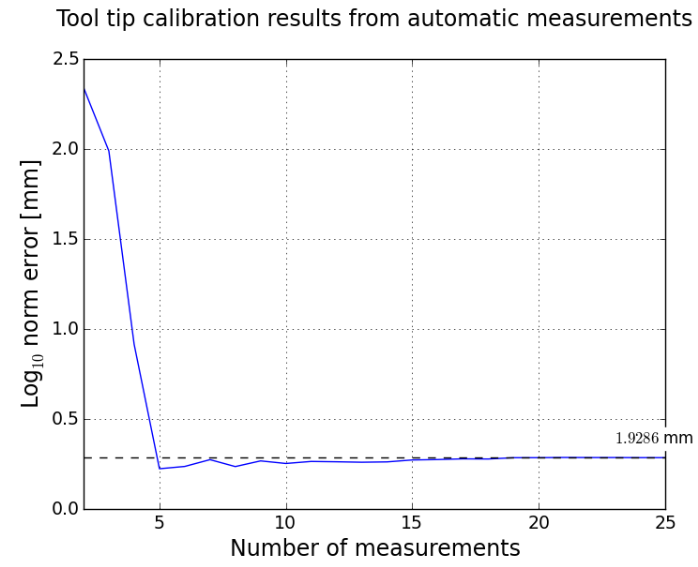


Figure 6.7 Tool tip max error to pen coordinates after calibration.

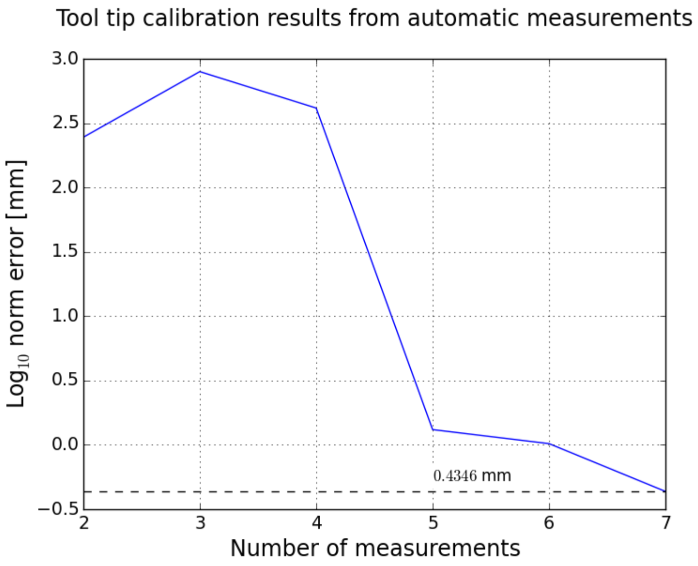


Figure 6.8 Tool tip max error to pen coordinates after greedy optimization.

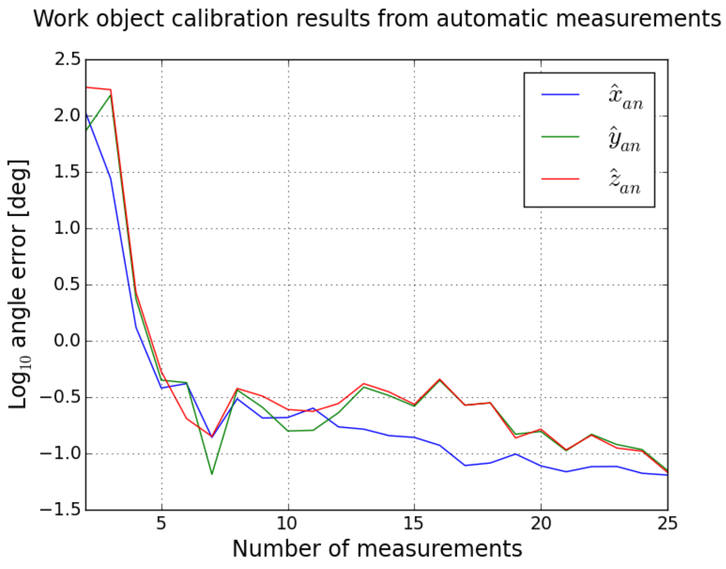


Figure 6.9 Error of basis vectors in degrees.

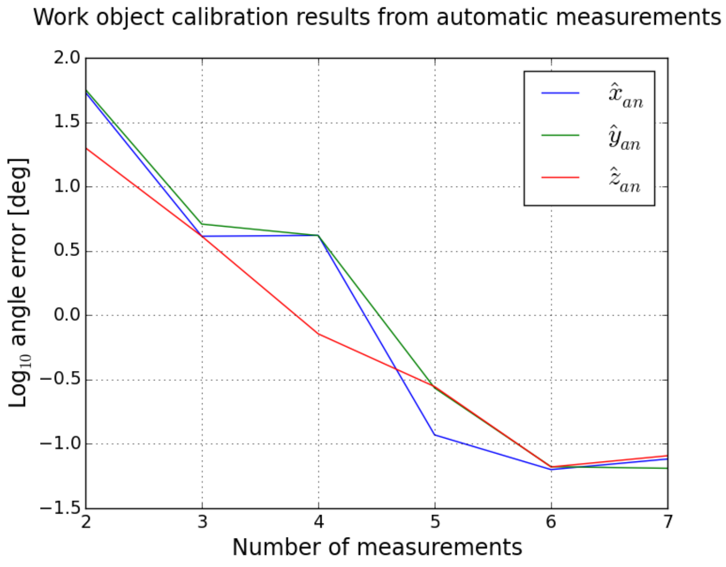


Figure 6.10 Error of basis vectors in degrees after greedy optimization.

6.3 Calibration Algorithm Precision Lower Bound

The Anoto digital pen does not report the pen tip position precisely, as can be seen in Figure 6.11.

Here we measure the error in the pen tip position by placing the pen on a piece of paper with printed Anoto pattern and rotating the paper and keeping the tip fixed in the same position, while increasing and decreasing the tilt angle of the pen relative to the paper.

The tilt angle was changed in the range of 0 and 50 degrees while not rotating the pen about its own symmetrical axis, it is skew angle was in other words 0. see Appendix A.3 for the rot, tilt and skew angle definitions of the pen orientation relative to the pattern surface.

The mean of the reported pen tip positions is the origin of rotation and is the exact physical pen tip position that the reported positions deviate from. We can see that the reported pen tip positions deviate from the real physical position by 0.3 - 0.4mm.

Neither is the IRB140 robot precise when it is placing the pen in different positions. The robot will introduce a small error relative to the expected pen tip placement which will affect the calibration results.

The calibration results from Figure 6.8 on page 52 can be seen in Table 6.1.

Observing the flange position deviation errors, here calculated with the L_2-norm, we can conclude that the flange position error is some hundredths of a millimetre in magnitude.

Repeated measurements of the flange error will reveal this to be consistent and comparing this with the robot datasheet, located in Appendix A.1, we an see that the position repeatability is tested to be on average 0.03mm.

Using this information we can expect a realistic lower bound on the precision of the calibration algorithm to be decided by the precision of the pen tip positions being reported.

The calibration algorithm results are therefore dependant on the pen being used, and in this case the pen tip calibration error can be no lower than 0.3 - 0.4mm.

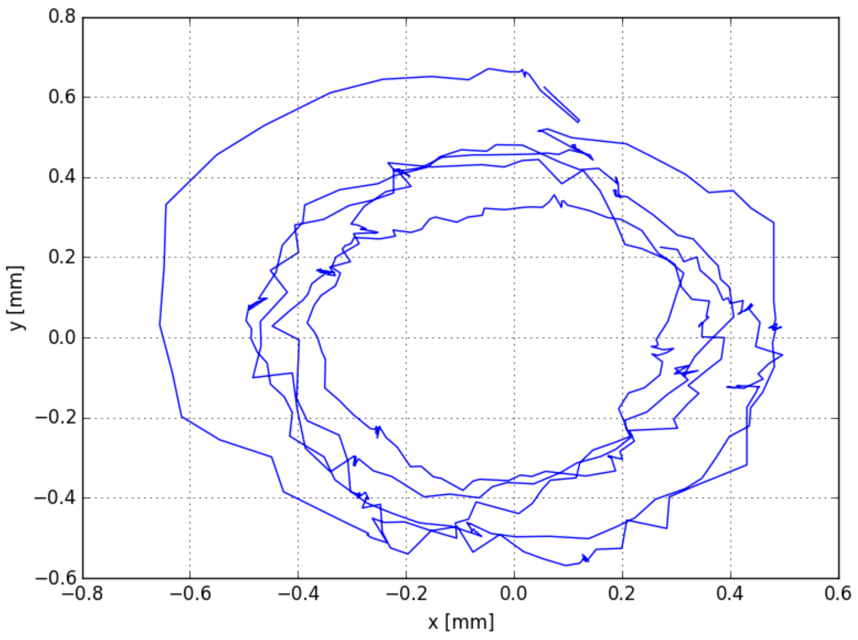


Figure 6.11 The measured pen tip position errors relative to the physical pen tip origin for pen tilt angles in the range of 0 to 50 degrees.

	tip^{error} [mm]	$\hat{x}_{an}^{error}$ [mm]	$\hat{y}_{an}^{error}$ [mm]	$\hat{z}_{an}^{error}$ [mm]	$flange^{error}$ [mm]
2	245.912655	53.960271	56.786330	19.991105	0.032864
3	794.959780	4.103650	5.102067	4.109336	0.002231
4	414.242329	4.169606	4.153502	0.716227	0.019773
5	1.313704	0.117373	0.271912	0.278748	0.010225
6	1.022351	0.063100	0.066487	0.066143	0.015158
7	0.434636	0.076218	0.064563	0.080832	0.034265

Table 6.1 Calibration errors and flange errors related to Figure 6.8 on page 52.

6.4 Calibration Algorithm Simulation

Here we try to evaluate the performance of the calibration algorithm using simulated data.

The expected tool positions and orientations were generated on a plane, and with the help of the forward and inverse kinematics implementations we could create the geometry information needed to generate simulated calibration results with the calibration algorithm.

A noise model of two components was used to simulate the pen tip position reporting errors.

The first component was an even circular noise distribution with a radius of up to 0.3mm, and the second component was a linear noise deviation, where the deviation direction depended on the orientation of the pen and the magnitude on the tilt of the pen.

The idea was to have a robust model that would mimic the real world pen behaviour, but also stress-test the algorithm.

We're essentially measuring the results of a "worst-case" behaviour of a pen that needs to be calibrated.

Here we also assume that the robot flange position errors are included in this noise model, since their errors have a magnitude of 0.03mm they should not affect the outcome significantly.

In the figures we have chosen an ideal tip error of 0.1mm as tolerance reference.

Figure 6.12 is an overview and Figure 6.13 is a close-up of the result.

We can see that after 325 measurements the maximum error is calibrated below 0.1mm, while the mean error quickly drops below the tolerance and stabilizes between a tip error of 0.1 and 0.01mm.

In the close-up in the range of 0 to 100 measurements and we can clearly see that after 35 measurements there's a good chance to obtain a good calibration.

The work object orientation calibration in Figure 6.14 shows a very pessimistic result where the tolerance chosen represents an angle max error of 0.1 degrees.

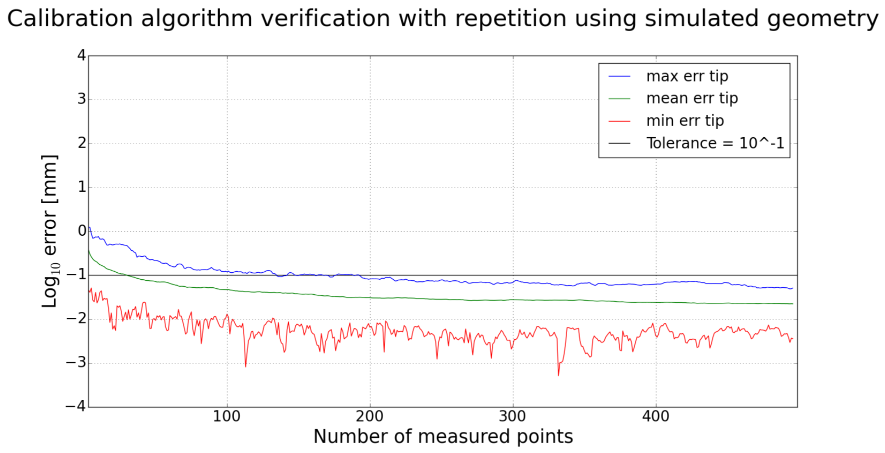


Figure 6.12 Pen-tip verification model using rectangular noise perturbation of pen-tip coordinates with a radius of 0.3mm, and an extra added noise component with 0.01mm radius and linear dependence on tilt. Point Spread: 200mm; Rot: [-180,180], Tilt: [0, 40], Skew: [-90, 270].

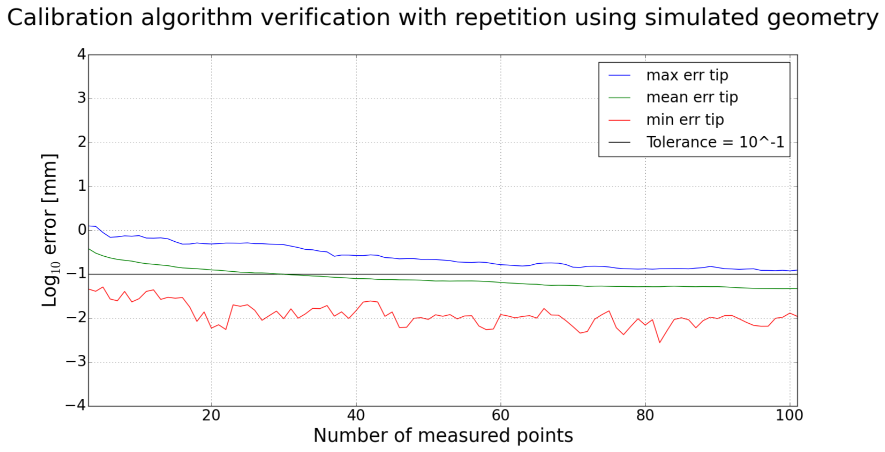


Figure 6.13 Pen-tip verification model using rectangular noise perturbation of pen-tip coordinates with a radius of 0.3mm, and an extra added noise component with 0.01mm radius and linear dependence on tilt. Point Spread: 200mm; Rot: [-180,180], Tilt: [0, 40], Skew: [-90, 270].

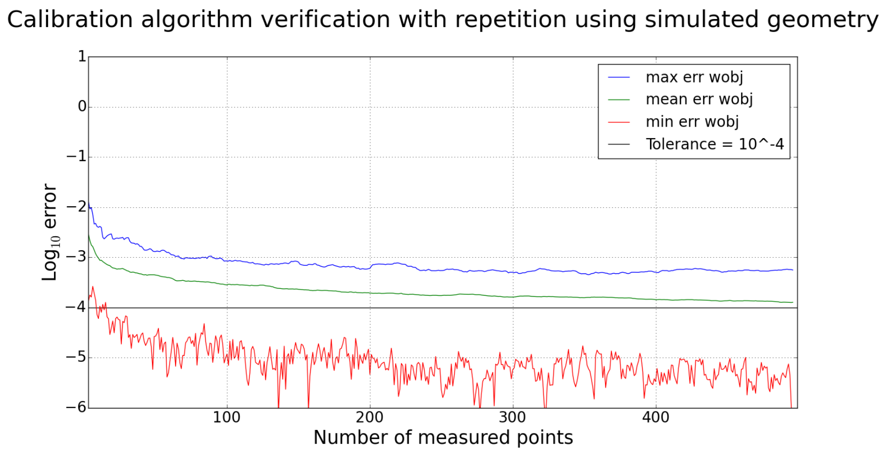


Figure 6.14 Work object verification model using rectangular noise perturbation of pen-tip coordinates with a radius of 0.3mm, and an extra added noise component with 0.01mm radius and linear dependence on tilt. Point Spread: 200mm; Rot: [-180,180], Tilt: [0, 40], Skew: [-90, 270].

6.5 Calibration Algorithm Orientation Estimation

We present here a verification of the tool orientation calibration algorithm. We only verified this by simulations, since it was too difficult and time consuming to perform the same verification with measurements in the Anoto robotics lab with the resources available.

Similar like before we set up a virtual geometry, we generate a set of pairs of tool positions and orientations with their corresponding flange positions and orientations. We also know the relative rotation transformation that transforms from the flange orientation to the tool orientation and use this as reference.

A simulated measurement noise with a uniform distribution and magnitude of $\pm 2 \cdot 10^{-3}$ [mm] was added to each element of the 3×3 reference matrix, since this was the error at the time with which the prototype pen was estimating its current orientation.

Figures 6.15, 6.16 and 6.17 illustrate the maximum, mean and minimum errors in degrees for each basis vector of the estimated orientation of the tool, which are plotted against the number of measurements used in the calibration of the orientation of the tool, i.e., the Anoto pen.

It is the author's impression that the mean-value results at 25 measurements reflects the behaviour of the Anoto pens, from experience of working with the pens.

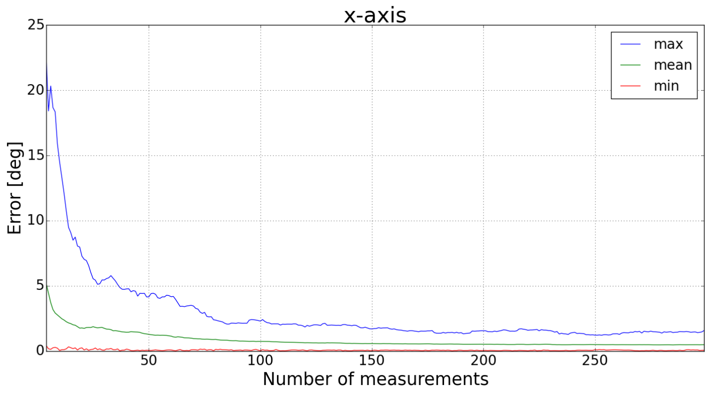


Figure 6.15 Tool orientation verification model. A calibration is performed from measurements of 4 to 300 tip positions of different orientations with a maximum tilt of 40 degrees for a tool tip of length 233mm. This is simulated 100 times to calculate maximum and minimum span of the orientation error.

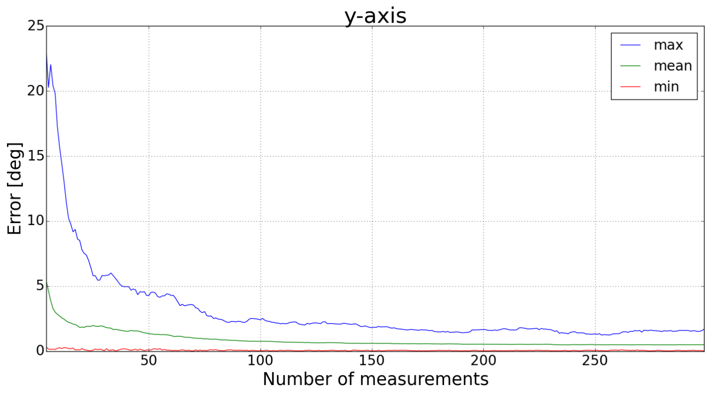


Figure 6.16 Tool orientation verification model. A calibration is performed from measurements of 4 to 300 tip positions of different orientations with a maximum tilt of 40 degrees for a tool tip of length 233mm. This is simulated 100 times to calculate maximum and minimum span of the orientation error.

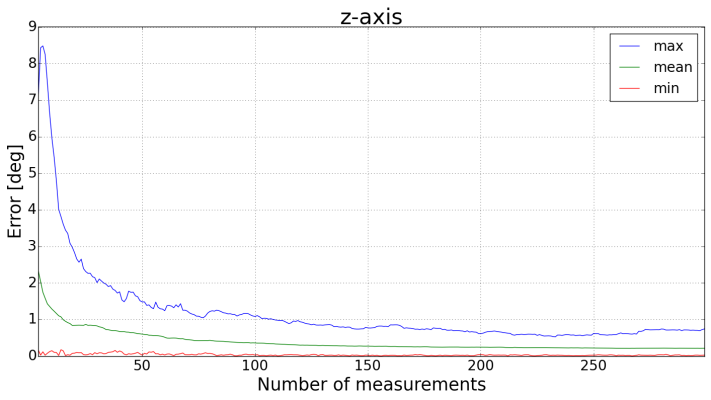


Figure 6.17 Tool orientation verification model. A calibration is performed from measurements of 4 to 300 tip positions of different orientations with a maximum tilt of 40 degrees for a tool tip of length 233mm. This is simulated 100 times to calculate maximum and minimum span of the orientation error.

6.6 Inverse Kinematics Over a Curve

Here we present an application of the forward and inverse kinematics of the robot to find a smooth robot motion over a curve.

Figure 6.18 shows the robot geometry presented as colourful links with the different frames attached on each joint as a set of red, green and blue basis vectors[3]. The tool is presented as a thick black line and the curve being traversed is a gray circle.

A close up of the curve and the tool can be seen in Figure 6.19 where the orientation of the tool is located as an extra large coordinate frame on the tip of the tool.

A set of smaller frames with the same orientations are placed at even locations on the curve. In this set-up, the robot is moving the tip of the tool, e.g., a pen tip over each location of the curve and keeping the pen orientation constant.

Figures 6.20 to 6.25 show the motion of each joint angle over time (blue) and the corresponding joint angle velocities (red) over the entire curve, which is traversed in 12.56 seconds. We will describe the motion of the robot that these figures represent for the first half of the curve, the reader can then work out the rest of the motion for the other half of the curve.

[3] Red for the x-axis, green for the y-axis and blue for the z-axis of the frame.

The total motion for the first half of the curve is a robot arm that is at first tilted backwards and keeps its elbow squeezed together to reach the point of the curve closest to its base, see Figures 6.21 and 6.22. The robot then follows the curve with the tool clockwise, see Figure 6.20, while extending its elbow-joints.

The last two joints of the robot is working to keep the tool orientation constant by counter-acting the change in orientation caused by the earlier joints, see Figures 6.24 and 6.25.

In Figure 6.18 the robot has reached the first quarter of the curve, right after the 2.512 seconds mark, while keeping the tool straight and not changing its orientation during the entire motion up to that point.

The determinant of the Jacobian of the robot can be seen in Figure 6.26 where we can see, by the magnitude of the values over time, that the robot is moving away from a critical joint configuration, since the determinant value is moving away from zero.

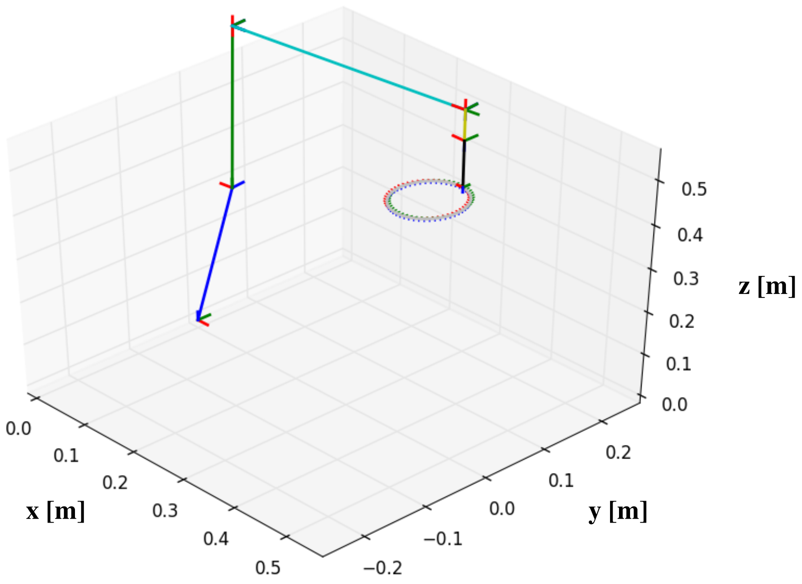


Figure 6.18 Robot joint links and joining frames, tool and trajectory. The robot is moving the tool in clock-wise direction and has traversed 1/4th of the curve, starting from the point closest to the robot base.

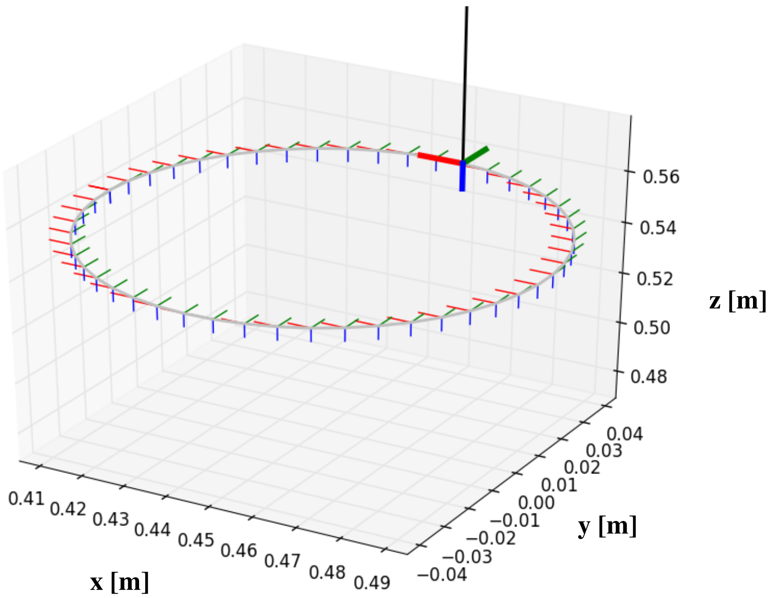


Figure 6.19 Close-up of tool (black) and trajectory with tool frames. The colors represents the frame coordinate axes x (red), y (green) and z (blue).

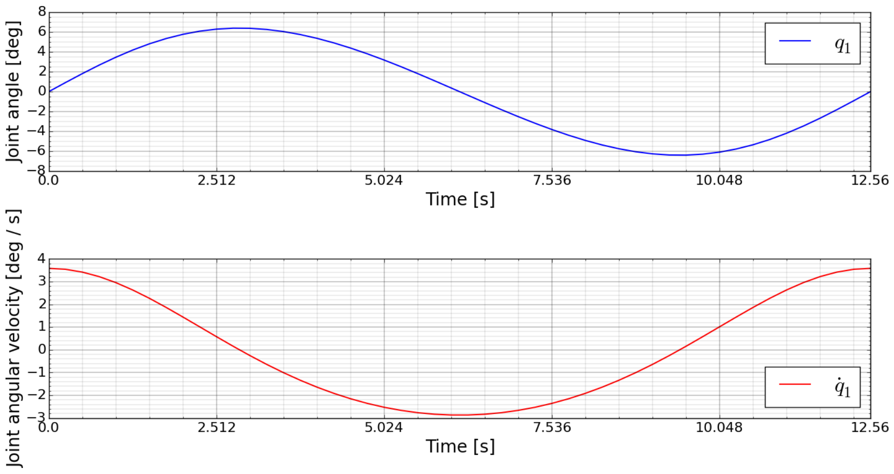


Figure 6.20 Change of joint angle over time (blue) and change of joint angular velocity over time (red) of the first link.

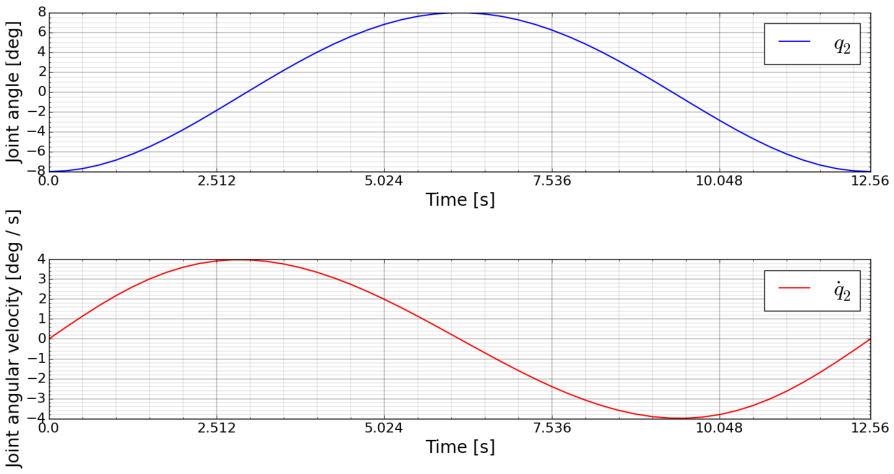


Figure 6.21 Change of joint angle over time (blue) and change of joint angular velocity over time (red) of the second link.

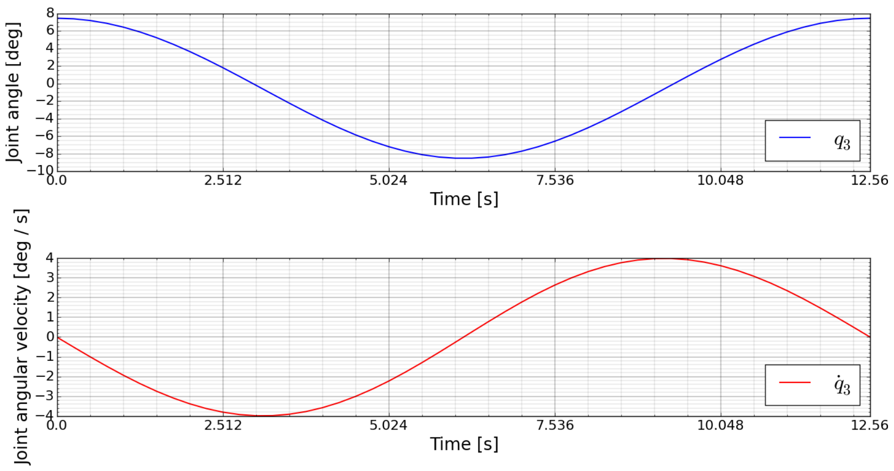


Figure 6.22 Change of joint angle over time (blue) and change of joint angular velocity over time (red) of the third link.

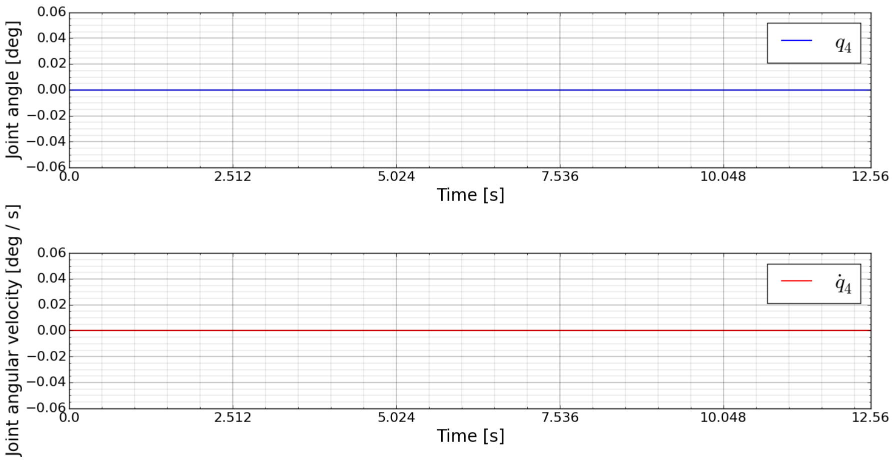


Figure 6.23 Change of joint angle over time (blue) and change of joint angular velocity over time (red) of the fourth link.

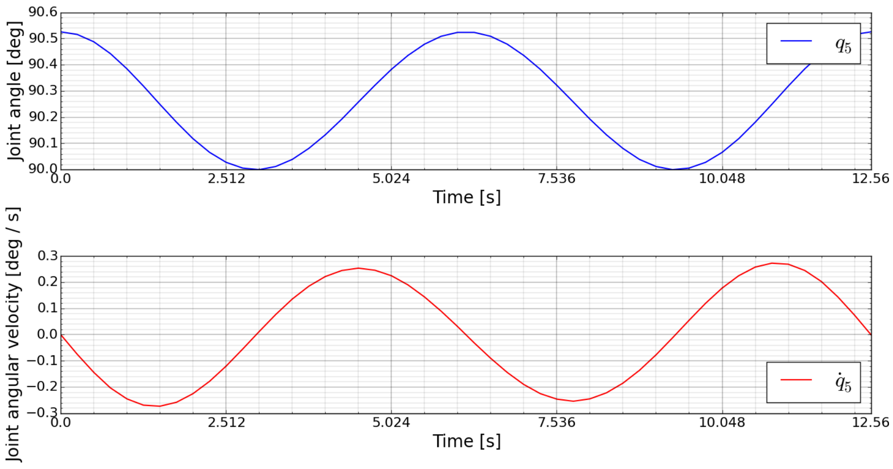


Figure 6.24 Change of joint angle over time (blue) and change of joint angular velocity over time (red) of the fifth link.

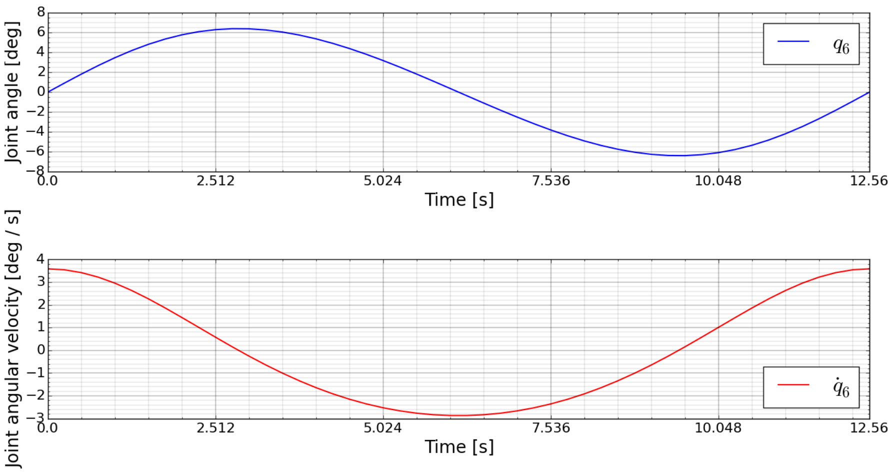


Figure 6.25 Change of joint angle over time (blue) and change of joint angular velocity over time (red) of the sixth link.

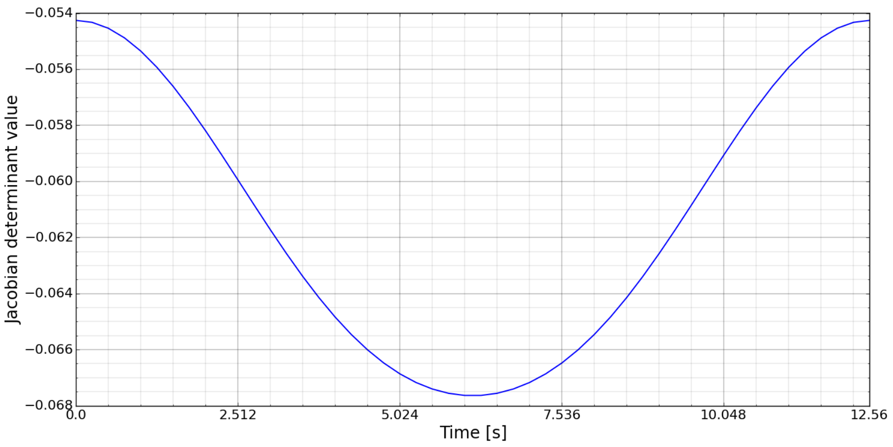


Figure 6.26 Change of the determinant of the Jacobian over time.

6.7 HID Parsing

During this master thesis the author developed a framework for controlling the robot using the streaming Bluetooth data from an Anoto pen. In this section we interpret and visualize the streamed data.

In Figure 6.27 the author has written his name, where the origin is the starting point of the cursive writing. In the pattern coordinate system the y-axis would normally range from negative values (upper left) to positive values (bottom left). This felt confusing since when we're plotting general curves it is usually done the the other way around, so this has been adjusted for readability.

Figure 6.28 shows the raw unfiltered data, that we are receiving from the pen, in the Anoto pattern coordiante system in Anoto distances.

The horizontal and vertical red lines are spikes that are being misinterpreted. This was revealed after some debugging to be caused by the Python module [Rusnak, 2015] used for handling the Bluetooth communication.

Figure 6.29 shows the same data in the pattern coordinate system in centimeters and with the final filtered result shown in blue from Figure 6.27.

The final result was obtained by a simple threshold limit filter on the input data.

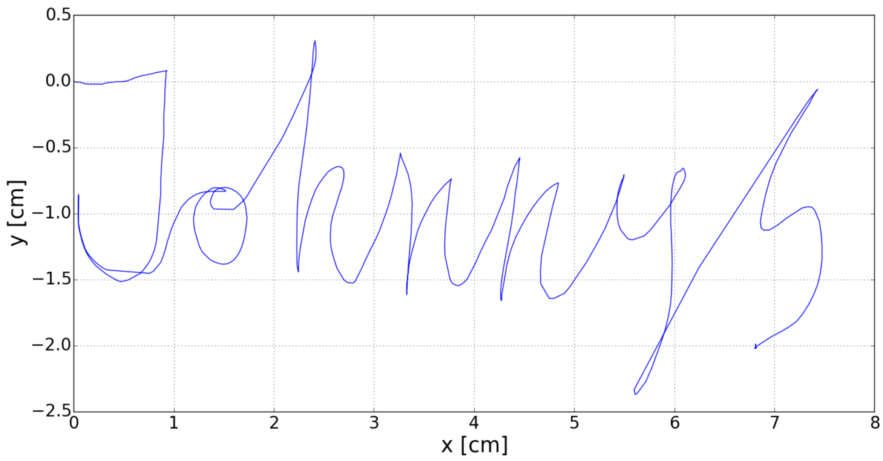


Figure 6.27 Parsed and filtered Bluetooth data from an Anoto pen of the author's name.

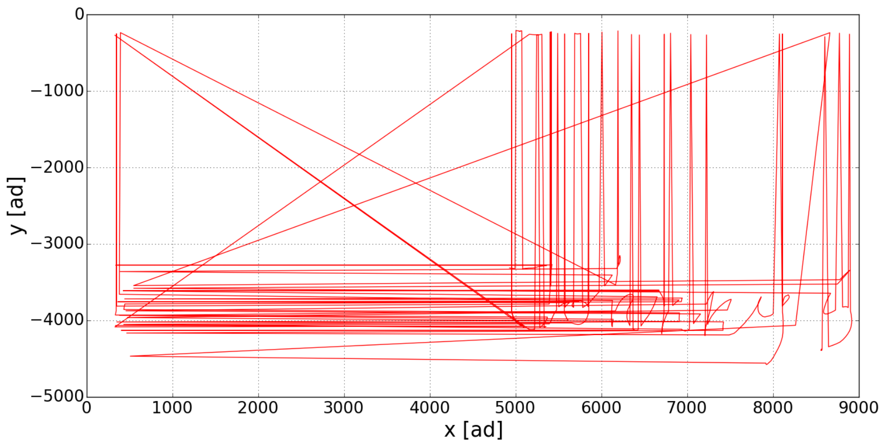


Figure 6.28 The filtered errors from the Bluetooth data.

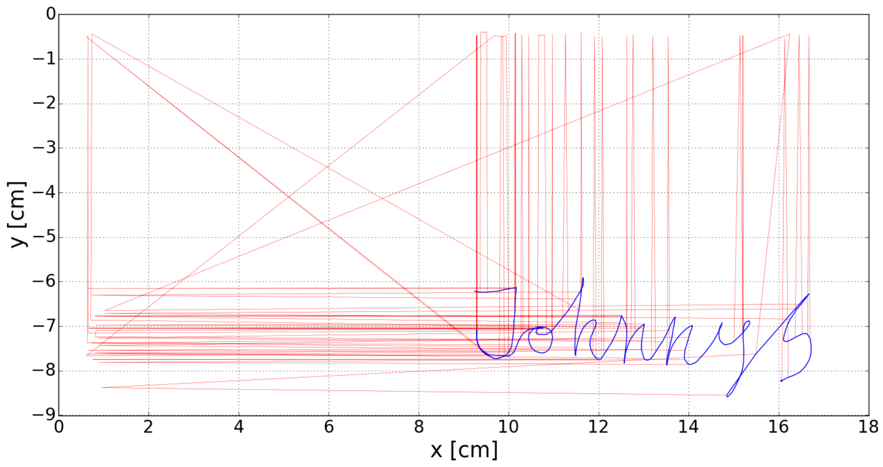


Figure 6.29 Parsed Bluetooth data from Anoto pen including errors and filtered signature.

7 Discussion

Work done in the field of hand-eye calibration and robot-world calibration show that separating the orientations from the translations will worsen the quality of the calibrations.

Different methods using quaternions, screws, dual quaternions and non-linear minimization show comparable results with difference in operation times and implementation difficulties. This problem has shown to be less intuitive than expected and has also shown the importance of not jumping to conclusion: when writing down the equations it is a quick jump to treat the orientations and translations problems as non-connected, but previous research in this field has proven that important informations is lost.

The Anoto pens used in this master thesis measures relative orientations with large noise components in tilt and skew and was for this reason discarded in the problem formulation and why only a theoretical verification using a simulation model is done. The results suggest that this might have been a hasty decision and that the noisy orientation measurements could be usable after all since the SVD method of fitting the orientation has shown to be robust and handle noise very well.

The work object world origin was left unsolved in this master thesis due to time constraint: many different solution strategies could be used and no time was left to explore different strategies.

Initial formulation of the calibration algorithm, that included the origin, gave bad results due to the origin became very large compared to the relative distance of the measurement points. This resulted in ill-conditioned systems of equations and the algorithm was reformulated to exclude the origin for these reasons.

The robot was set to a very slow speed during measurements to guarantee that no response delay would end up accidentally crushing the pens used in this master thesis. For this reason one measurement would take 2-3 minutes to perform and that's counting the time it takes for the robot to change position and orientation. One measurement run of 240 measurements took 10 hours.

From the results this is not a problem and repeated tests have shown that 10-16 measurements are enough if care is taken to how the measurements are performed.

The results in this master thesis are from measurements in an ordered grid with symmetrical angles[1] which shows that this is not enough to obtain a good calibration fast.

The research papers mention that the joint rotational axes must be far from parallel to each other to get good calibration results and that a small span will noticeably worsen the calibration results. Care was taken for this situation by taking large joint angle spans, but this does not guarantee non-parallel joint axes and can definitely contribute to why the calibration results were not better.

[1] As close to symmerical angles as can be expected from a pen taped on to a pen-holder which was not constructed for the pens used in this master thesis.

8
Conclusion

This master thesis has shown multiple uses of the Anoto dot pattern and Anoto pen for different applications.

One application is the calibration of the robot tool and work object, when an Anoto pen has been mounted on the robot and the work object consists of a surface with the Anoto dot pattern printed onto it.

Another is the use of an Anoto pen and paper as replacement of the flexpendant to create a light-weight and intuitive control device to control the IRB140 robot arm. The Python programming language and interpreter were used together with ABB's RAPID programming language to create a framework to handle the communication between the user and the robot. In that application it was demonstrated how this could be done via the use of serial communication.

The inverse and forward kinematics were derived for the robot and used for simulations and for determining if a trajectory could be traversed. This helped in the theoretical verification in the development of the calibration algorithm, and was also used for determining if a set of measurements could be performed at the Anoto robot lab for a given curve, work object and pen set-up.

The calibration details from the results section show that a naive approach to calibrating the tool tip, i.e., using all of the measurement values results in a deviation of 1mm or more with slow convergence compared to the manual calibration of 0.5mm deviation.

By using a greedy optimization strategy of successively adding measurements that improve the calibration, and removing measurements that worsen it, we get a calibration comparable to a manual calibration with fast convergence.

The work object calibration, i.e., the orientation estimation of the Anoto pattern surface shows promising results with few measurements and faster convergence, but can also be improved with the same optimization strategy.

The simulated results for the pen tip calibration show slow convergence and that a good calibration can be obtained after 35 measurements. The work object calibrations results were pessimistic but this is very likely due to the simulation was designed to mimic a worst case scenario of calibrating a poor pen, so in such a case the work object results should should be expected to be bad and that the work object calibration is more sensitive to the quality of the pen measurements.

The tool orientation estimation shows more promising theoretical results when taken into account the very pessimistic noise model in the reported pen orientation, with matrix element errors of $\pm 2 \cdot 10^{-3}$mm.

By performing manual measurements, and using measurements of the error in the robot flange position, a realistic lower bound on the precision of the calibration algorithm was decided to be no lower than 0.3 - 0.4mm, and depended on the performance of the pen.

The illustrated control board was developed through a user study with volunteers from Anoto. The resulting design felt intuitive to use and the user feedback was positive.

The robot had a delay of ≈ 0.25s[1] which gave a sluggish impression when controlling the robot. The movement length was controlled by the distance to the control board origin and helped in minimizing the "sluggish" feeling.

Replacing the flexpendant with Anoto technology or a similar set-up is viable.

[1] This is the author's estimation from working with the robot—this value is a subjective estimation and not obtained from measurements.

9 Future Work

This paper mainly focuses on the calibration algorithm and on the calibration of the tool tip. It does not take into account the environment and any possible obstacles within it.

An initial guess of the position and orientation of the tool tip is required to steer the tip of the pen to touch the surface with the Anoto pattern without breaking the pen.

This means that the pen must be mounted by the user in some rigid fashion.

The calibration process has a manual set-up stage, and the calibration area must be free of obstacles which makes the calibration process not entirely autonomous.

The techniques used by robotics researchers to detect obstacles [Harada et al., 2014] in the environment and to estimate the position and orientation of the tool to pick up are possible candidates for improvement of the current calibration process developed in this paper.

The calibration process can be improved by solving for both the positions and orientations together, instead of treating them as separate problems[1].

Screw theory and dual quaternions are the more straightforward methods that produce good results compared to the non-linear methods found so far.

This does not guarantee a better calibration due to the noisy orientation data from the pens, but it is worth investigating.

The robot speed should be sped up to shorten the calibration time. The most consuming part of this process is the time it takes for the robot to perform a single measurement.

[1] This means to also use the noisy orientation data from the Anoto pens.

I suggest that the time is shorten to at least 15-30s per measurement.

Instead of a greedy optimization algorithm a RANSAC[2] approach could be used to speed up the optimization step of the calibration.

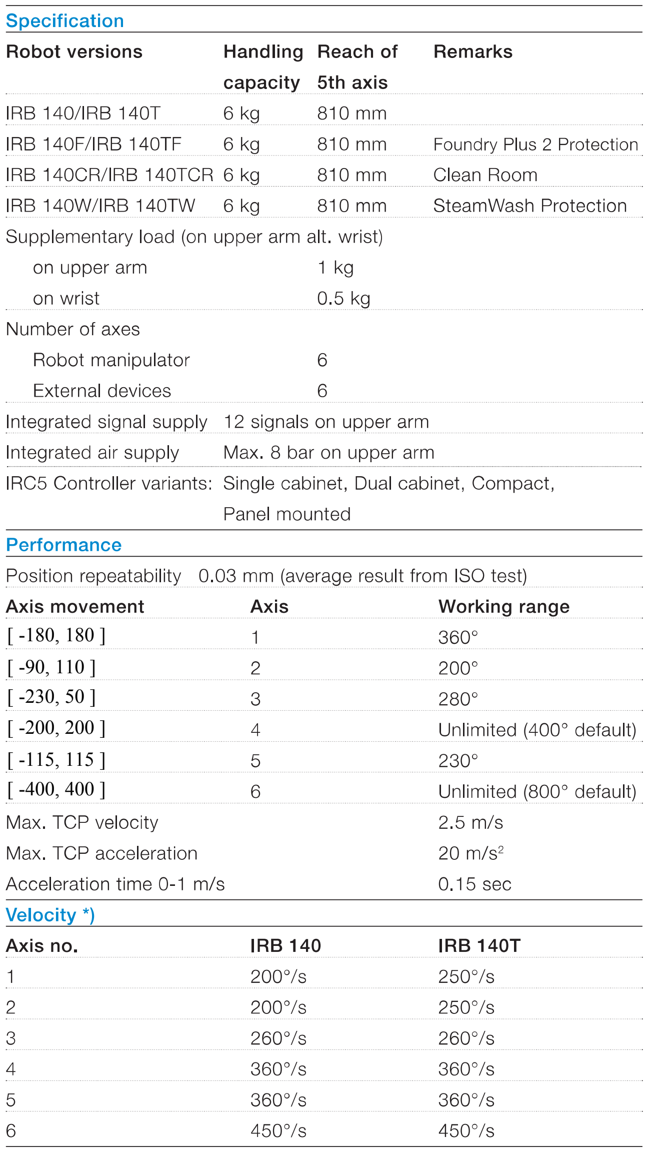
The forward and inverse kinematics are known and presented in this master thesis and could be use to guarantee that the joint axes are far from parallel to each other for each measurement.

[2] Abbreviation of Random Sample Consensus.

A

Appendix

A.1 Technical Specification Robot ABB IRB140-6/0.8



Specification

Robot versions	Handling capacity	Reach of 5th axis	Remarks
IRB 140/IRB 140T	6 kg	810 mm	
IRB 140F/IRB 140TF	6 kg	810 mm	Foundry Plus 2 Protection
IRB 140CR/IRB 140TCR	6 kg	810 mm	Clean Room
IRB 140W/IRB 140TW	6 kg	810 mm	SteamWash Protection
Supplementary load (on upper arm alt. wrist)			
on upper arm		1 kg	
on wrist		0.5 kg	
Number of axes			
Robot manipulator		6	
External devices		6	
Integrated signal supply	12 signals on upper arm		
Integrated air supply	Max. 8 bar on upper arm		
IRC5 Controller variants:	Single cabinet, Dual cabinet, Compact, Panel mounted		

Performance

Position repeatability 0.03 mm (average result from ISO test)

Axis movement	Axis	Working range
[-180, 180]	1	360°
[-90, 110]	2	200°
[-230, 50]	3	280°
[-200, 200]	4	Unlimited (400° default)
[-115, 115]	5	230°
[-400, 400]	6	Unlimited (800° default)
Max. TCP velocity		2.5 m/s
Max. TCP acceleration		20 m/s^2
Acceleration time 0-1 m/s		0.15 sec

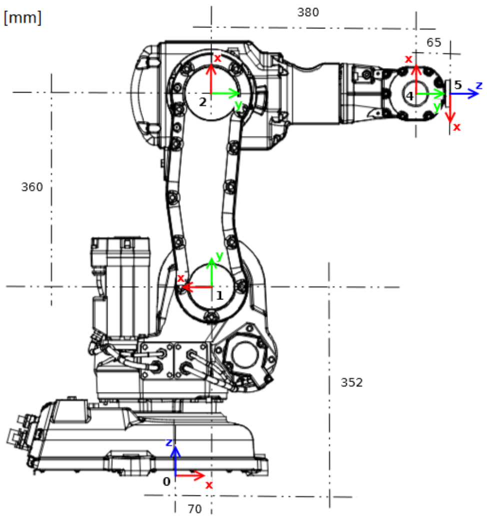
Velocity *)

Axis no.	IRB 140	IRB 140T
1	200°/s	250°/s
2	200°/s	250°/s
3	260°/s	260°/s
4	360°/s	360°/s
5	360°/s	360°/s
6	450°/s	450°/s

A.1 Technical specifications of the IRB140 model from ABB.

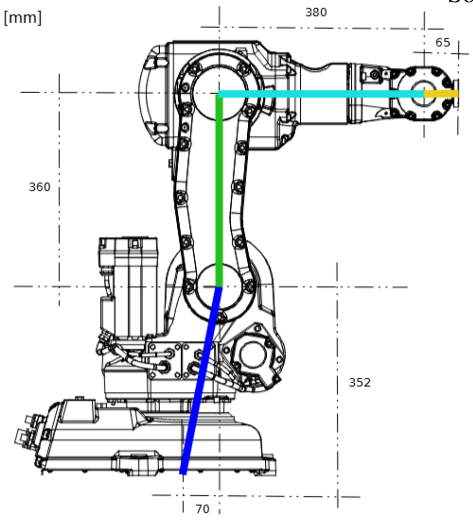
Source: ABB.

A.2 Kinematic Chain



A.2 Assigned frames to the joints of the IRB140 in base-frame coordinates. Frame 3 has been omitted for clarity.

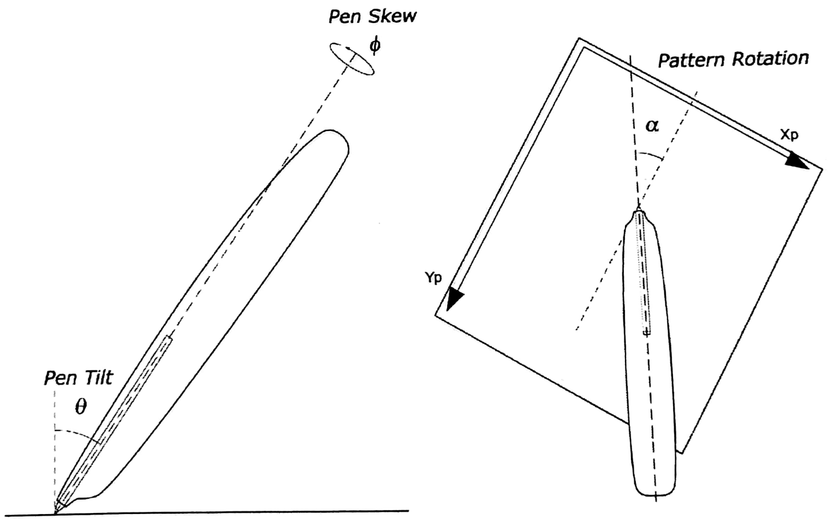
Source: Source: ABB.



A.3 The manipulator links that form the skeleton of the forward kinematics chain. These are displayed when plotting the robot configurations.

Source: Source: ABB.

A.3 Pen Orientation Definitions



A.4 Definition of the angles rot, tilt and skew, relative to the pattern coordinate system, used to describe the orientation of the pen relative to the dot-pattern. Image supplied by Anoto.

Source: Anoto AB.

A.4 Anoto Pen HID Descriptor

```
INPUT(9)[INPUT]
  Field(0)
    Physical(Digitizers.Stylus)
    Application(Digitizers.Pen)
    Usage(5)
      Digitizers.TipSwitch
      Digitizers.BarrelSwitch
      Digitizers.Eraser
      Digitizers.Invert
      Digitizers.InRange
    Logical Minimum(0)
    Logical Maximum(1)
    Report Size(1)
    Report Count(5)
    Report Offset(0)
    Flags( Variable Absolute )
  Field(1)
    Physical(Digitizers.Stylus)
    Application(Digitizers.Pen)
    Usage(1)
      GenericDesktop.X
    Logical Minimum(0)
    Logical Maximum(8816)
    Physical Minimum(0)
    Physical Maximum(165)
    Unit Exponent(-1)
    Unit(SI Linear : Centimeter)
    Report Size(32)
    Report Count(1)
    Report Offset(8)
    Flags( Variable Absolute )
  Field(2)
    Physical(Digitizers.Stylus)
    Application(Digitizers.Pen)
    Usage(1)
      GenericDesktop.Y
    Logical Minimum(0)
    Logical Maximum(7126)
    Physical Minimum(0)
    Physical Maximum(133)
    Unit Exponent(-1)
    Unit(SI Linear : Centimeter)
    Report Size(32)
    Report Count(1)
    Report Offset(40)
    Flags( Variable Absolute )
  Field(3)
    Physical(Digitizers.Stylus)
    Application(Digitizers.Pen)
    Usage(1)
      Digitizers.TipPressure
      Logical Minimum(0)
      Logical Maximum(2047)
      Physical Minimum(0)
      Physical Maximum(2047)
      Unit Exponent(-1)
      Unit(SI Linear : Centimeter)
      Report Size(16)
      Report Count(1)
      Report Offset(72)
      Flags( Variable Absolute )
    Field(4)
      Physical(Digitizers.Stylus)
      Application(Digitizers.Pen)
      Usage(1)
        Digitizers.003d
      Logical Minimum(-9000)
      Logical Maximum(9000)
      Physical Minimum(-9000)
      Physical Maximum(9000)
      Unit Exponent(-2)
      Unit(English Rotation : Degrees)
      Report Size(16)
      Report Count(1)
      Report Offset(88)
      Flags( Variable Absolute )
    Field(5)
      Physical(Digitizers.Stylus)
      Application(Digitizers.Pen)
      Usage(1)
        Digitizers.003e
      Logical Minimum(-9000)
      Logical Maximum(9000)
      Physical Minimum(-9000)
      Physical Maximum(9000)
      Unit Exponent(-2)
      Unit(English Rotation : Degrees)
      Report Size(16)
      Report Count(1)
      Report Offset(104)
      Flags( Variable Absolute )
    Field(6)
      Physical(Digitizers.Stylus)
      Application(Digitizers.Pen)
      Usage(1)
        Digitizers.0041
      Logical Minimum(0)
      Logical Maximum(36000)
      Physical Minimum(0)
      Physical Maximum(36000)
      Unit Exponent(-2)
      Unit(English Rotation : Degrees)
      Report Size(32)
      Report Count(1)
      Report Offset(120)
      Flags( Variable Absolute )

Digitizers.TipSwitch ---> Key.Touch
Digitizers.BarrelSwitch ---> Key.Stylus
Digitizers.Eraser ---> Key.Btn0
Digitizers.Invert ---> Key.ToolRubber
Digitizers.InRange ---> Key.ToolPen
GenericDesktop.X ---> Absolute.X
GenericDesktop.Y ---> Absolute.Y
Digitizers.TipPressure ---> Absolute.Pressure
Digitizers.003d ---> Absolute.XTilt
Digitizers.003e ---> Absolute.YTilt
Digitizers.0041 ---> Absolute.Misc
```

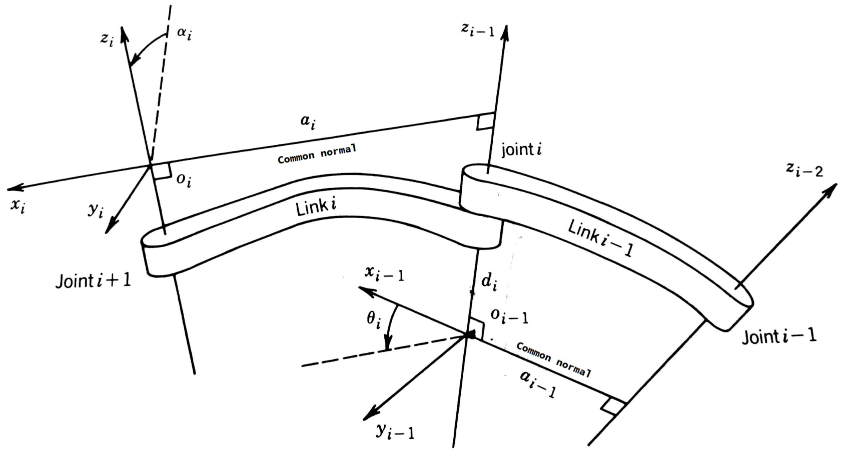
Listing A.1 Anoto pen HID descriptor for Bluetooth. (Parsed with Linux/Ubuntu)

A.5 Denavit-Hartenberg Parameters

The following is summarized[1] from [Spong and Vidyasagar, 1989] and [Craig, 1986].

It is possible to give physical interpretations on the parameters **length** a_i, **twist** α_i, **offset** d_i and **angle** θ_i for the homogeneous mapping $\underline{\mathbf{A}}_i$ of link $\{i\}$ and joint $\{i-1\}$.

The **length**[2] is the distance between the axes z_{i-1} and z_i and is measured along the axis x_i. The **twist**[3] is the angle between the axes z_{i-1} and z_i, measured in a plane normal to x_i. The positive direction for the twist is determined by the right-hand rule from z_{i-1} to z_i. The **offset**[4] is the distance between the origin o_{i-1} and the intersection of the x_i and z_{i-1} axes measured along the z_{i-1} axis. The **angle**[5] is the angle between the axes x_{i-1} and x_i measured in a plane normal to the z_{i-1} axis, see Figure A.5.



A.5 Illustration [Spong and Vidyasagar, 1989] of DH parameters for Eq. (5.2).

Source: Mark W. Spong and M. Vidyasagar, Robot Dynamics and Control.

[1] Note that Craig uses the modified DH-convention and Spong the standard DH-convention.
[2] $trans_{x_i}(a_i)$: translation a_i on axis x_i.
[3] $rot_{x_i}(\alpha_i)$: rotation α_i on axis x_i
[4] $trans_{z_{i-1}}(d_i)$: translation d_i on axis z_{i-1}
[5] $rot_{z_{i-1}}(\theta_i)$: rotation θ_i on axis z_{i-1}

A.6 Denavit-Hartenberg Convention

The following is summarized from[6] [Spong and Vidyasagar, 1989] and [Craig, 1986].

Revolute joint: the axis z_i for a rotational joint points in the axis of rotation.

Prismatic joint: the axis z_i for a translational-joint points in the axis of translation.

1. Locate and label the joint axes x_0 ... z_{n-1}.
2. Establish the base frame and set the origin anywhere on the z_0-axis.
   Choose x_0, y_0 so that we obtain a right-handed frame.

   For i = 1, ..., $n-1$, perform steps (a) to (c).

   a) Locate the origin o_i where the common normal to z_i and z_{i-1} intersects z_i.
      - If z_i intersects z_{i-1} locate o_i at this intersection.
      - If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

   b) Establish x_i along the common normal between z_{i-1} and z_i, or in the direction normal to the $z_{i-1}z_i$ plane if z_{i-1} and z_i intersect.

   c) Establist y_i to complete a right-hand frame.
3. Establish the end-effector frame $o_nx_ny_nz_n$.

   a) Assume the n_{th} joint is a revolute joint, we set z_n along the direction z_{n-1}

   b) Establish the origin o_n along z_n preferably at the tip of a tool. (or the center of a gripper)

      If the tool is a gripper:
      - Set y_n in the direction of the gripper closure and set $x_n = y_n \times z_n$

      Otherwise:
      - Set x_n and y_n conveniently to form a right-hand frame.

[6] Note that Craig uses the modified DH-convention and Spong the standard DH-convention.

A.7 Python Framework Implementation

```
PROC server()
    VAR iodev comport;
    VAR byte buffer{80};
    VAR string text;
    VAR num len;
    VAR string cmd_string;

    TPWrite "InitArap";
    ArapInit;

    ! Test the com port in read write mode
    Open "com2:", comport \Bin;

    WHILE continue DO
      cmd_string := ReadStrBin2(comport \Time:=300000);
      TPWrite "Recv = " + cmd_string;
      ArapDoCommand( cmd_string );
      ! TODO: ArapHandler calls report func
      !       OR return string?
      WriteStrBin comport,"OK 200"+"\0a";
    ENDWHILE

    ! Handle errors
  ERROR
    IF ERRNO=ERR_FILEOPEN THEN
      TPWrite "Could not open comport";
    ELSEIF ERRNO=ERR_FILEACC THEN
      TPWrite "Could not write to comport";
      Close comport;
    ELSEIF ERRNO=ERR_DEV_MAXTIME THEN
      TPWrite "Com port read timeout";
      Close comport;
    ENDIF

    RAISE;
ENDPROC
```

Listing A.2 arap.prg:com

```
FUNC string ReadStrBin2 ( VAR iodev dev, \num Time)
! Converts serial binary data to string

  VAR num character;
  VAR bool readmore := TRUE;
  VAR string result:="";

  WHILE readmore DO
      IF Present(Time) THEN
          character := ReadBin(dev \Time:=Time);
      ELSE
          character := ReadBin(dev);
```

```
        ENDIF

        TEST character
            CASE 10:
                readmore := FALSE;
            CASE 13:
                readmore := FALSE;
            DEFAULT:
                result := result + ByteToStr(character\Char);
        ENDTEST
    ENDWHILE

    RETURN result;
  ENDFUNC
```

Listing A.3 arap.prg:com

```
      CASE "ExecStr":
        get_next_arg callname, cmd;
        get_next_arg arg, cmd;
        ArapExecStr callname, arg;
```

Listing A.4 arap.prg:arap

```
  ! Executes the given procedure with string as parameter
  LOCAL PROC ArapExecStr( string call, string arg )
      % call % arg;
    ERROR
      % a_rep_func % ERRNO, "ExecStr Failed";
  ENDPROC
```

Listing A.5 arap.prg:arap

```
    PROC initHAPTOR()
        currSpeed := c_speed_parrot;
        setConfig( c_confdata );
        currConf := c_confdata;

        selToolAndWObj;
        currTool := sel_tool;
        currWobj := sel_wobj;
        c_oldPos := CRobT(\Tool:=currTool \Wobj:=wobj0);
        ConfL \Off;
        Open "HOME:" \File:="HAPTOR.LOG", logfile \Write;
        Close logfile;
    ENDPROC
```

Listing A.6 hptr_control.mod

```
     PROC getFlange()
        VAR iodev comport;
        Open "com2:", comport \Bin;
        WriteStrBin comport, ValToStr(CPos(\Tool:=tool0 \WObj:=
    wobj0)) + "\0a";
```

```
        Close comport;
    ENDPROC

     PROC getTCP()
        VAR iodev comport;
        Open "com2:", comport \Bin;
        WriteStrBin comport, ValToStr(CPos(\Tool:=currTool \WObj:=wobj0)) + "\0a";
        Close comport;
    ENDPROC
```

Listing A.7 hptr_control.mod

```
    PROC getJ()
        VAR jointtarget jtarget;
        VAR robjoint jvals;
        VAR iodev comport;

        jtarget := CJointT();
        jvals :=  jtarget.robax;

        Open "com2:", comport \Bin;
        WriteStrBin comport, ValToStr(jvals) + "\0a";
        Close comport;
    ENDPROC
```

Listing A.8 hptr_control.mod

```
    PROC RelBasePos(string xyz)
        VAR pos rel_pos;
        VAR robtarget c_target;
        VAR bool ok;

        ok := StrToVal(xyz, rel_pos);
        TPWrite("OK: "+ValToStr(ok));
        TPWrite("new_pos: "+ValToStr(rel_pos));

        c_target := CRobT(\Tool:=currTool \Wobj:=wobj0);
        c_target := Offs(c_target, rel_pos.x, rel_pos.y, rel_pos.z);
        MoveL c_target, currSpeed, fine, currTool \WObj:=wobj0;
    ENDPROC
```

Listing A.9 hptr_control.mod

```
class robotSerial:
    def __init__(self, debug_state = True):
        self._debug = debug_state
        if self._debug:
            self.port = 0
        else:
            self.port = self.connect()
            self.port.flushInput()
            self.port.flushOutput()
        return


    def connect(self):
        """
        Assumes that the robot is connected to first serial port.
        Must be run as root/superuser in Linux (sudo).
        TODO exception handling
        TODO kill all existing connections
        """

        settings_dict = {
            'port'          :   0,
            'baudrate'      :   9600,
            'bytesize'      :   serial.EIGHTBITS,
            'parity'        :   serial.PARITY_NONE,
            'stopbits'      :   serial.STOPBITS_ONE,
            'timeout'       :   10,
            'xonxoff'       :   0,
            'rtscts'        :   1,
            'writeTimeout':   None,
            'dsrdtr'        :   0
            }

        comportHandle = serial.Serial( **settings_dict )
        log.info('connect: ' + str(comportHandle))

        return comportHandle
```

Listing A.10 arap.py

```
    def command(self, arapCommand='', argument=None, sync=True):
        """ Send command to Arap running on robot controller.
            if sync is True, call receive, otherwise continue (
    asynchronous call)
            TODO exception handling
        """
        if type(argument) in [tuple, numpy.ndarray]:
            argument = list(argument)
        _str = str(argument).replace(' ','');

        string_value = arapCommand + ' ' + _str

        if type(argument) == type(None):
            arapString = 'Exec ' + arapCommand
        elif type(argument)==int or type(argument)==float:
            arapString = 'ExecNum ' + string_value
        else:
            arapString = 'ExecStr ' + string_value

        log.info( 'arapString: {0}'.format(arapString) )
        result   = self.send(arapString)

        if (result == 0) & sync:
```

```python
            result = self.receive()
            log.info( 'arapResult(sync): {0}'.format(result) )
        else:
            result = 'SYNC OFF'
            log.info( 'arapResult: {0}'.format(result) )

        if not result:
            if not self._debug:
                raise ArapException('can not execute command: %s' % str(arapCommand))
        return result
```

Listing A.11 arap.py

```python
    def send(self, arapString):
        # TODO: Exception handling?
        #       Add raise ArapException( ... )
        if type(arapString) != str:
            log.error('arap.send argument %s is not a string!' % str(arapString) )
            errorCode = -1
        elif len(arapString) > 80:
            log.error('arap.send argument %s is %d characters (max. allowed: 80)' % arapString, len(arapString))
            errorCode = -2
        else:
            if self._debug:
                log.debug('arap.send(debug):: send %s' % arapString)
            else:
                log.debug('arap.send:: send %s' % arapString)
                self.port.write(arapString + '\n')
                ##self.port.flush() #wait for write to finish
            errorCode = 0
        return errorCode
```

Listing A.12 arap.py

```python
    def receive(self):
        """Receive available lines on the serial port.
        """
        # TODO: Exception handling?
        #       Add raise ArapException( ... )
        if self._debug:
            log.debug('arap.receive(debug)')
            receivedString = 'debug'
        else:
            time_out = 60; # Set a time-out of 60 sec (changed from 3 to 5 to allow for changeOrient etc)
            start_time = time.time()
            cont = True
            while cont:
                bytes_avail = self.port.inWaiting()
                elapsed_time = time.time()-start_time
                if ( (bytes_avail > 0) or (elapsed_time>time_out)):
                    cont= False
            # print elapsed_time
            if elapsed_time>time_out:
                log.error('Nothing to read from serial port')
            receivedString = ''
            while self.port.inWaiting() > 1:
                receivedString = receivedString + self.port.readline()
                log.debug('receive: ' + receivedString.rstrip())
        return receivedString
```

Listing A.13 arap.py

```
class HIDReport:

    def _fmt_hid_piece(self, piece):
        result = list(piece)
        #result.reverse()
        #return int(reduce(lambda x,y: str(x)+str(y), result), 16)
        if len(piece) == 1:
            import pdb; pdb.set_trace()
        return int(reduce(lambda x,y: str(y)+str(x), result), 16)

    def __init__(self, report=[], hex_report=None):
        result = OrderedDict()

        if hex_report is None:
            hex_rep = map(lambda val: format(val, 'x'), report)
        else:
            hex_rep = hex_report

        result['id']       = int(hex_rep[0], 16)
        result['buttons']  = int(hex_rep[1], 16)
        result['x']        = self._fmt_hid_piece( hex_rep[2:2+4] )
        result['y']        = self._fmt_hid_piece( hex_rep[6:6+4] )
        result['pressure'] = self._fmt_hid_piece( hex_rep[-10:-10+2] )
        result['tiltx']    = self._fmt_hid_piece( hex_rep[-8:-8+2] )
        if result['tiltx'] & 32768 == 32768:
            result['tiltx'] -= 65535
        result['tilty']    = self._fmt_hid_piece( hex_rep[-6:-6+2] )
        if result['tilty'] & 32768 == 32768:
            result['tilty'] -= 65535
        result['twist']    = self._fmt_hid_piece( hex_rep[-4:] )
        result['raw']      = list(hex_rep)
        self.report = OrderedDict(result)

    def __str__(self):
        return json.dumps(self.report, indent=4)

    @property
    def data(self):
        return dict(self.report)

    @property
    def pos(self):
        return self.report['x'], self.report['y']

    @property
    def angles(self):
        return [self.report[key]/100.0 for key in ['tiltx','tilty
','twist']]

    @property
    def buttons(self):
        return self.report['buttons']

    @property
    def pressure(self):
        return self.report['pressure'] / 2047.0
```

Listing A.14 hidparser.py

```
class BTPen:
    def _enum(self):
        devices = hid.enumerate()
        if not devices:
            raise Exception('No bluetooth device found!')

        for dev in devices:
            if 'pen' in dev['product_string'].lower():
                return dev

    def vid(self):
        return self._enum()['vendor_id']

    def pid(self):
        return self._enum()['product_id']

    def __init__(self):
        self.dev = hid.device()
        try:
            self.dev.open(self.vid(), self.pid())
        except IOError:
            raise IOError('Could not connect to bluetooth pen!')
        print json.dumps(self._enum(), indent=4)
        self._init = True

    def read(self):
        hid_report = self.dev.read(20)
        print hid_report
        return HIDReport(hid_report)

    def record(self, num_reports=100, num_warmup=50):
        res = []
        while len(res) < (num_reports + num_warmup):
            read_hid = self.read()
            data = read_hid.data
            res.append(data)
            print 'Coord #{}'.format(len(res))
        return res[num_warmup:]

    def __del__(self):
        if self._init:
            self.dev.close()

            del self.dev
            self.dev = None

            del self._init
            self._init = None
```

Listing A.15 hidparser.py

```
class PenED:
    def __init__
    def __del__
    def disconnect
    def reset
    def trigger_hover
    def coords_on
    def coords_off
    def __reset_states
    def stop
    def check_hit
    @property
        def position
    @property
        def orientation
    @property
        def hit
    @property
        def alive
    @property
        def fsr
    @property
        def fsr_adc
    @property
        def pen_down
    @property
        def pen_up
```

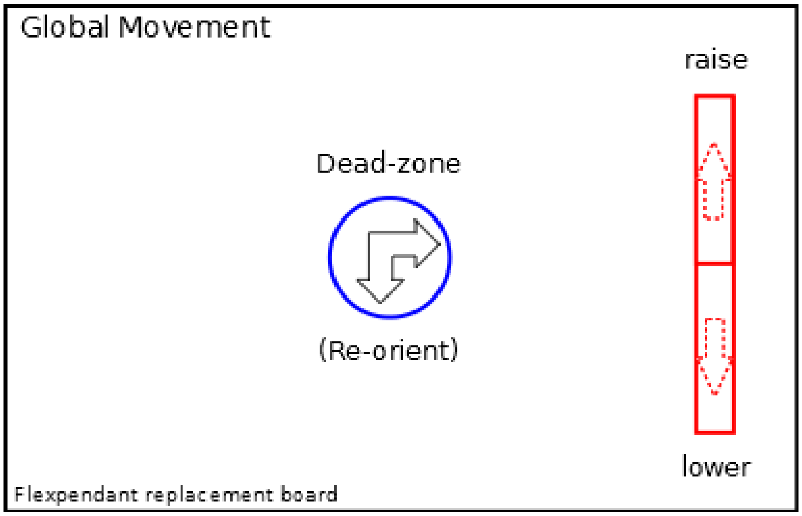
Listing A.16 Proxy interface implementation of the physical Anoto pen.

```
class Robot
```

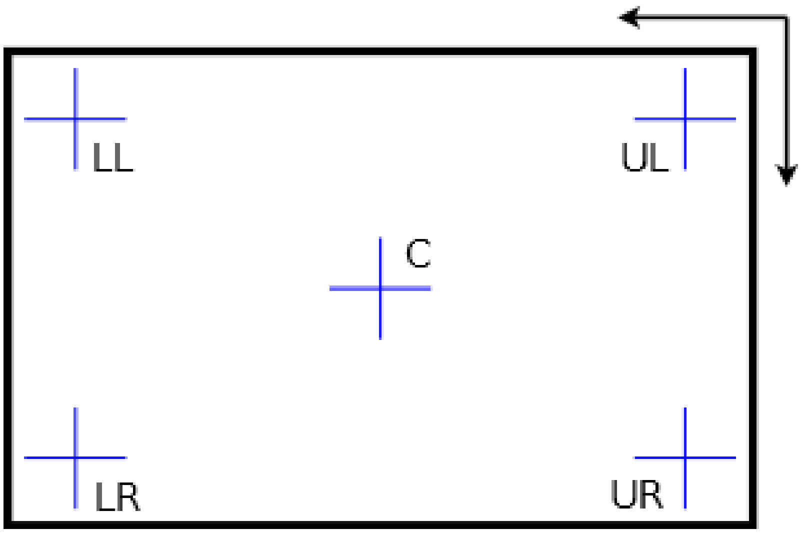
```
    def _upload_files(self):
        ftp = robotFileTransfer(debug_state = debug)
        ftp.connect()
        ftp.upload('ARAPMOD.prg', robot_com.paths['serial'])
        ftp.upload('hptr_control.mod', robot_com.paths['server'])
        ftp.upload('common.mod', robot_com.paths['server'])
        ftp.upload('calibrationdata.mod', robot_com.paths['server'])
        ftp.disconnect()
        self._ftp = ftp

    def _init_serial(self):
        pass
        arap = robotSerial(debug_state = debug)
        self._arap = arap
        arap.load('hptr_control.mod')
        arap.load('hptr_control.mod') # perform 2 times if pen is inited before robot
        arap.load('common.mod')
        arap.load('calibrationdata.mod')
        arap.command('initHaptor')
        self.move_to_door()

    def __init__(self,lock = None, angle=45, pen_interface=None, num_data=16):
        self._upload_files()
        self._init_serial()

        self.lock = lock
        self.all_data = []

        self.start_time = time.strftime('%H%M')

        self.alive = False
        self._finished = False
        self.num_data_points = num_data
        if pen_interface:
            self.pen = pen_interface
            self.pen_hit_thread = Thread(target=pen.check_hit)
            self.pen_hit_thread.start()
            time.sleep(1)
            with self.lock:
                self.alive = self.pen.alive
            if pen.alive:
                self.move_to_ready(angle)
                self.save_tool_pos()
    def __del__
    @property
        def arap
    def arap
    def __abort
    def move_to_door
    def set_vel_molusk
    def set_vel_parrot
    def set_vel
    def set_ang_vel
    def set_rel_tcp_z
    def save_tool_pos
    def move_to_saved_pos
    def move_to_saved_pos_ori
    def move_to_saved_ori
    def move_tcp
    def move_wobj_tcp
    def move_flange
    def rel_tool_ori
    def rel_tool_dir
    def rel_tool_z
    def abs_tool_dir
    def move_to_ready
    def get_flange
    def get_tcp
    def get_joints
    def __grab_data
    def __find_start_pos
    def __search_tool_z
    def __rel_toolz_ori
    def paper_verify
    def paper_search
    def parrot
```

Listing A.17 Proxy interface implementation of the IRB140 robot.

A.8 Control Board Design and Target Surface



A.6 Control board design without the Anoto dot-pattern.

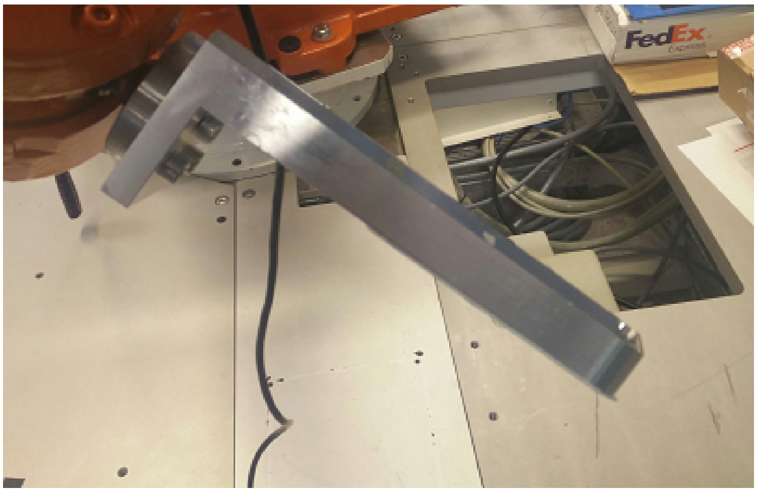


A.7 Exaggerated depiction of target surface with its cross markers at known Anoto coordinates, excluding the Anoto dot pattern.

A.9 Measurements Lab Setup

A.8 An Anoto *Live* pen in its cradle with its digital display turned off.

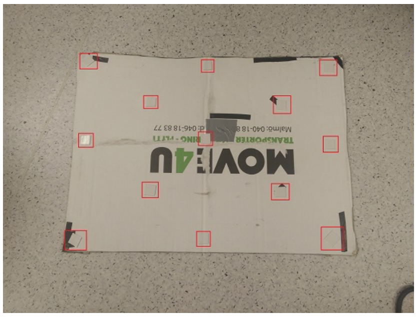
A.9 A prototype pen with its green status LED active.



A.10 The pen holder seen from its left side with the cover and screws removed.

A.11 The calibration tip in its wingnut.

A.12 The calibration tip mounted on the IRB140.



A.13 Underside of the work object with the 13 white adhesive tape squares placed evenly across its surface. The adhesive tapes are marked with red squares to make them more visible.

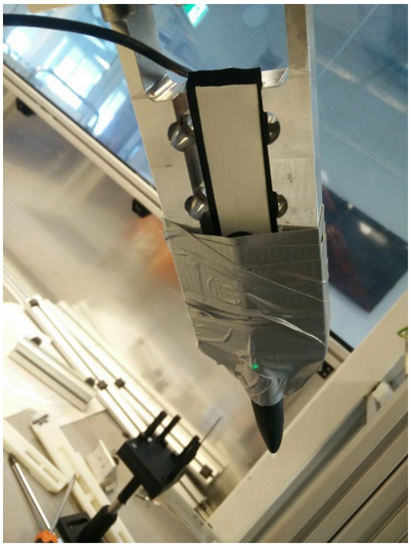
A.14 The target placed on the work object surface.

A.15 Pen holder with adhesive tape attached on its surface.

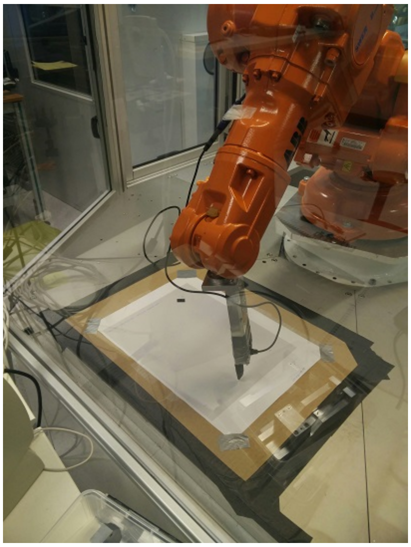
A.16 An Anoto *Live* pen with its cradle attached on the adhesive tape on the pen holder.

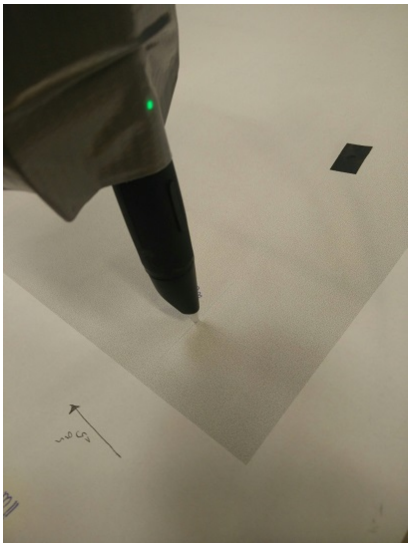
A.17 An Anoto *Live* pen and cradle fastened firmly with duct tape seen from its top side.

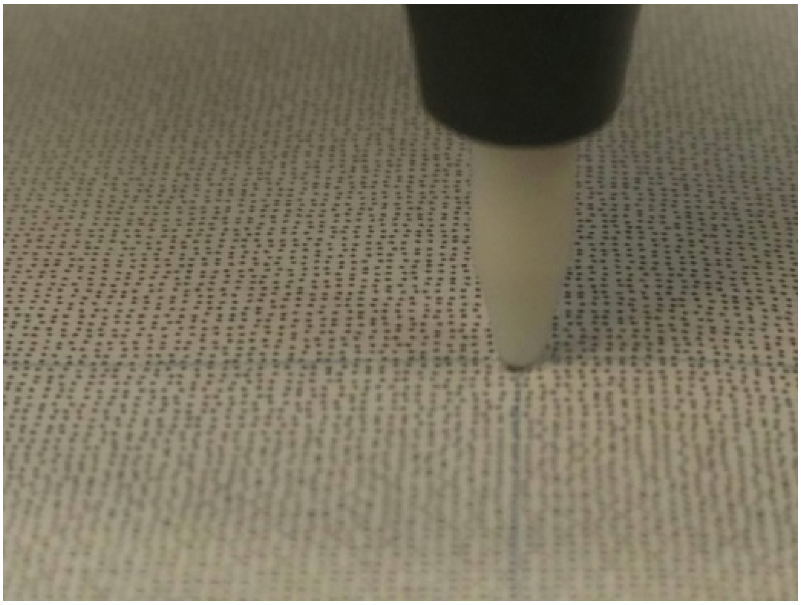
A.18 An Anoto *Live* pen and cradle fastened firmly with duct tape seen from its left side


A.19 A prototype pen fastened firmly with duct tape seen from its top side. Its green status LED can be seen through the duct tape.

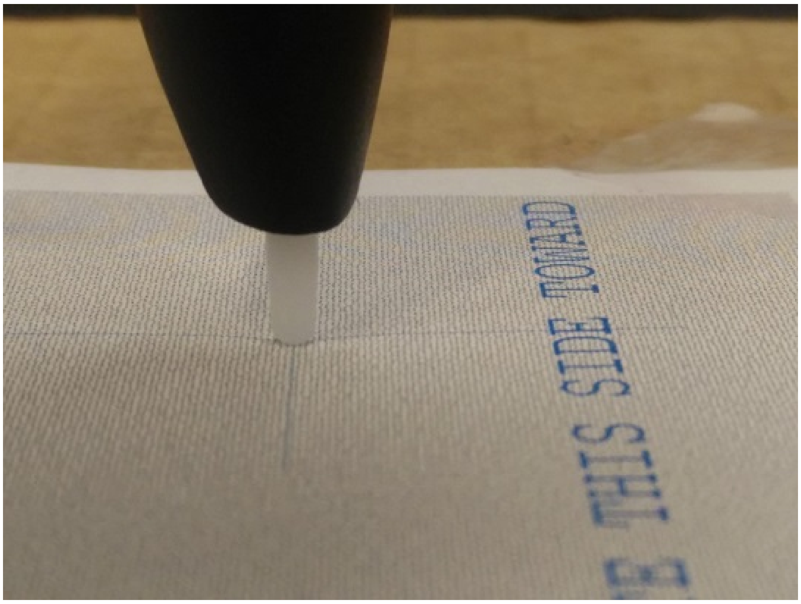
A.20 A prototype pen fastened firmly with duct tape seen from its left side.


A.21 The robot performing a measurement on the target with the prototype pen.


A.22 Close up of the prototype pen with target surface.



A.23 Anoto *Live* pen tip placed on a target marker after manual tool calibration.



A.24 Prototype pen tip placed on a target marker after manual tool calibration.

Bibliography

ABB Ltd (2016a). *Fanta can challenge*. Data Sheet. URL: `http://new.abb.com/products/robotics/industrial-robots/irb-140` (visited on 2016-04-10).

ABB Ltd (2016b). *Fanta can challenge level II - Superior motion control ABB robotics*. Embedded YouTube video. URL: `http://www.youtube.com/embed/SOESSCXGhFo` (visited on 2016-04-10).

ABB Ltd (2016c). *IRB140 industrial robot*. Data Sheet. URL: `http://new.abb.com/products/robotics/industrial-robots/irb-140/irb-140-data` (visited on 2016-04-10).

ABB Ltd (2016d). *IRB140T high speed upgrade industrial robot upgrade program*. URL: `http://new.abb.com/products/robotics/industrial-robots/irb-140` (visited on 2016-04-10).

Allen, P. K., A. Timcenko, B. Yoshimi, and P. Michelman (1993). "Automated tracking and grasping of a moving object with a robotic hand-eye system". *IEEE Transactions on Robotics and Automation* **9**:2, pp. 152–165.

Anoto AB (2017a). *The Anoto partner program*. URL: `http://www.anoto.com/anoto-partner-network` (visited on 2017-01-26).

Anoto AB (2017b). *The power to capture it all*. URL: `http://www.anoto.com/legal-notice` (visited on 2017-01-21).

Anoto Group AB (2016a). Case Studies. URL: `http://www.anoto.com/enterprise/case-studies` (visited on 2016-04-13).

Anoto Group AB (2016b). *Maternity support services welcomes new data capture solutions*. Case Studies: Healthcare. URL: `http://www.anoto.com/enterprise/case-studies/healthcare` (visited on 2016-04-13).

Anoto Group AB (2016c). *Nationwide service and 24/7 emergency calldesk realised with digital writing*. Case Studies: Facilities Management. URL: `http://www.anoto.com/enterprise/case-studies/facilities-management` (visited on 2016-04-13).

Bluetooth Special Interest Group (2016a). *Bluetooth specification human interface device profile 1.1*. Specification. URL: `https://developer.bluetooth.org/TechnologyOverview/Pages/HID.aspx` (visited on 2016-04-20).

Bluetooth Special Interest Group (2016b). *Bluetooth website*. URL: `https://www.bluetooth.com` (visited on 2016-04-20).

Bruyninckx, H. (2010). *Robot Kinematics and Dynamics*. Katholieke Universiteit Leuven, Leuven, Belgium.

Chen, H. H. (1991). "A screw motion approach to uniqueness analysis of head-eye geometry". In: *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, pp. 145–151.

Chou, J. C. and M Kamel (1991). "Finding the position and orientation of a sensor on a robot manipulator using quaternions". *The International Journal of Robotics Research* **10**:3, pp. 240–254.

Compaq, DEC, IBM, Intel, Microsoft, NEC, and Nortel (2016). *Device class definition for human interface devices (HID)*. Firmware Specification—6/27/01 Version 1.11. URL: `http://www.usb.org/developers/hidpage/HID1_11.pdf` (visited on 2016-04-20).

Craig, J. J. (1986). *Introduction to Robotics: Mechanics & Control*. Addison-Wesley, Reading, Massachusetts. ISBN: 0-201-10326-5.

Daniilidis, K. (1999). "Hand-eye calibration using dual quaternions". *The International Journal of Robotics Research* **18**:3, pp. 286–298.

Destiny Wireless (2017). *A note on Anoto AB*. Blog Post. URL: `http://digitalpennews.typepad.com/blog/anoto` (visited on 2017-01-21).

Ernst, F., L. Richter, L. Matthäus, V. Martens, R. Bruder, A. Schlaefer, and A. Schweikard (2012). "Non-orthogonal tool/flange and robot/world calibration". *The International Journal of Medical Robotics and Computer Assisted Surgery* **8**:4, pp. 407–420.

Freidovich, L. B. (2013). *Control Methods for Robotic Applications*. Lecture Notes. St. Petersburg, Russia.

Google Inc. (2016). *Guidelines for stylus accessory creators*. URL: `https://source.android.com/devices/accessories/stylus.html#guide-creators` (visited on 2016-04-21).

Harada, K., T. Tsuji, K. Nagata, N. Yamanobe, and H. Onda (2014). "Validating an object placement planner for robotic pick-and-place tasks". *Robotics and Autonomous Systems* **62**:10, pp. 1463–1477.

Hartenberg, D. J. and R. Scheunemann (1955). "A kinematic notation for lower-pair mechanisms based on matrices". *Trans ASME J. Appl. Mech.* **23**, pp. 215–221.

Heller, J., D. Henrion, and T. Pajdla (2014). "Hand-eye and robot-world calibration by global polynomial optimization". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, pp. 3157–3164.

Horaud, R. and F. Dornaika (1995). “Hand-eye calibration”. *The International Journal of Robotics Research* **14**:3, pp. 195–210.

Livescribe, Inc (2017). *About Livescribe*. URL: `https://www.livescribe.com/en-us/company` (visited on 2017-01-26).

Luca, A. D. (2015). *Robotics 1 trajectory planning*. Lecture Notes. URL: `http://www.diag.uniroma1.it/~deluca/rob1_en/13_TrajectoryPlanningJoints.pdf` (visited on 2017-01-14).

McDermott-Wells, P. (December 2004 / January 2005). “What is bluetooth?” *IEEE Potentials Magazine*.

Microsoft (2015). *Supporting usages in digitizer report descriptors*. Specification. URL: `https://msdn.microsoft.com/en-us/library/windows/hardware/jj151564%28v=vs.85%29.aspx` (visited on 2015-03-15).

Microsoft, Intel, and Logitech (2016). *Hid usage tables 10/28/2004 version 1.12*. Manual. URL: `http://www.usb.org/developers/hidpage/Hut1_12v2.pdf` (visited on 2016-04-21).

Myronenko, A. and X. B. Song (2009). “On the closed-form solution of the rotation matrix arising in computer vision problems”. *CoRR* **abs/0904.1613**. arXiv: `0904.1613`. URL: `http://arxiv.org/abs/0904.1613`.

NECS, inc (2017). *Anoto digital pen promotion*. Blog Post. URL: `https://necs.com/blog-article.php?id=39` (visited on 2017-01-21).

Pettersson, M.-P. and T. Elsö (1999). “Positionsbestämning p**r**a en yta försedd med ett positionskodningsmönster”. Pat. SE 9903541-2. (Patent- och Registreringsverket). Publication number: SE 517445, Published as: SE517445 C2, SE9903541 L.

Ruland, T., T. Pajdla, and L. Krüger (2012). “Globally optimal hand-eye calibration”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 1035–1042.

Rusnak, P. (2015). *hidapi 0.7.99.post14*. Python Module. URL: `https://pypi.python.org/pypi/hidapi/0.7.99.post14` (visited on 2015-11-06).

Shiu, Y. C. and S. Ahmad (1989). “Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb”. *IEEE Transactions on Robotics and Automation* **5**:1, pp. 16–29.

Silberman, S. (2001). *The hot new medium: paper*. Interview. URL: `http://www.wired.com/2001/04/anoto` (visited on 2016-04-13).

Spong, M. W. and M. Vidyasagar (1989). *Robot Dynamics and Control*. 1st. John Wiley & Sons, New York, NY, USA. ISBN: 0-471-61243-X.

Strobl, K. H. and G. Hirzinger (2006). “Optimal hand-eye calibration”. In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, pp. 4647–4653.

Tsai, R. Y. and R. K. Lenz (1989). “A new technique for fully autonomous and efficient 3d robotics hand/eye calibration”. *IEEE Transactions on Robotics and Automation* **5**:3, pp. 345–358.

Umeyama, S. (1991). “Least-squares estimation of transformation parameters between two point patterns”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13.4**, pp. 376–380.

Xms Penvision (2017). *The ingenious way to create & complete forms*. Newsletter. URL: `http://penvision.com/newsletter/r35release.html` (visited on 2017-01-21).

Zhuang, H., Z. S. Roth, and R. Sudhakar (1994). “Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form ax= yb”. *IEEE Transactions on Robotics and Automation* **10**:4, pp. 549–554.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* April 2018 *Document Number* TFRT-6051
Author(s) Johnny Sjöberg	*Supervisor* Per Lidström, Department of Mechanical Engineering, Lund University, Sweden Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Robot Tool Calibration of an Active Pen with Python using an Enabled Surface from Anoto Technology

Abstract

This master thesis has shown multiple uses of the Anoto dot pattern and Anoto pen for different applications. One application is the calibration of the robot tool and work object. Another is the use of an Anoto pen and paper as replacement of the ABB Flexpendant to create a lightweight and intuitive control device to control an industrial robot, experimentally demonstrated for the IRB140 robot arm. The Python programming language and interpreter were used together with ABB's RAPID programming language to create a framework to handle the communication between the user and the robot.
The inverse and forward kinematics were derived for the robot and used for simulations and for determining if a trajectory could be traversed. This helped in the theoretical verification in the development of the calibration algorithm. The calibration details from the results section show that a naive approach to calibrating the tool tip, i.e., using all of the measurement values results in a deviation of 1mm or more with slow convergence compared to the manual calibration of 0.5mm deviation. By using a greedy optimization strategy of successively adding measurements that improve the calibration, and removing measurements that worsen it, we get a calibration comparable to a manual calibration with fast convergence.
The work object calibration, i.e., the orientation estimation of the Anoto pattern surface shows promising results with few measurements and faster convergence, but can also be improved with the same optimization strategy. By performing manual measurements, and using measurements of the error in the robot flange position, a realistic lower bound on the precision of the calibration algorithm was decided to be no lower than 0.3 - 0.4mm, and depended on the performance of the pen. The flexpendant control board replacement was developed through a user study with volunteers from Anoto.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-102	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/