TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

OSSI TEIKARI

REPRODUCING HANDWRITING WITH POSITION-CONTROLLED ROBOTS

Master of Science thesis

Examiner:
Prof. Hannu-Matti Järvinen
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 13th January 2016

TIIVISTELMÄ

OSSI TEIKARI: Käsinkirjoituksen toistaminen paikkaohjatuilla teollisuusroboteilla
Tampereen teknillinen yliopisto
Diplomityö, 51 sivua
Marraskuu 2016
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Pervasive Systems
Tarkastaja: professori Hannu-Matti Järvinen

Kosketuslaitteiden suorituskyvyn testaus edellyttää testijärjestelmältä korkeaa tarkkuutta ja toistettavuutta. Tämän vuoksi teollisuusrobotteja voidaan pitää hyvänä työkaluna kosketuslaitteiden testaukseen. Testausprosessissa on tärkeää laitteen testaaminen eleillä ja toiminnoilla, jotka ovat mahdollisimman lähellä laitteen aitoa käyttötarkoitusta. Jotkin eleet, kuten näytön painelu ja pyyhkiminen, on suhteellisen helppo toteuttaa teollisuusroboteilla. Monimutkaisemmat eleet, kuten käsinkirjoitus, tuottavat kuitenkin huomattavasti suurempia haasteita testijärjestelmälle.

Tämän työn päätavoitteena on realistisen käsinkirjoituksen toistaminen teollisuusroboteilla. Robotin tulisi ensinnäkin pystyä toistamaan alkuperäisen kirjoituksen rata korkealla tarkkuudella. Tämän lisäksi sen tulisi myös kyetä jäljittelemään kirjoituksen temporaalisia ominaisuuksia, kuten kynän nopeutta ja kiihtyvyyttä eri pisteissä. Tavoitteen saavuttamiseksi käsinkirjoitettu data nauhoitetaan digitaaliseen muotoon ja lopputuloksen parantamiseksi näytteistettyä dataa esikäsitellään eri tavoin. Tämän jälkeen data siirretään robottiohjaimelle, jonka säätöjärjestelmä pyrkii toistamaan kirjoituksen mahdollisimman tarkasti.

Projektia varten nauhoitettiin useiden koehenkilöiden kirjoittamaa tekstiä käyttäen apuna digitaalista piirtopöytää sekä tätä projektia varten luotua PC-sovellusta. Lisäksi valittu teollisuusrobotti integroitiin PC-ympäristöön. Tämän jälkeen selvitettiin erilaisia keinoja, joilla robotti kykenisi toistamaan kirjoitusdatan nopeus- ja kiihtyvyysprofiilit mahdollisimman hyvin. Tärkeimpiä keinoja robotin kyvykkyyden parantamiseen havaittiin olevan kirjoitusdatan approksimointi käyttäen mahdollisimman pientä määrää pisteitä, sekä robotin liikkumisparametrien muuttaminen tekstin eri pisteissä. Lopuksi nauhoitettu kirjoitusdata toistettiin robotilla, ja tuloksia vertailtiin alkuperäisiin näytteisiin.

Kirjoitusdatan approksimointi sekä robotin liikeparametrien säätäminen vaikuttivat positiivisesti tuloksiin. Tuloksia analysoitaessa kuitenkin havaittiin, että robotti ei kykene jäljittelemään ihmisen kirjoitusta täydellisesti. Erityisesti suuret kiihtyvyyden vaihtelut tuottivat järjestelmälle merkittäviä vaikeuksia. Tällaisia vaihteluita syntyy ihmisen kirjoittaessa pienehköä tekstiä suurella nopeudella. Potentiaalisimpien liikedatan approksimointialgoritmien jatkokehitys saattaisi tuottaa parannuksia tuloksiin tulevaisuudessa. Toisena kehitysehdotuksena on robotin vaihtaminen malliin, joka sallii syvemmän pääsyn sen sisäisiin liikkeen säätöjärjestelmiin ja tarjoaa näin lisää vaihtoehtoja liikkeiden optimointiin.

ABSTRACT

OSSI TEIKARI: Reproducing Handwriting with Position-Controlled Robots
Tampere University of Technology
Master of Science Thesis, 51 pages
November 2016
Master's Degree Programme in Information Technology
Major: Pervasive Systems
Examiner: Professor Hannu-Matti Järvinen

Keywords: industrial robots, handwriting, touch device testing

Testing the performance of touchscreen devices requires high accuracy and repeatability from the tester. This is why industrial robots can be considered excellent tools for this task. When performing end-to-end testing for touchscreen devices, realistic touch inputs have to be applied to the device under test. Simple inputs such as taps and swipes can be accomplished relatively easily with industrial robots. However, to test more complicated features, the robots should also be able to perform more complicated actions, such as handwriting. This introduces much bigger challenges for the test system.

The goal of this project is to utilize an industrial robot to reproduce handwriting as accurately as possible. In addition to replicating the original pen path, the robot should also be able to replicate temporal qualities such as pen velocity in different points of the path. First of all, this project includes sampling real handwriting data in digital format. To improve the robot performance, the data has to be pre-processed in various ways. The data is then transmitted to the robot controller, which will replicate the sampled writing trajectories as well as possible. Lastly, the results have to be measured so that the performance of the system can be evaluated.

During the project, handwriting was sampled from several test persons with a software application created specifically for this project. Software for interfacing with the chosen industrial robot platform was also developed. After this, different methods to best adapt the raw handwriting data to the robot were studied. These methods include approximating the data with fewer sample points, as well as adjusting the robot movement parameters in different points of the writing. Finally, the sampled handwriting dataset was replicated with the robot, and the results were compared with the original samples.

As a conclusion, the data approximation methods and robot movement parameter tuning were able to significantly improve the robot movement performance. However, it was discovered that the robot was not able to perfectly replicate all aspects of human writing. Especially the high pen accelerations within small distances were difficult to reproduce with the robot. Further improving the results would require more work with the data approximation algorithms, and perhaps switching the robot to a model that offers deeper access to the underlying movement control system.

PREFACE

Overall, this project offered me a chance to work on truly diverse and interesting tasks. Getting to create software that works with real robot hardware allowed me to catch a glimpse of the intersection of various engineering fields. This experience will certainly benefit me in the future.

I would like to thank OptoFidelity for providing all the necessary tools and equipment that were required for this thesis. I’m also grateful that the company allowed me to work on this thesis alongside all the other ongoing projects. This helped me complete the work within a reasonable time. I especially want to thank Janne Honkakorpi and Kimmo Jokinen (OptoFidelity), and professor Hannu-Matti Järvinen (TUT), for providing valuable feedback and ideas during the project.

Tampere, 13th November, 2016

Ossi Teikari

TABLE OF CONTENTS

TERMS AND ABBREVIATIONS

CAD	Computer-Aided Design
CP	Continuous Path motion, corresponds to robot Cartesian-space movement
DP algorithm	Douglas–Peucker algorithm
DUT	Device under Test
EMR	Electromagnetic Resonance
LPI	lines per inch
NC machine	Numerical Control machine
OptoFidelity TnT	OptoFidelity Touch and Test, a software suite for touch performance testing
OptoFidelity TPPT	OptoFidelity Touch Panel Performance Tester, a product for measuring accuracy of touch panels
PID controller	Proportional–integral–derivative controller, feedback control mechanism commonly used in industrial systems
PTP	Point-to-Point motion, corresponds to robot joint-space movement
TCP	Tool Center Point, center point of the robot end-effector

1. INTRODUCTION

Device manufacturers have always been looking to make user interaction with touchscreen devices as natural and comfortable as possible. Many of the most natural interaction methods are still the same ones that people have been using for years, long before the emergence of the first touchscreen product. These methods include, for example, voice interaction, as well as gestures such as swipes, flicks, and taps, pressure sensitivity, and handwriting.

Many people consider handwriting and drawing very natural ways of expressing themselves. Even though people are nowadays able to type incredibly fast using an on-screen keyboard, many users still appreciate the possibility of being able to jot down freehand notes or sketch a design just like one would on real paper. There already exists a variety of mobile applications that support handwriting gestures. These include Evernote and Microsoft OneNote. Microsoft is even offering native handwriting recognition capability in its Surface Pro line of products. Even Apple is reportedly working on a new kind of UI that is based on handwriting [27].

Having been handwriting all their lives using regular pen and paper, many people have developed an intuition about how handwriting "should" feel like. This makes creating natural handwriting experience on a touch device a significant challenge. For a natural writing experience, position of the stylus tip should be sensed as accurately as possible by the touch panel, and the end-to-end latency of the device should be minimized.

Developing hardware and software for these kinds of devices is often an iterative process which requires a large amount of testing. One way of carrying out handwriting testing for such touch devices would be doing all the user interaction tests manually. However, in addition to being very labor-intensive, this kind of testing is not very repeatable, and it often produces subjective test results. If repeatable and objective test sequences are desired, another option is to use robots to perform non-intrusive handwriting testing. In this approach, a highly repeatable industrial robot would be used to write a set of natural handwriting examples on the device. The device performance would then be measured using cameras and sensors. After making changes to the hardware or software of the developed product, effects on device performance could be measured by performing the same standardized tests again. Based on past discussions, such approach could be very interesting for the existing and potential customers of OptoFidelity.

The main goal of this thesis is to experiment with using industrial robots to perform handwriting on touch-screen devices, and implement a demonstration system that achieves this. The ultimate goal is to be able to test how well current touch-screen devices are able to perceive handwriting input. In this context, the following handwriting characteristics are deemed most relevant:

- pen tip X and Y coordinates
- pen tilt
- pen tip velocity
- pen tip acceleration
- pen tip pressure.

From the end-user application point of view, it makes sense for modern touch-screen devices to be able to accurately measure these parameters. For example, there already exists drawing applications for tablets where the stroke form is varied based on the tilt of the input stylus, and where the stroke width is varied based on the applied pressure [1]. On the other hand, pen tip velocity and acceleration may have an effect on how accurately touch panels are able to measure the tip X and Y coordinates.

Some similar work has previously been done in this area. Before the emergence of modern printers, plotters were commonly used e.g. in the *Computer Aided Design* (CAD) industry. Plotters are essentially devices that are capable of printing vector graphics to paper or other surfaces, by using a pen. Human handwriting samples could theoretically be converted into vector graphics files and reproduced with plotters. However, this would likely require a lot of manual work regarding managing pen up and down points, speeds, and pressures. Pen angle would also be difficult to emulate with a plotter device. Availability and support are also questionable, since nowadays plotters are largely obsolete and mainly used by hobbyists [19].

A company called Bond has also created a robot that is capable of replicating human handwriting with real pen and paper [10]. The company has launched a commercial service where customers can provide the company with samples of their handwriting, which the Bond robot is then able to learn and replicate. Customers can then order arbitrary amounts of handwritten notes, invitations, etc., printed using their own handwriting style.

Researchers in Rijksuniversiteit Groningen and Fraunhofer IPK built a robotic handwriting system using a commercial CNC machine [9]. The system is pictured in *Figure 1*. The researchers attached a pen to the machine with a custom-designed holder. The pen was kept at a 55-degree angle compared to the surface, which allegedly corresponds to the most natural pen angle for human writers [9]. The pen holder included a spring for suspension in the Z axis direction. Linearity of the spring was utilized while mapping the robot Z distance to resulting pen tip pressure.

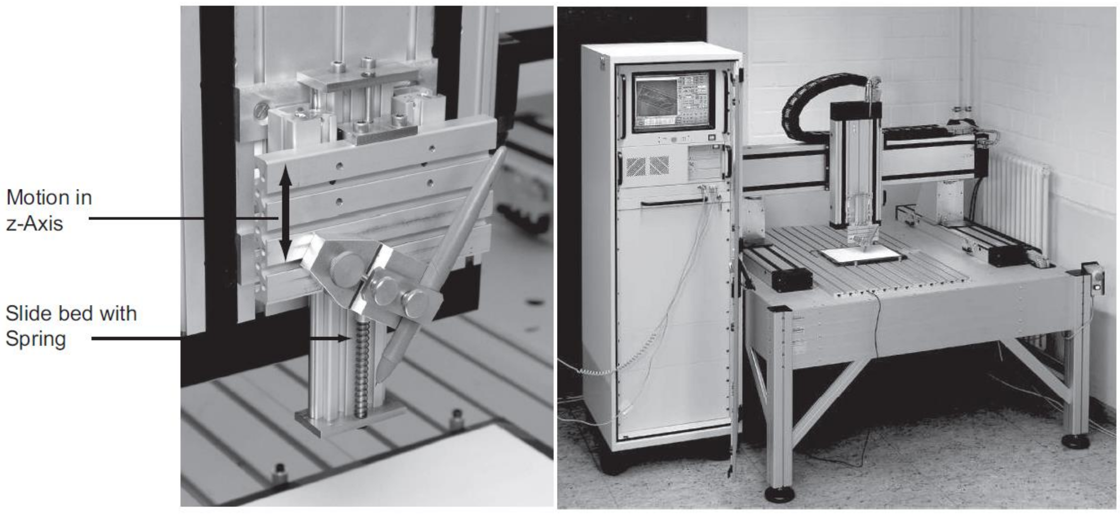

Figure 1. *CNC machine utilized for handwriting replication [9]*

The researchers then used an electronic pen and tablet for capturing human handwriting trajectories. Trajectories were converted into G-code commands that are understood by the CNC machine. In addition to X and Y coordinates, sampled pen pressure values could be replicated by converting them into Z coordinates. The project yielded some potential results regarding replicating the XY trajectory, as well as the pressure curve of the original sample. However, it is reported that the CNC machine has a maximum rated velocity of 214 mm/s. Moreover, the researchers state that during testing the machine speed was limited to 33 mm/s due to mechanical limitations of the pen holder. These limitations mean that the system cannot achieve high enough speeds for replicating fast human handwriting.

In the context of this thesis, target use case for the handwriting robot differs from the above examples. First of all, the goal is not to write on real paper, but on touchscreen devices. The second and more important difference is that to enable as natural testing as possible, the goal is not just to control how the produced writing looks, but also how it is produced regarding pen speed, acceleration, pressure, and tilt angle.

The organization of the thesis is following: in chapter 2, necessary basic theory behind industrial robots is introduced. Chapter 3 discusses concrete use cases and goals for the handwriting robot. It also describes what kinds of challenges these goals introduce, and offers some possible solutions to the challenges. Chapter 4 illustrates the system implemented within this thesis, and chapter 5 shows what kind of results the system can produce. Chapter 6 concludes the work and discusses some potential areas for future development.

2. INDUSTRIAL ROBOTS

2.1 Background

This chapter discusses fundamental topics about industrial robots and their control. Focus is kept strictly within topics that are necessary in order to understand the design decisions made while implementing the handwriting robot system. More comprehensive introduction into robotics can be found in various textbooks, e.g. [11] and [14].

There does not exist a single exhaustive definition for the term “robot”. Generally speaking, robots are machines that are not limited to just one specific task but can be programmed to perform various kinds of work. This separates them from common automated machinery [13]. Industrial robots are a subset of robots that are used to perform industrial tasks such as welding, painting, and repetitive pick-and-place movements.

Industrial robots can be divided into *serial chain* and *parallel chain* robots. Serial chain robots are the most common type of robot in industrial settings. They consist of rigid links that are connected together to form a simple chain from the robot base plate to the end-effector. Parallel chain robots consist of multiple serial links that extend to support a common end-effector. For clarity, this chapter explains the theoretical concepts using *serial chain robot* as an example. The same principles can also be used to understand the principles of *parallel link robots* such as the Fanuc M-1ia delta robot [17].

2.2 Basic Mechanical Structure

Serial chain industrial robots can be modeled simply by rigid links that are joined together by joints. The first end of the first link is connected to the robot base plate, and the robot tool, often called end-effector, is attached to the end of the last link. Examples of basic joint types are revolute joints (capable of rotation with respect to a single axis) and prismatic joints (capable of movement along a single straight axis). Rotating electric motors can easily be used to actuate revolute joints, whereas linear motors can directly be used to actuate prismatic joints [15]. Rotary and linear encoders are commonly used to measure positions of the joints. Other joint types, such as spherical joints, are shown in *Figure 2*, but their use in commercial robots is not as common due to increased mechanical complexity [12].

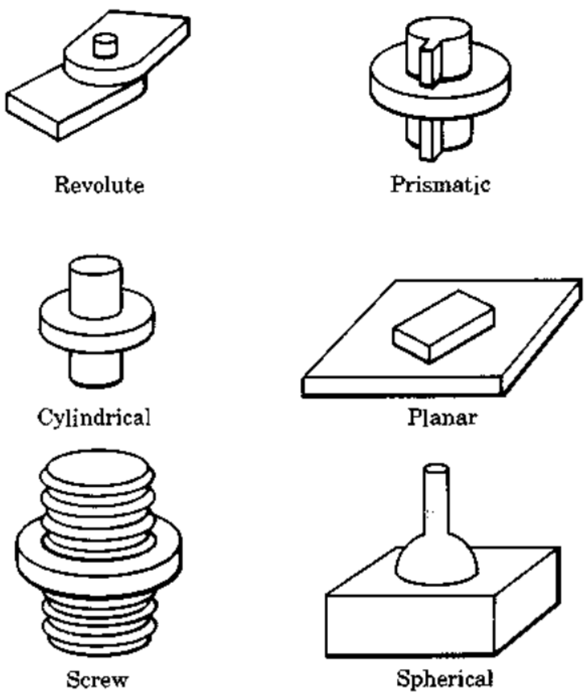

Figure 2. *Six different types of joints [12]*

Number of degrees of freedom of a robot corresponds to the minimum number of variables needed to specify a single position or “state” of the robot. Six degrees of freedom is the minimum amount needed to be able to position the robot end effector in any position and orientation in the 3-dimensional Cartesian space [16]. An industrial robot with six degrees of freedom can be constructed e.g. by connecting five rigid links together with six rotational joints. A common example of this type of a robot can be seen in Figure *3*.

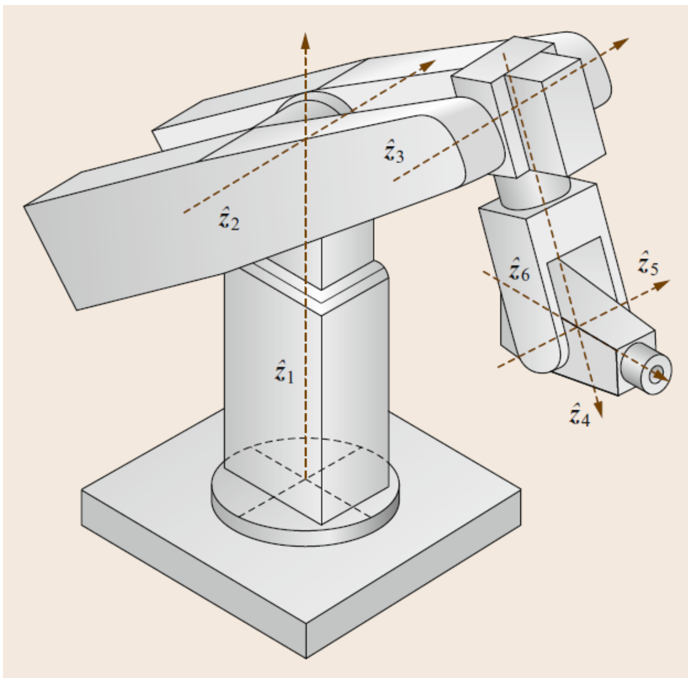

Figure 3. *Industrial robot with six degrees of freedom [15]*

Other types of widely used industrial robots include Cartesian, SCARA, and Delta style robots. Examples of these are shown in Figure *4*, Figure *5*, and Figure *6*, respectively.

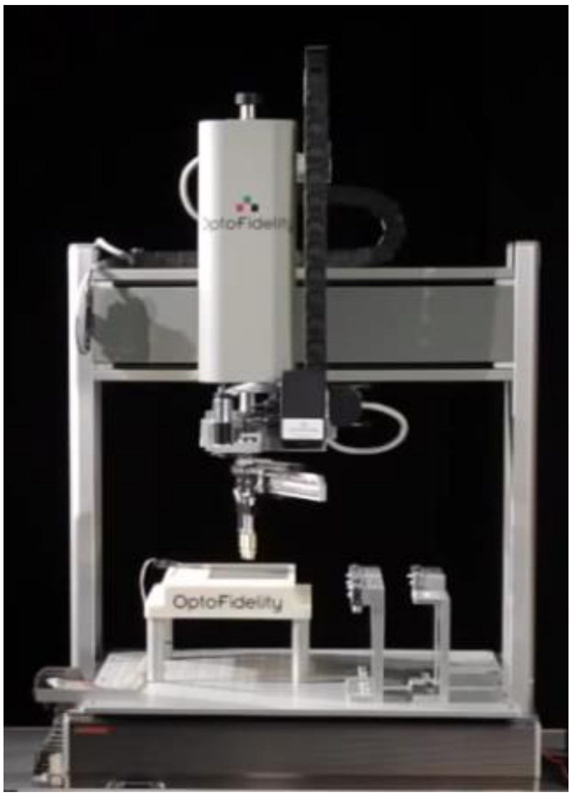

Figure 4. *OF-400 Cartesian robot [2]*

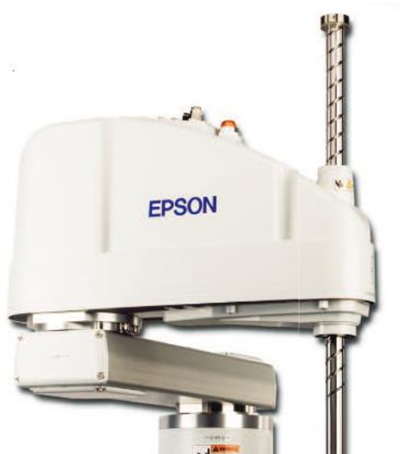

Figure 5. *Epson SCARA robot [18]*

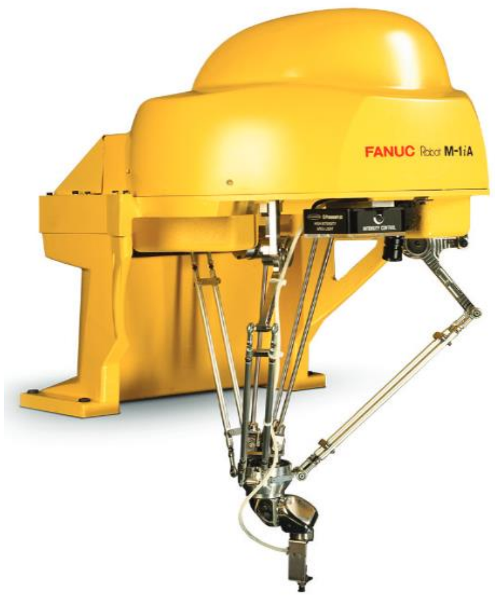

Figure 6. *Fanuc M-1ia Delta robot [17]*

Each robot type has its own inherent advantages and disadvantages regarding qualities such as maximum payload, speed, accuracy, repeatability, and work space size and shape. Choosing a suitable robot depends strongly on the task at hand.

2.3 Basic Kinematics

Industrial robots utilize several different coordinate systems, or frames. The *base frame* or *world frame* is a fixed frame where the origin is typically located inside the robot base plate. The *tool frame* is a coordinate system attached to the tip of the robot end-effector. Other examples of commonly used coordinate frames are shown in Figure *7*.

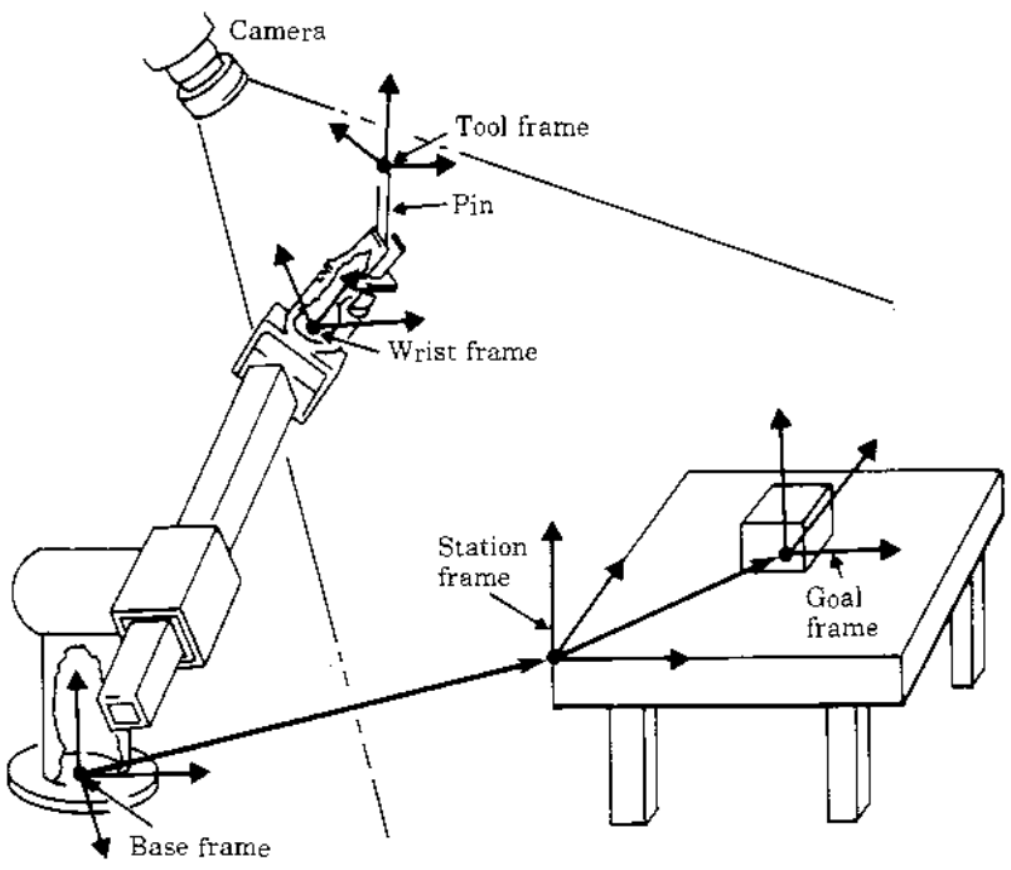

Figure 7. *Common robot coordinate frames [11]*

Practically any real-world task performed with a robot involves moving the robot tool frame, or *Tool Center Point* (TCP), through a defined path of points. Final goal of the task itself may be painting, welding, or testing the screen of a smartphone, but for the robot to be usable for any such real work, three fundamental problems have to be solved.

Firstly, to be able to move the robot TCP to a specific Cartesian point, we have to be able to determine the Cartesian position of the TCP at any given time. The robot joint positions are measured with encoders for each joint independently. These joint positions along with information about the robot geometry can be used to calculate where the TCP is located at in the Cartesian space. This process is called *forward kinematics*. For example, calculation of forward kinematics for the six axis robot in Figure *3* on page 5, would take into account angles of each joint as well as the known length of each link, and output the TCP position relative to the base frame. Forward kinematics for serial link robots is well-known and can be computed with a single homogenous transformation matrix [15].

Secondly, to be able to drive the robot TCP into a certain Cartesian point, there has to be a way to compute all possible joint angles that would place the TCP in this target point. This process is called *inverse kinematics*. Solving the inverse kinematics problem is notably harder than forward kinematics, since a closed-form solution cannot necessarily be acquired. If there exists no solution for inverse kinematics, the target point is located outside the robot work space [13].

The third problem is related to the speed at which the robot TCP is moving. If one wants to replicate human handwriting with a robot, it should be possible to control how fast the pen is moving. This problem is commonly solved by acquiring the so-called Jacobian matrix for the robot [13]. Jacobian transforms robot joint speeds into speeds in the Cartesian space. By solving inverse of the Jacobian it is possible to solve required joint speeds to achieve a desired TCP speed in Cartesian space. If the Jacobian for a point cannot be inverted, the point is said to be a singular point for the robot. In a singular point, the speed and movement of the robot may become unpredictable.

2.4 Motion Control

This section firstly gives a general overview to robot motion control. Two common industrial robot motion control schemes are then reviewed. Commercial industrial robots generally support at least one of these two ways of controlling robot motion.

2.4.1 Overview

Motion control of industrial robots is performed, at the low level, by using feedback controllers to control the voltage or current fed to each joint motor. Joint position information read from joint encoders is used to close the feedback control loop. Each joint is typically controlled by an individual controller, but more sophisticated control schemes also exist [13].

The target position can be specified either in joint or Cartesian space. If a Cartesian position is used, inverse kinematics has to be initially calculated to convert the position into joint space. Figure 8 shows an overview of the control scheme. Initial Cartesian target position x_d is converted into joint space representation q_d by calculating inverse kinematics. The controller uses feedback signal from the joint encoder to generate a control signal τ for the joint motor.

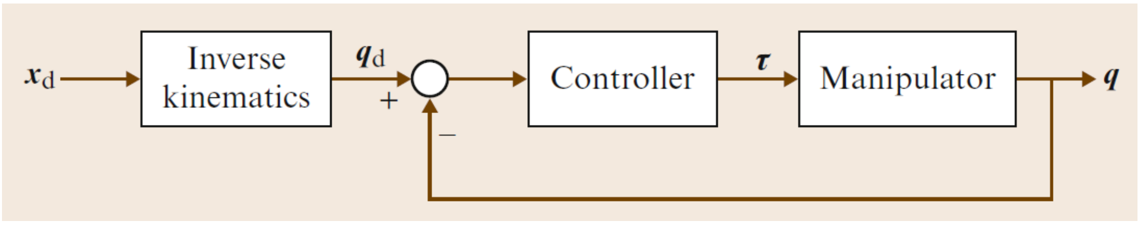

Figure 8. *General overview of robot joint control [14]*

PD and PID controllers have traditionally been used to implement the independent joint controllers. Their greatest advantage over more sophisticated controllers is simplicity in design and parameter tuning, which is often preferred in industry [14].

The end-user of a commercial robot is often not allowed to directly control the currents or voltages fed to the joint motors. In fact, according to [24], 90% of current robots do not offer this possibility to the user. Robot manufacturers typically implement their own proprietary low-level joint controllers and only offer the option for high-level position control for the end-user. This means that the user is merely allowed to input a set of points for the robot, along with desired speed values for each point. The internal control system of the robot then takes care of driving the robot through these points, while attempting to reach the desired speed at each point. Because each joint motor has an upper limit for the torque it can produce, and each link of the robot serial chain has a certain mass, each joint will have a finite maximum acceleration. Therefore, there is no guarantee that the robot will be able to reach the speed values along the trajectory.

There are two position control schemes commonly offered by industrial robots. The first one can be described as joint space control, and the second one operational space control [14]. These methods are discussed in sections 2.4.2 and 2.4.3, respectively.

2.4.2 Point to Point Motion

In *Point to Point* (PTP) motion the user simply specifies a target joint configuration into which he wants to drive the robot. This configuration represents the desired TCP end position in Cartesian space. The low-level joint controllers of the robot then take care of driving each joint into this exact state. Joints might be moved one by one, or all joints might start and stop moving at the same time, meaning that the joint with the longest distance to travel will move at maximum default speed, whereas the other joints will move at a slower speed [13].

The kind of joint space movement where joints move at different speeds is called linear joint interpolation. In a Cartesian robot such as the one in Figure *4* on page 6, linear joint interpolation will also be linear in Cartesian space. However, when controlling more complicated robot configurations with PTP motion, it is difficult to predict the robot path in Cartesian space. This is why PTP motion is mostly used for tasks such as spot welding, where it is sufficient that the correct target position is reached, and it is not necessary to precisely control the TCP position and orientation during the path.

On the other hand, benefits of PTP motion include fast and easy computation regarding joint control. This is due to the fact that since there are no requirements for the resulting Cartesian path, each joint can move at constant velocity (except for acceleration and deceleration zones close to end points). This means that there is no need to calculate inverse kinematics or Jacobians along the path.

2.4.3 Continuous Path Motion

Continuous Path (CP) motion is used in tasks that require the ability to specify the full path (position, orientation and speed) of the robot end-effector. Arc welding along a defined path is an example of a task like this [13]. While doing this type of welding, it is important that the robot follows the given path, and that the end-effector does not stay in contact for too long or too short of a time with each point.

Two types of CP motion that are commonly supported by commercial robots are linear and circular interpolation. Linear interpolation simply means that the robot TCP follows a linear path in Cartesian space between the start and end point. In circular interpolation the user specifies start, middle, and end points for the movement. These points are used to define a circle, along which the robot will move.

For a non-Cartesian robot, linear and circular-interpolated CP motion is notably more difficult than PTP motion, because in CP motion the speed of each joint has to vary along the path. This requires calculating inverse kinematics at each controller time interval [13]. While interpolating a path in the Cartesian space, the robot may also run into singularity points. Depending on the path and robot configuration, it might be very difficult to predict if a CP interpolated motion from one point to another will result in singularity at some point of the path.

2.4.4 Motion Parameters

Many industrial robots include parameters that can be specified to modify the motion in some way. Common parameters include robot speed and acceleration that can be specified for each point given to the robot. Speed parameter is commonly specified in millimeters per second for CP motion, and degrees per second for PTP motion. Acceleration is commonly given as percentage of maximum total acceleration, but other conventions also exist. Third commonly used parameter is related to smoothing the robot path. This may be specified, for example, as percentage of maximum smoothing, or maximum drift from the original path points. Demonstration of path smoothing can be seen in Figure *9*.

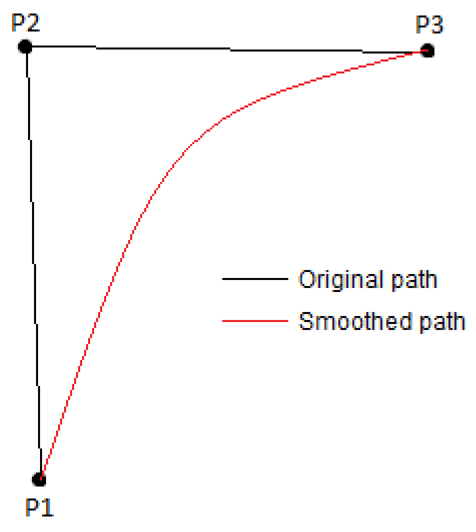

Figure 9. *Effect of path smoothing*

In addition to the value of the smoothing parameter, the amount of smoothing that the robot performs might depend on the speed at which the robot is moving. Therefore, it is not easy to predict the exact smoothed path that the robot will take between points.

2.5 Fanuc M-1ia Robot

Fanuc M-1ia industrial robot illustrated in Figure *6* on page 7, was chosen as the robot platform for this project. The robot combines traditional three axis delta-style structure with a three axis wrist. This kind of a six axis structure is rarely seen in delta-style robots. In fact, Fanuc is the only manufacturer that was found to be making six axis delta robots. Six degrees of freedom make it possible to control not just the location but also orientation of the robot TCP. This is useful in handwriting experiments, since the pen tilt angle could also be tracked.

The M-1ia robot has maximum payload of 0.5kg, and it has a cylinder-shaped work area with 200mm diameter and 100mm height. Each wrist joints have a maximum speed of 1400 degrees/s and movement repeatability is rated at 0.02mm [17]. The robot comes with Fanuc teach pendant and R-30iB Mate controller. R-30iB controller supports both PTP and CP motion. CP control option offers linear interpolation and circular arc interpolation capabilities. Speed, acceleration and CNT values can be specified for each motion point. CNT value corresponds to robot path smoothing, and its values are in range 0 – 100%.

The system was equipped with a custom-made stylus holder that uses a spring to allow suspension in one direction perpendicular to the *device under test* (DUT) screen surface. The somewhat linear nature of the spring-induced force could also be used as a primitive open-loop force control method. To achieve this, a high accuracy scale could be used to map spring displacement in millimeters with produced force in grams.

3. HANDWRITING

3.1 Use Cases

This section discusses some of the use cases there could exist for robotic handwriting. The use cases discussed here are kept within OptoFidelity's company focus, i.e. related to non-intrusive device testing.

3.1.1 Panel Accuracy

Touch panel accuracy is an important part of the user experience. Accuracy for individual gestures such as taps and swipes can already be tested e.g. with the OptoFidelity *Touch Panel Performance Testing* (TPPT) products [2]. However, accuracy of the panel may depend on the velocity and acceleration that the pen or finger is moving at. For determining the panel accuracy during handwriting, a realistic handwriting input has to be produced.

3.1.2 Panel Latency

Latency is another very important factor regarding touch-device user experience. Researchers from Microsoft Research Group and University of Toronto built a system that could demonstrate various touch latencies varying from 1ms to 100ms [4]. They concluded that while current devices typically offer latencies in the 50ms – 100ms range, users could clearly perceive latencies well below 10ms. Touch-screen latencies can be measured e.g. with the OptoFidelity Watchdog product [3]. While it may be that the device latency does not differ between the case where a straight line is drawn to the screen or where handwriting input is used, measuring handwriting latency would still make an interesting additional test case.

3.1.3 Effectiveness of Handwriting Recognition Algorithms

Handwriting recognition algorithms can be divided into off-line and on-line algorithms [5]. Off-line algorithms take a picture of a written text as input. The text is then preprocessed by gray scaling, removing noise, and segmenting characters and words. Different features are extracted from the image and used for optical character recognition or character recognition.

On-line algorithms utilize a time-series of the writing features, i.e. they take inputs such as X and Y coordinates, pen pressure, and so on, as a function of time. Due to gathering input data while the user is writing, on-line algorithms can generally perform better than their off-line counterparts. Approaches based on statistics and machine learning are widely used with on-line recognition algorithms. These approaches include probabilistic classifiers, Neural Networks, Hidden Markov Models, and many more [5].

Some of the current devices, such as the Microsoft Surface Pro series, offer native on-line handwriting recognition capabilities [6]. Additionally, there are several companies that offer third-party applications and libraries with on-line and off-line recognition capabilities [7], [8]. By gathering a database of handwriting samples and robotically reproducing them on a device, algorithm and application developers could test the performance both for their on-line and off-line algorithms. This would provide a non-intrusive test case where all the errors due to the device are taken into account, but that is also automated and repeatable.

3.2 System Requirements

Use cases introduced in the previous section can be used to define a set of requirements for the performance of the handwriting robotic system. Requirements for the system performance can be divided into three parts:

1. The robot system can replicate the trajectory (X and Y coordinates) of the handwriting sample with high enough accuracy
2. The system can replicate the temporal speed profile of the sample with high enough accuracy
3. The system can replicate the pressure and pen tilt profile of the sample with high enough accuracy

For measuring off-line handwriting recognition algorithm performance, it is enough that the first requirement is fulfilled.

For measuring worst-case panel accuracies and latencies it is generally enough that the first requirement is fulfilled, and that the second requirement is fulfilled in a way that at each point of the trajectory the robot velocity is equal to or greater than the pen velocity during the sample. If we assume that pen tilt or pressure has an effect on panel accuracy or latency, also the third requirement has to be fulfilled.

For measuring on-line handwriting recognition algorithm performance, at least the first two requirements should be fulfilled. If the algorithm also takes into account pen tilt and pressure, also the third requirement has to be fulfilled.

It is very difficult to come up with a single number for the “high enough accuracy” mentioned above. As with all testing and measurement systems, the required level of accuracy depends on the test case, e.g. desired accuracy of the tested touch-screen technology itself.

3.3 Handwriting Replication

Digital pen tablets made by Wacom were utilized in this project to record realistic handwriting samples that could be reproduced with the robot. The utilized Wacom tablets are able to record a time series of pen coordinates, pressure, and tilt values during the handwriting process. As described in section 3.2, ideally it should be possible to replicate any sampled handwriting trajectory with high enough accuracy regarding both spatial and temporal qualities. In practice, this means that:

1. The robot TCP should reach each coordinate, pressure, and tilt value included in the time series within specified accuracy and in the correct order
2. The robot TCP should reach the correct speed at each point within specified accuracy

These requirements essentially form a three-objective control problem, where one has to control position, axial force, and speed of the robot TCP, all at the same time. It turns out that in reality, it is very difficult to solve all these control tasks at the same time with commercial robots. Each of the three sub problems are discussed in the following sections.

3.4 Position Control

Since position control is already offered by most of the industrial robots, it is clearly the easiest of the three control cases for the robot programmer. Fanuc robot used in this project accepts coordinates in the Cartesian XYZWPR form, where W, P, and R correspond to rotation around X, Y, and Z axis, respectively. X, Y, and Z are specified in millimeters, whereas W, P, and R are specified in degrees.

Points recorded from Wacom tablets include XYZ coordinates, as well as pen tilt angle, which is specified as two parameters called azimuth and altitude. Raw Wacom X and Y coordinates are in pixels and they can be converted to millimeters by using the *lines per inch* (LPI) value of the tablet. Raw Z coordinates are discrete values in range 0 – 1024. They can be converted to millimeters by attaching the Wacom stylus to a robot and mapping robot Z movement in millimeters with the raw Z coordinate.

Lastly, Wacom azimuth and altitude values must be converted to Fanuc WPR values. Because the context of this thesis is testing handwriting on touchscreen devices, the operating space is effectively a two-dimensional plane that can only be touched from the above. Because of this restriction only two of the WPR coordinates are required to reach any pen angle with the robot: R coordinate, and either one of the W and P coordinates. If the robot coordinate frame is positioned so that each coordinate axis is parallel to the corresponding Wacom coordinate frame axis, then the conversion is simply:

$$\begin{cases} W = azimuth \\ P = 0 \\ R = altitude \end{cases}$$

After Wacom values are converted to XYZWPR they can be transferred to the robot. The robot controller will take care of driving the robot TCP through each point.

3.5 Force Control

It is possible to implement a primitive open-loop force control similar to the one in [9]. The robot tool holder would need to be equipped with a linear spring, after which the robot Z coordinate could be mapped with resulting force using e.g. a precision scale. However, this approach has some limitations. First of all, the device under test should be positioned in a very precise manner, i.e. no corner of the screen would be higher than the other. Secondly, mechanical imperfections along with the lack of feedback control will likely limit the accuracy of the setup.

3.6 Controlling Speed

3.6.1 Challenges

As it was explained in section 2.4.1, most commercial robots have low-level joint controllers that cannot be directly accessed by the end user. The end user is only allowed to access the high-level position control loop. While it is certainly possible to take the series of sampled handwriting points and speeds and input them directly to the robot position controller, there is no guarantee that the robot will actually reach the points at desired speed.

In fact, it quickly became evident that this approach results in a very slow motion of the robot. This happens mainly because the typical series of points sampled with the Wacom tablet is very dense, i.e. there are many points per written character. Because controlling the robot in Cartesian path mode requires heavy run-time inverse kinematics computation by the low-level robot controller, the robot controller’s look-ahead control might not have the capacity to take into account the huge amount of sample points in a single stroke. Past research efforts show that one way to increase the robot TCP speed in this case is to use Point to Point joint-space control [31]. This means that to ease the computational load on the robot controller the robot should be run in Point to Point mode and all points should be streamed to the robot in joint coordinates. This approach was also experimented with the Fanuc robot. However, in this case, it did not provide any speed benefits over using Cartesian motion.

Another approach to increase the robot speed while following a handwriting path is to simply reduce the number of points by approximating them so that the handwriting path would become less dense. This should increase the robot speed due to decreased load on the movement controller and its look-ahead logic.

It should be noted that even if successful, this approach will not lead to perfect speed tracking capabilities. This is due to the fact that reducing points also reduces number of possible points where robot speed can be altered. If the sampled handwriting trajectory contains speed variations in high frequency, it might be the case that after approximations the new path has too few points to be able to communicate all original speed variations to the robot.

Achieving highly accurate speed trajectory tracking of arbitrary speed curves with high-frequency variation would most likely require an altogether different kind of control method. The high-level position control that is used here would have to be replaced with a closed-loop controller. This is impossible with the closed structure of most commercial robot controllers. Instead, decreasing the amount of points primarily aims to increase the robot speed during handwriting, while sacrificing slightly regarding actual speed tracking capability. While not perfect, this approach is still good enough for some of the use cases defined in section 3.1. For example, testing panel worst-case accuracy doesn’t require following the speed trajectory exactly as long as the robot is able to reach high enough speeds during each point in the path. This case would certainly be better than the original situation with extremely slow robot speed and thus no speed tracking capability at all. Ways to approximate handwriting trajectories with less points are discussed in sections 3.6.2 - 3.6.5.

3.6.2 Douglas–Peucker Algorithm

Douglas–Peucker algorithm is one of the most widely used line simplification algorithms [25]. It takes a sequence of points and tries to approximate them by connecting the first and last points with a line. Distance from the line is calculated for each intermediary point, and the largest distance is found. If this distance is smaller than the maximum allowed approximation error, the algorithm terminates. Otherwise, point with the largest distance is used to divide the points into two new segments, and the algorithm is recursively applied to both segments. The algorithm is not optimal, but has been found to usually produce the highest quality approximations when compared with many similar algorithms [25]. Worst-case running time for the algorithm is $O(n^2)$.

In practice, the algorithm works fairly well for approximating handwriting trajectories. Wacom tablets have a relatively high, constant sampling frequency. Consequently, the largest gains from using Douglas-Peucker were seen in writing segments that resembled straight lines, particularly ones that were also written in slow to medium speed. These kinds of segments tend to contain a large amount of unnecessary sample points that slow down the robot movement controller. Figure *10* shows raw sampled points from four samples of letter ‘i’. Figure *11* and Figure *12* show the same samples approximated with

DP algorithm using different error parameters. Using DP algorithm with 0.01mm error reduced the total amount of points by 80.8% removing 674 of the 827 original points. Using 0.1mm error reduced points by 95% removing 793 points. Discarding these points provided significant speed improvements.

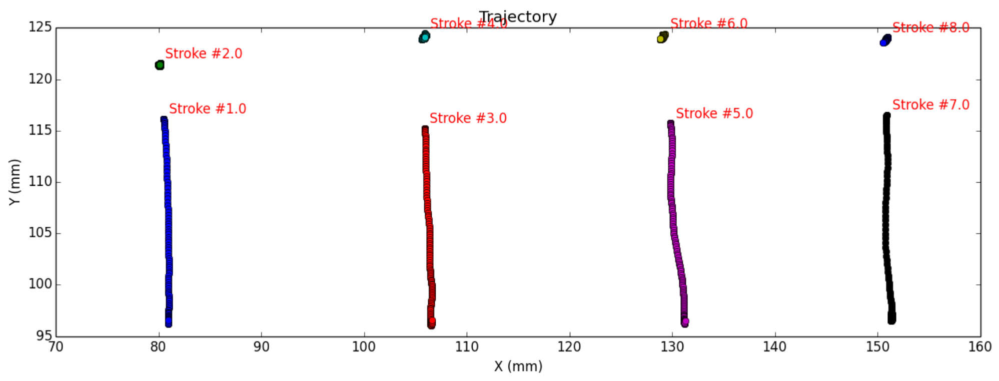

Figure 10: *Four samples of letter 'i'*

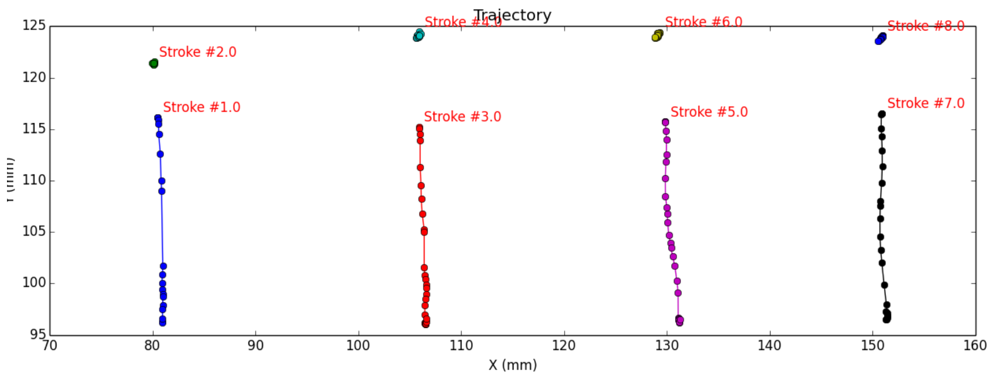

Figure 11: *Samples approximated with DP algorithm (error = 0.01mm)*

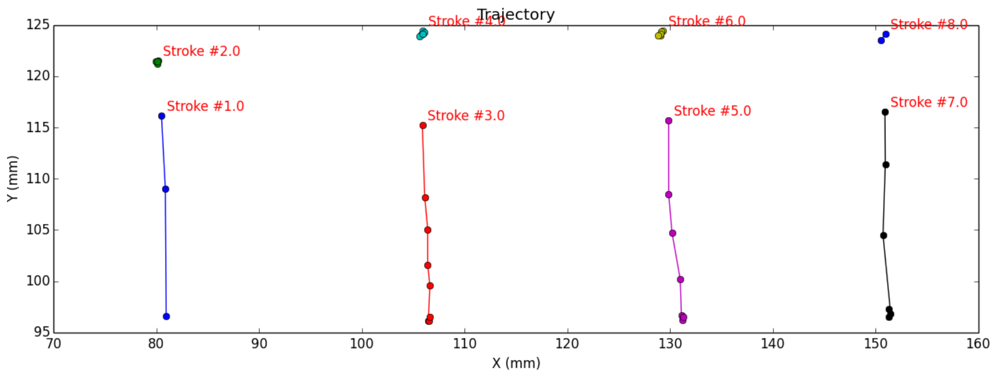

Figure 12: *Samples approximated with DP algorithm (error = 0.1mm)*

Segments with high amount of curvature, such as letters "s" and "o" were also affected, but not as much. This can be seen from Figure *13*, Figure *14*, and Figure *15*.

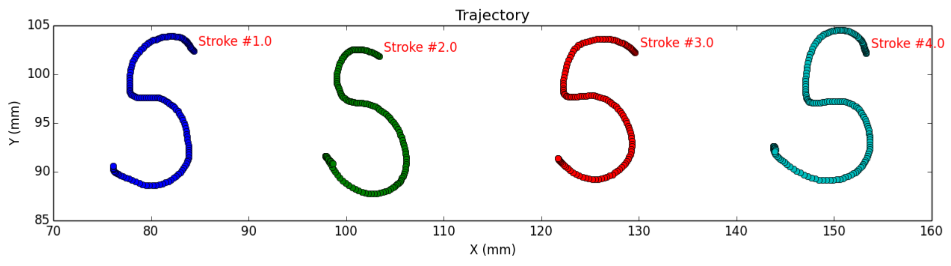

Figure 13: *Four samples of letter 's'*

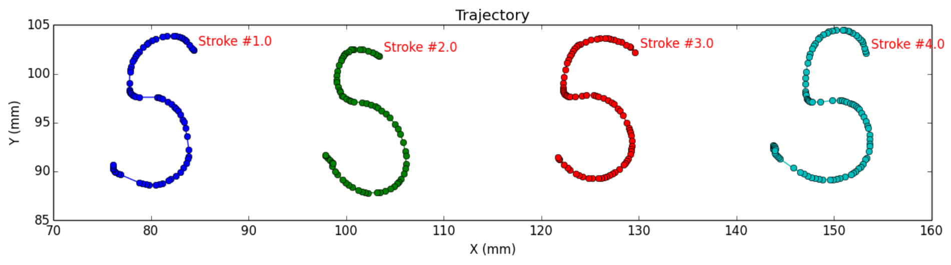

Figure 14: *Samples approximated with DP algorithm (error = 0.01mm)*

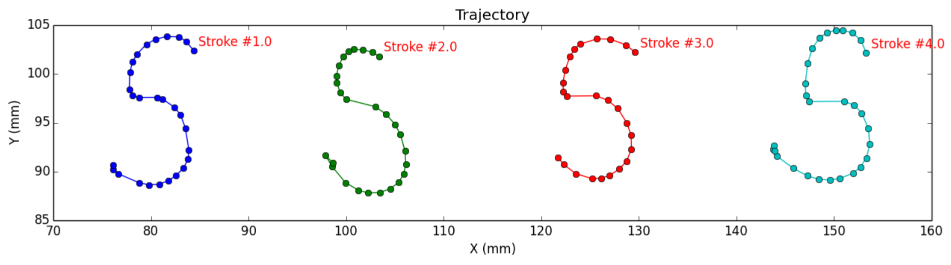

Figure 15: *Samples approximated with DP algorithm (error = 0.1mm)*

Here DP algorithm with 0.01mm error reduced the total amount of points by 45.2% removing 265 of the 583 original points. Using 0.1mm error reduced points by 81.4% removing 477 points.

3.6.3 Motivation for Using Circle Arcs

It would, of course, be ideal if the chosen approximation algorithm worked well on highly curved paths as well. As stated in section 2.4.3, many industrial robots support two types of Cartesian motion: linear interpolation and circular arc interpolation. This implies that even if it would be relatively easy to approximate paths with e.g. Bezier curves, this would not offer any performance benefits, since the robot is not capable of Bezier interpolation movement. On the other hand, if handwriting curves could be approximated with circular arcs and straight lines, this could significantly reduce the amount of atomic movement commands that have to be sent to the robot. As it was discussed in section 3.6.1, this could lighten the load on the robot movement controller, and lead to higher robot speed.

This approach was initially verified by approximating a perfect half circle with radius of approximately 79mm, with the DP algorithm, using 0.1mm approximation error. Resulting points were entered to the robot and ran in linear interpolation movement mode. This movement path is shown in Figure *16*.

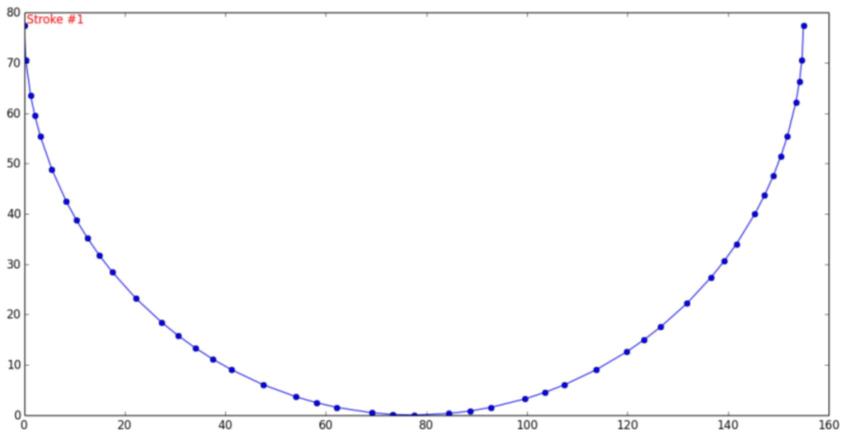

Figure 16: *Half circle approximated with DP (error = 0.1mm)*

The same path was then converted into a single circular arc movement command for the robot. The half circle was drawn using both approaches while varying robot speed and smoothing settings. 20 runs were performed with each setting, and movement durations were recorded in a PC-based Python script. Results are shown in Table *1*.

Table 1: *Average durations for drawing half circles with Fanuc M-1ia robot*

Speed (mm/s)	Smoothing (%)	Average duration (s) (linear commands)	Average duration (s) (circular arc command)
100	0	5.11	2.73
100	25	3.87	2.71
100	50	3.38	2.70
100	100	3.13	2.70
1000	0	3.32	0.54
1000	25	1.57	0.51
1000	50	1.26	0.51
1000	100	1.26	0.51

From the results it can be seen that while increasing the robot smoothing setting allows linear interpolation movements to reach higher speeds, circular interpolation movement is still faster in all cases. Using high smoothing setting also causes the robot TCP path to deviate from the original programmed path, whereas this did not appear to affect circular interpolation movement. While the PC environment and robot communication interface surely introduce some error in the measured durations, the system can be considered consistent enough to detect the kind of large relative differences seen in Table *1*.

After validating the idea, available literature on the subject was researched. It became evident that the problem of approximating arbitrary curves with circular arcs has been studied extensively in the past decades. Part of the reason for the vast research interest are commercial Numerical Control (NC) machines. NC machines are capable of understanding movement commands in language called G-code, which is also restricted to containing either straight line segments or circular arcs. Optimizing NC machine movements with circular arc approximation has a clear commercial application: reducing NC machine cycle times while preserving high path accuracy.

3.6.4 Simple Circular Arc Approximation Algorithm

Initially a simple algorithm similar to the one in [26] was implemented. Implementation process started by creating a program that can find the center point and radius of a circle that passes through three fixed points. Mathematics involved in this problem is well-known and a solution using perpendicular bisectors is introduced here. The solution is adapted from [26].

Assumption: P_1 (x_1, y_1), P_2 (x_2, y_2) and P_3 (x_3, y_3) are three collinear points.

Let A and B be center points of the two line segments drawn from point P_1 to P_2, and from point P_2 to P_3, and let C be the center point of the circle defined by these three points. Let L1 and L2 be the lines passing through P1, P2, and P2, P3, respectively. The setting is illustrated in Figure *17*.

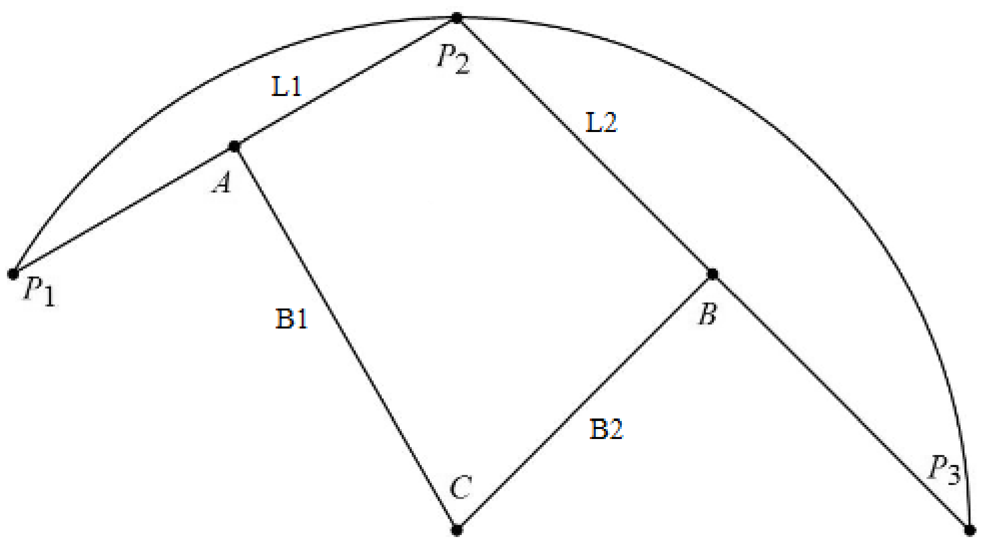

Figure 17: *Circle C, passing through points* $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, *and* $\boldsymbol{P_3}$ *[26]*

Circle center point C is the point where perpendicular bisectors of the two line segments intercept. We begin by finding the equations for these perpendicular bisectors.

Because points A and B are middle points of the two line segments, obviously coordinates for the points A and B can be written as

$$A = (x_a, y_a) \ = (\frac{x_2+x_1}{2}, \frac{y_2+y_1}{2})$$

and

$$B = (x_b, y_b) = (\frac{x_3+x_2}{2}, \frac{y_3+y_2}{2})$$

Here we recall that slope of a perpendicular line can be calculated as the negative reciprocal of the slope of the original line. Since lines L1 and L2 are perpendicular to B1 and B2, slopes for B1 and B2 can be written as

$$k_{B1} = \frac{x_1-x_2}{y_2-y_1}$$

and

$$k_{B2} = \frac{x_2-x_3}{y_3-y_2}$$

Because both B1 and B2 pass through point C, we can write a system of equations for the perpendicular bisectors as

$$\begin{cases} y_c - y_a - k_{B1}(x_c - x_a) = 0 \\ y_c - y_b - k_{B2}(x_c - x_b) = 0 \end{cases}$$

Solving for the center point x_c and y_c

$$x_c = \frac{\left(\frac{y_1}{2}-\frac{y_3}{2}+\frac{\left(\frac{x_1}{2}+\frac{x_2}{2}\right)*(x_1-x_2)}{y_1-y_2}-\frac{\left(\frac{x_2}{2}+\frac{x_3}{2}\right)*(x_2-x_3)}{y_2-y_3}\right)}{\frac{x_1-x_2}{y_1-y_2}-\frac{x_2-x_3}{y_2-y_3}}$$

$$y_c = \frac{\left(-x_1^2x_2+x_1^2x_3+x_1x_2^2-x_1x_3^2+x_1y_2^2-x_1y_3^2-x_2^2x_3+x_2x_3^2-x_2y_1^2+x_2y_3^2+x_3y_1^2-x_3y_2^2\right)}{2(x_1y_2-x_2y_1-x_1y_3+x_3y_1+x_2y_3-x_3y_2)}$$

When the center point is known, radius of the circle can be calculated using any of the three known points that are on the circle. For example, using P1:

$$r = \sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2}$$

Direction of the arc is equal to the sign of the following determinant [26]

$$\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{vmatrix}$$

The second mathematical concept needed in the algorithm is called Hausdorff distance. For an arc with radius r and center point (x_c, y_c), Hausdorff distance to an arbitrary point (x_n, y_n) can be calculated as [26]

$$d = \left|\sqrt{(x_n - x_c)^2 + (y_n - y_c)^2} - r\right|$$

The actual circular arc approximation algorithm accepts maximum allowed error ε as an input parameter. The algorithm begins by iterating through all pairs of path points P_i, P_j that have at least two middle points $P_m^k \in Q$, where $k = 0, 1, 2, \ldots$ and $\|Q\| \geq 2$. For each P_i, P_j, a set of intercepting circles are determined by using each of the middle points P_m^k as the third circle point, one at a time. For each circle, Hausdorff distance is then calculated between the circle arc and each middle point, and the largest distance is recorded.

Of this set of all possible circles passing through P_i, P_j, and P_m^k, the best circle arc is chosen, i.e. one with the smallest recorded distance. If this distance $d_{ijk} \leq \varepsilon$, the corresponding circle arc is inserted into the set of accepted arcs.

After constructing the set of best arcs between any three points on the path, the final task is to find the subset of arcs that lead from the path start point to the path end point with the least amount of arcs possible. This task can be formulated as a graph search problem. Each path point is considered to be a graph vertex, and the end points of each arc determine edges connecting vertices together. The final approximation result is obtained by finding the shortest path between start and end vertices of the graph.

One major drawback of the implemented algorithm is its time complexity. Finding all possible circles in a way described above results in running time of $O(n^3)$. Consequently, to be able to get results in a reasonable time from a large amount of sample points, the raw points were first pruned with DP algorithm introduced in the previous section, and then approximated with the circular arc algorithm. Four approximations of letter ‘s’ were first approximated with DP algorithm using 0.1 mm error, and then with the circular arc algorithm using 0.02 mm error. Results are illustrated in Figure *18*. Green lines designate parts where an arc couldn’t be fitted and a line is used instead.

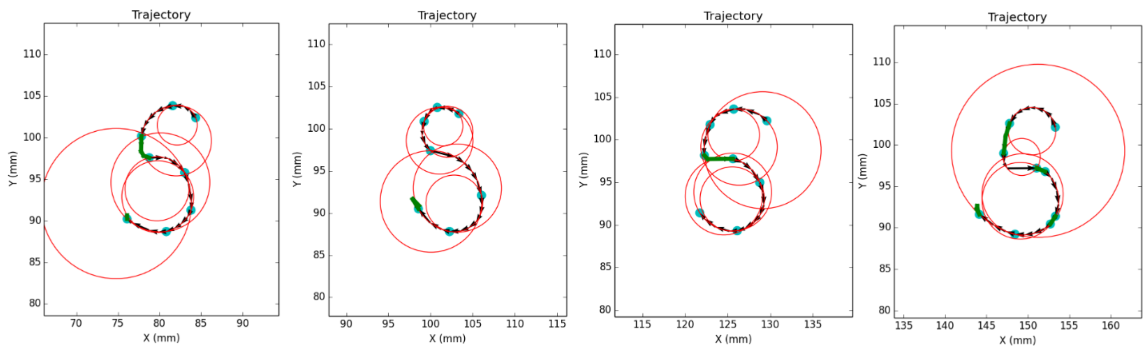

Figure 18: *Samples of letter 's' approximated with circular arcs*

Comparing to Figure *15*, it can be seen that arc approximation clearly reduces the required amount of points. For example, in the second sample of Figure *18* the path can be represented by 6 circle arcs and one line segment, which corresponds to 7 move commands for the robot. In Figure *15* the second sample contains 27 points (line segments), which require as many move commands for the robot.

However, the second major drawback of the algorithm is the fact that two consequent arcs produced by the algorithm are often not G^1 continuous. This means that tangents of the arcs are pointing in different direction in the collision point, i.e. the path is not smooth and it has sharp corners. Initially it was hoped that increasing the robot smoothing parameter would smooth out these collision points, but practical tests showed that this is not the case. The resulted writing path contained non-smooth points even when smoothing was set to 100%.

3.6.5 Biarc Approximation

To overcome the problem of non-smooth curves produced by the simple circular arc approximation, the paths could be approximated with G^1 continuous curves, such as biarcs or smooth arc splines. A biarc includes two circular arcs which have equal tangents in their common point [28]. Smooth arc splines consist of circular arcs and line segments so that tangents in the colliding points are equal [30].

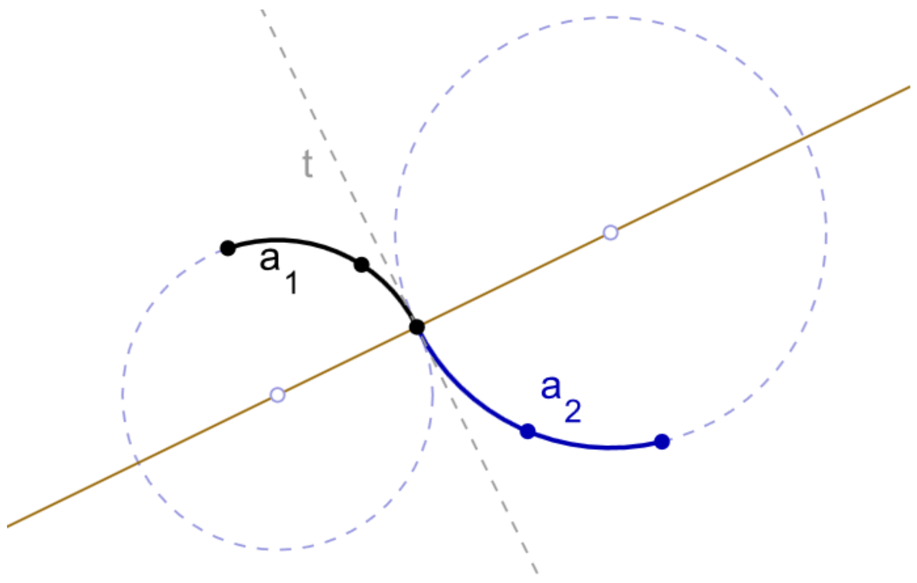

Figure 19: *A single biarc curve [29]*

Approximating arbitrary curves with biarcs has been an active research topic during recent years. Some efforts use an approach where a biarc is fitted between two points on the path, and the tangent for each point is calculated from the line that connects it with the next point. The downside of this approach is that since the curve breakpoints and tangent data have to be selected from a finite set of points, and thus the results are often not optimal [30]. Other promising approaches include the Smooth Minimum Arc Path (SMAP) algorithm developed in University of Passau [30]. These algorithms were not implemented in the scope of this project, but they would make an interesting topic for future work regarding the subject.

4. IMPLEMENTATION

4.1 Constraints

For this project, a custom 3D-printed stylus holder was manufactured and fitted to the Fanuc robot. The holder includes a spring that allows the attached stylus to yield in the Z axis direction when the pen is pressed against a surface. The holder also houses a camera that can be used in positioning the device under test in the robot work area.

Spring-based open-loop force control, as introduced in section 3.5, was not implemented during this project. The same applies for pen tilt control. For the force control to work, the holder would need to flex in a linear manner along the Z axis direction, while controlling the tilt could change the required direction of flex at any given time. To achieve this, more attention would have to be given for the mechanical design of the stylus holder. For these reasons, main focus was given to speed and position control of the robot.

4.2 Architecture Overview

The implemented software is run in regular Windows PC environment. Communication between the PC and robot was implemented using Fanuc robot interface, which is a proprietary Ethernet-based COM interface available as an extra option for most Fanuc robots. Communication with the robot consists of two elements:

1. A PC program used to convert Cartesian points into robot movement commands from PC to the robot controller.
2. A program running in the robot controller that receives these commands and controls the robot accordingly.

All handwriting-related software was separated from the communication interface. This software consists of two main parts:

1. A recording application that interfaces with Wacom tablets and provides a canvas for the user to draw in, ability to collect points from the Wacom tablet, and save the points in a pre-defined CSV format.
2. Software package that is able to parse saved points, calculate speed and acceleration values, draw attribute graphs, run various approximations on the set of points, and finally create a complete writing sequence including start and stop periods and pen lifts between strokes.

High-level illustration of the software architecture is shown in Figure *20*.

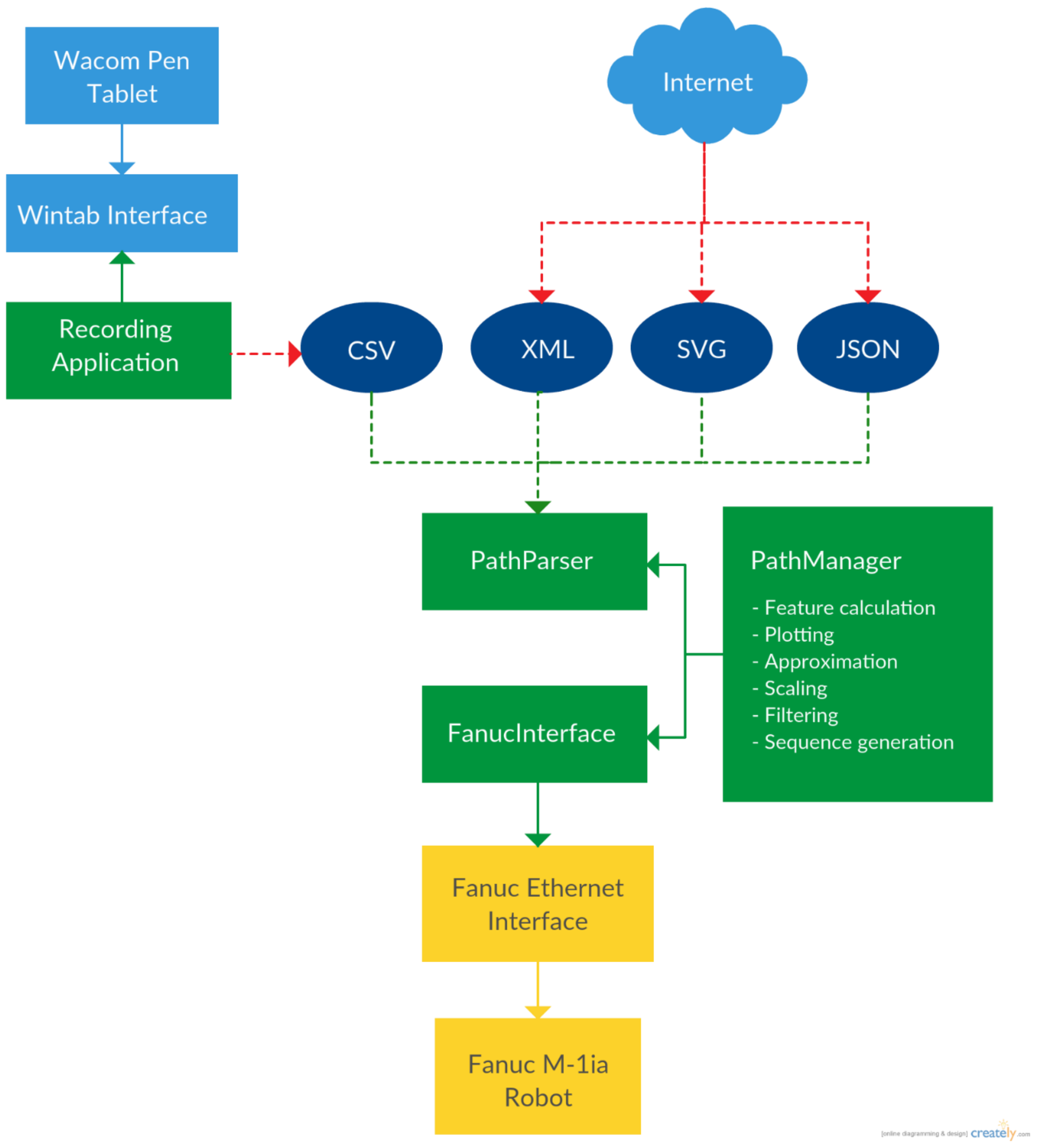

Figure 20: *Overview of System Architecture*

All green elements of the diagram were implemented within the scope of this project in Python. Python was chosen mainly due to its wide use within the company and extensive amount of available of libraries. This will make integration with e.g. OptoFidelity Touch and Test (TnT) platform easy in the future.

4.3 Working Demo for MWC 2016

Demo version of the system was showcased in Mobile World Congress 2016, which was held in Barcelona. Exhibition guests were given a chance to write a text using Wacom

Cintiq tablet. The text was then replicated by the Fanuc robot. Figure *21* shows the final demo setup.

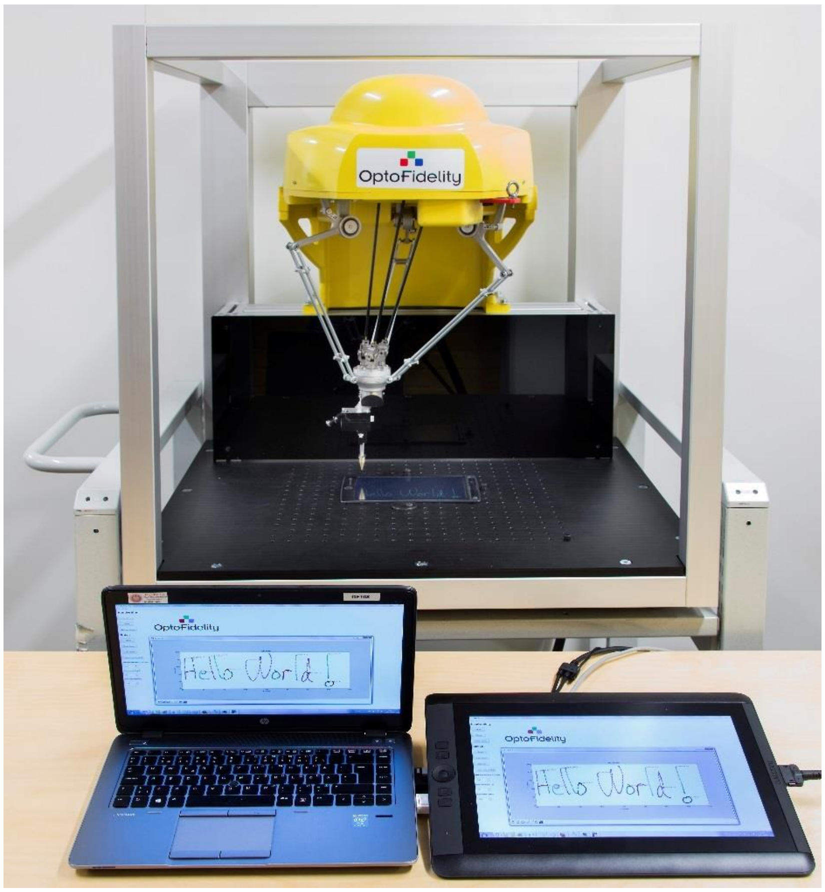

Figure 21: *Demo System for MWC Exhibition*

During the course of the exhibition, comments and use cases were gathered related to the system and handwriting in general. The demo generated a lot of interest, and valuable feedback was also received.

5. MEASUREMENTS AND RESULTS

5.1 Recording Real Handwriting

To obtain a sample dataset for testing, real handwriting was recorded from three test persons. A documented standard set of texts was used so that expanding the sample set would be easy in the future. In addition to obtaining test samples, the aim for this experiment was to acquire approximate figures on typical human handwriting speed. Following sub sections show the setting and results of this writing experiment.

5.1.1 Measurement Setting

The test persons were asked to write various text samples in English. Selected texts were:

1. The quick brown fox
2. jumps over the lazy dog
3. The
4. quick
5. brown
6. fox
7. jumps
8. over
9. the
10. lazy
11. dog
12. www.optofidelity.com
13. 1234567890
14. john.smith@gmail.com
15. The writer's *own name*, written as fast as possible

The first two lines contain all letters of the English alphabet, and they are commonly used in writing measurements similar to this one. Apart from many similar experiments, however, the scope of this thesis is specifically writing on touchscreen devices. Touchscreen devices have a rather limited space for writing, which leads to sentences being commonly written in multiple consecutive parts. This is why in this experiment the single sentence is divided into two parts and both parts are written separately. Additionally, each word of the sentence is also written separately.

In addition to the sample sentence, a web address, e-mail address and string of numbers were selected, as they reflect different styles of writing. Writers were also asked to write their own name as fast as they can. This is done mainly to acquire an upper limit for the typical human handwriting speed. Writers were first given a chance to familiarize themselves with the equipment and the text samples. Then each text was written 3 – 5 times.

5.1.2 Utilized Equipment

Handwriting samples were gathered in digital form using two different devices made by Wacom. These devices were selected mainly due to their relatively high accuracy and sampling rate, high number of measured attributes, and good availability. Both devices are able to measure the following attributes of the input stylus:

- X and Y coordinates of the pen tip
- Z coordinate of the pen tip, i.e. how high the writer lifts his pen in between strokes (only up to a few centimeters)
- pen pressure
- pen tilt angle

Each measurement is decorated with a timestamp generated by the device hardware itself. Since the timestamps are not affected by host operating system or intermediary software, pen tip speed and acceleration can be reliably calculated using XY coordinates and timestamp values.

Both devices transmit pen data in packets using Wintab API, which can be used to capture the data packets in Windows environments [21]. A Python application communicating with the Wintab API was written for this project. The application includes:

- a drawing surface mapped with coordinates of the Wacom tablet
- ability to capture desired packets through Wintab interface
- ability to save the captured data in CSV format

The application can be used to capture handwriting or drawing with any device supporting the Wintab API.

The first device is Wacom Intuos Pro, which is a pen tablet that can be connected to a PC and used as a regular input device. The Intuos Pro does not contain a screen of its own. It uses Wacom's proprietary technology based on *electromagnetic resonance* (EMR) to sense the X and Y coordinates of the pen tip touching the tablet surface [20]. Due to the lack of an integrated screen the tablet was accompanied with a Wacom Inking Pen. The Inking Pen is essentially a combination of a regular pen and Wacom stylus. It allows the user to place a regular sheet of paper on the Intuos tablet, and while writing, ink is deposited on the paper and the pen movements are also recognized by the tablet. This creates a paper-like writing experience for the test persons.

The second device is Wacom Cintiq 13 HD. It utilizes similar EMR technology as Intuos, while also incorporating an LCD screen in the same device. This makes it possible for the user to get instantaneous feedback of his writing using the screen instead of paper. Writing experience between the two devices is different. While writing on Intuos is exactly like writing on normal paper, writing on Cintiq feels very much like writing on any touch-screen device with a stylus.

5.1.3 Results

Overview of the handwriting recording results are shown in Table *2* for the regular text samples 1 – 14, and in Table *3* for the signatures of each test person.

Table 2: *Calculated handwriting features (test items 1 - 14)*

	Intuos Pro	Cintiq HD
Average speed (mm / s)	42.0	57.2
Maximum speed (mm / s)	466.4	298.0
Average acceleration (mm / s^2)	90.2	268.2
Maximum acceleration (mm / s^2)	12690	16580
Average pen azimuth (degrees)	135.4	131.6
Average pen altitude (degrees)	56.5	63.1

Table 3: *Calculated handwriting features (test item 15)*

	Intuos Pro	Cintiq HD
Average speed (mm / s)	103.7	111.7
Maximum speed (mm / s)	769.0	550.6
Average acceleration (mm / s^2)	220.0	471.4
Maximum acceleration (mm / s^2)	38330	22630
Average pen azimuth (degrees)	138.4	136.2
Average pen altitude (degrees)	58.3	60.9

Surface of the Cintiq tablet feels slicker with less friction, and it was found out that users tended to write slightly faster with the Cintiq than with the Intuos. However, users also wrote slightly larger text on the Cintiq, which affects the speed and acceleration values in the tables above.

5.1.4 External Handwriting Sources

There exist several Internet databases that contain a large amount of writing samples by a large amount of writers, e.g. [22] and [23]. These databases are commonly used by researchers to test the performance of new handwriting recognition algorithms. These databases could also be used with the robot system developed in this thesis to test the end-to-end performance of the recognition algorithms. The samples in IAM database [22], for example, are stored in XML format. To be able to reproduce those samples with the robot system would merely require writing a simple converter between the XML format and the format understood by the robot software. This would be a fast and convenient way to acquire a huge set of test samples, but due to licensing restrictions it was not done in the scope of this project.

5.2 Measuring Robot Handwriting Performance

Handwriting measurements were carried out by writing samples that were collected earlier with the Intuos tablet. Wacom stylus was attached to the stylus holder, and Intuos tablet was placed under the robot work area. Writing data was recorded using the same recording software as with human writers.

Sections 5.2.1 – 5.2.3 show how the speed and accuracy of the robot is affected by approximating the sample points with less number of points, and by adjusting the smoothing parameter in different ways. Section 5.2.4 presents the final measurement results.

5.2.1 Effect of Path Approximation

To demonstrate the speed–accuracy tradeoff caused by path approximation, sample texts were approximated with Douglas–Peucker algorithm. Here, an example of one sample text is introduced. Figure *22* shows the writing sample without any approximation. Figure *23* and Figure *24* show the same writing approximated with 1% and 10% errors, respectively. Error parameters are calculated for each stroke individually, i.e. a 1% approximation of a stroke with a length of 10mm contains maximum of 0.1mm deviations of the original. All other settings such as robot speed and smoothing were kept identical in the three cases.

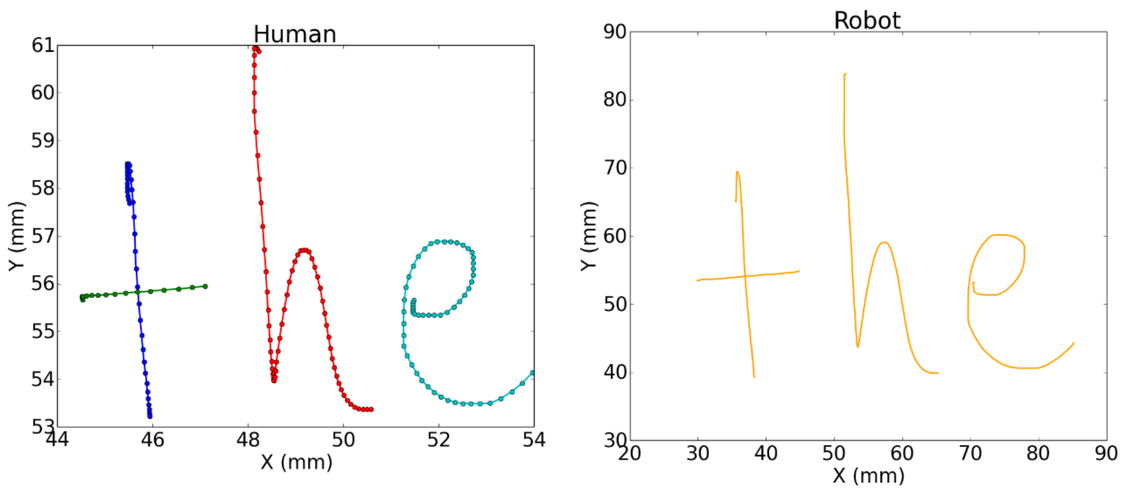

Figure 22: *Writing sample, max_error = 0 %*

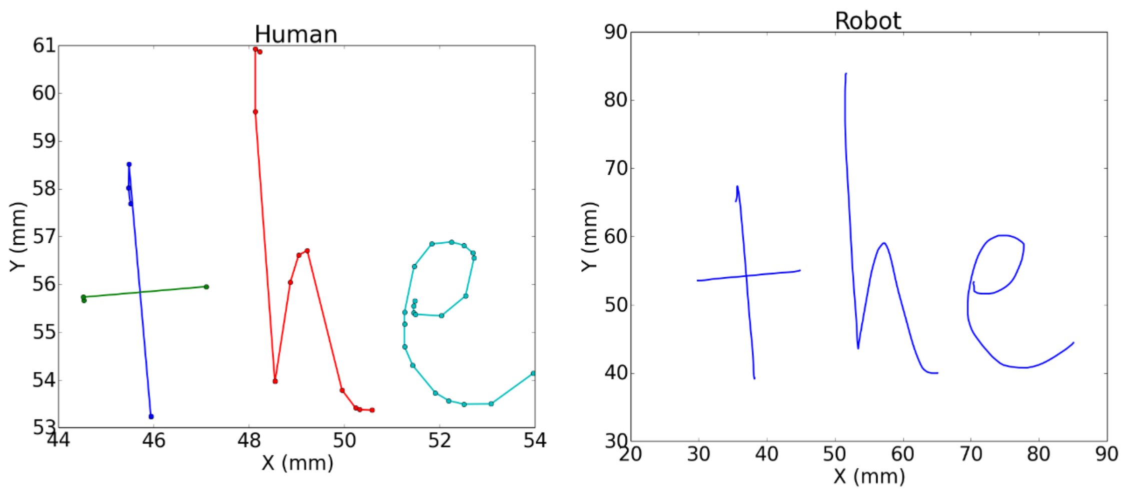

Figure 23: *Writing sample, max_error = 1 %*

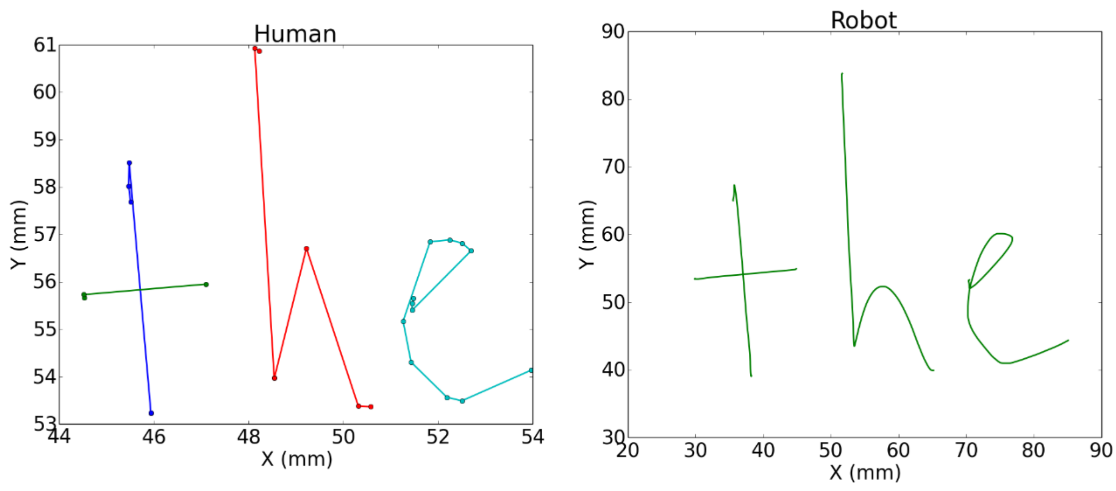

Figure 24: *Writing sample, max_error = 10 %*

As it can be seen from the pictures, the robot is capable of reproducing the writing path fairly accurately with lower approximation settings. It was discovered that in most cases, 1 % approximation is fairly close to the original sample, but in the end, it is up to the user to decide how close to the original he wants the sample to be.

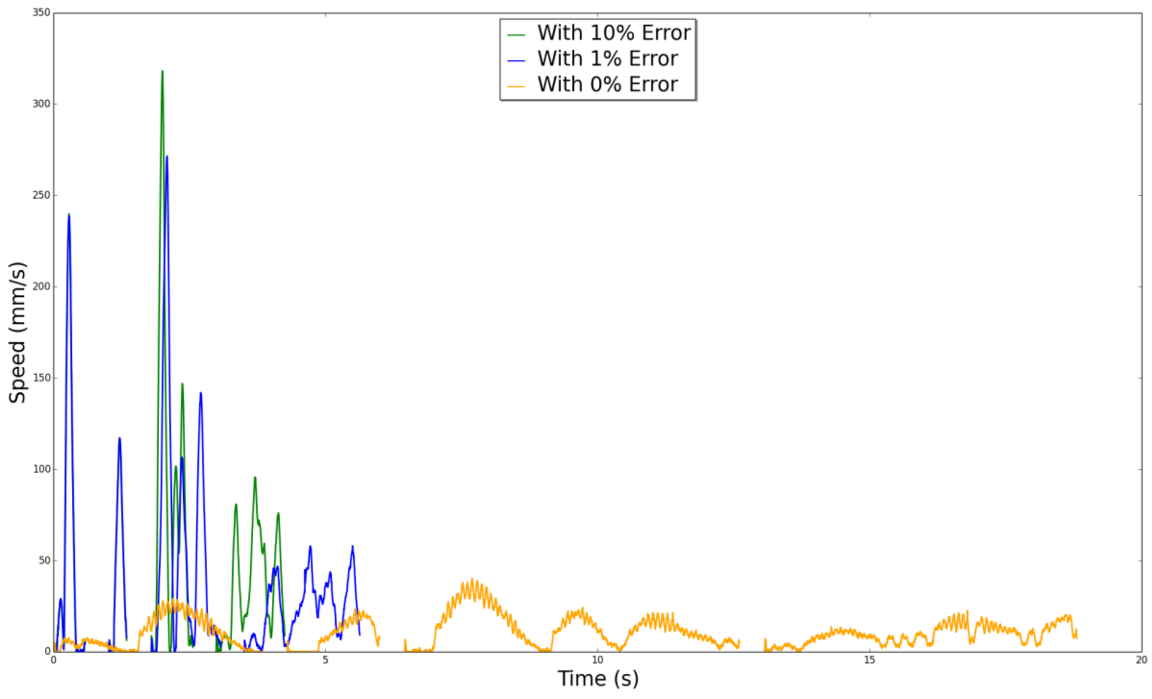

Figure 25: *Robot speed with varying approximations*

As it can be seen from Figure *25*, pruning the path points has a significant effect on the speed that the robot is able to reach during the path. Average and maximum speed values for the three cases are:

Approximation error (%)	Average speed (mm/s)	Maximum speed (mm/s)
0	10.4	40.1
1.0	40.7	271.4
10.0	53.5	318.1

During testing it was discovered that even though approximation with higher error is usually infeasible, some moderate amount of approximation almost always improves the robot speed. As it was discussed earlier, this effect is most clearly visible in relatively straight line segments where there are many “redundant” sample points.

5.2.2 Effect of Robot Smoothing

Another way to increase robot speed is allowing the robot to smooth out its movements. Figure *26* shows a writing sample written in different smoothing settings. 0%, 10%, and 50% smoothing settings were used. 0% and 10% settings produce a near-identical result, whereas 50% smoothing results in a clearly different trace.

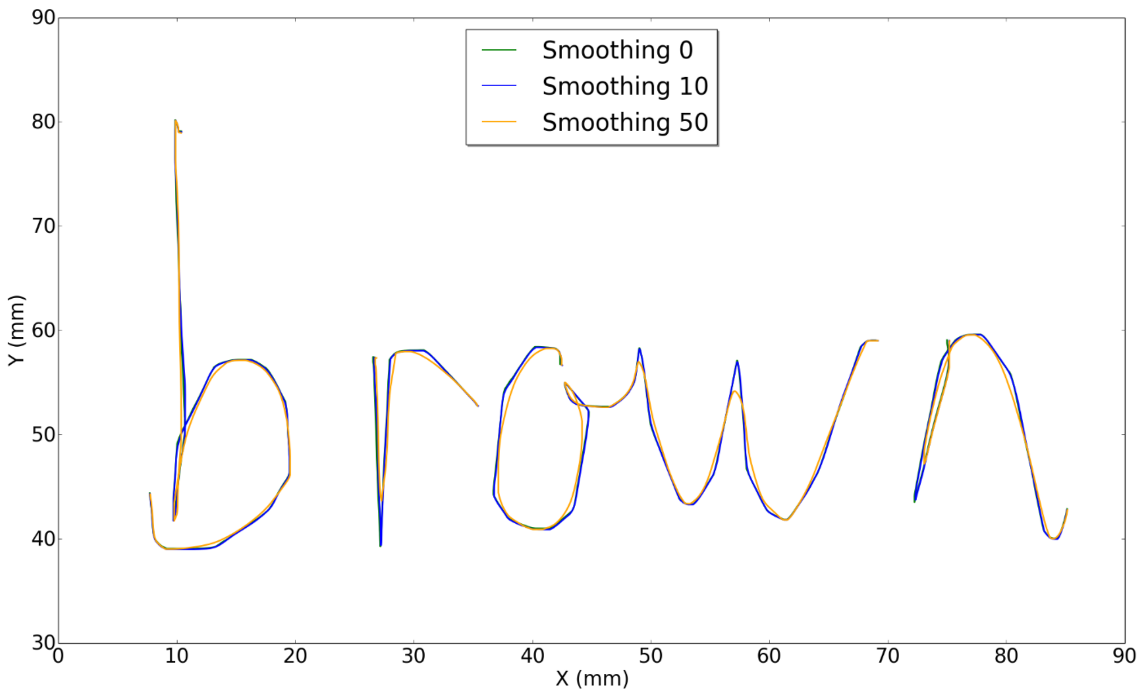

Figure 26: *Robot path with varying smoothing settings*

It was discovered that using some amount of smoothing is generally preferable, since otherwise the final robot path will often contain sharp edges. This is most clearly visible when a high amount of approximation is used, i.e. when there are few sample points for each character. On the other hand, in some cases, these kinds of sharp edges in the robot path are desired. E.g. the bottom points of letters “r” and “n” contain these kinds of edges, and the robot should be able to produce them. High smoothing may smooth out these desired edges as well, as can be seen from the orange line in Figure *26*.

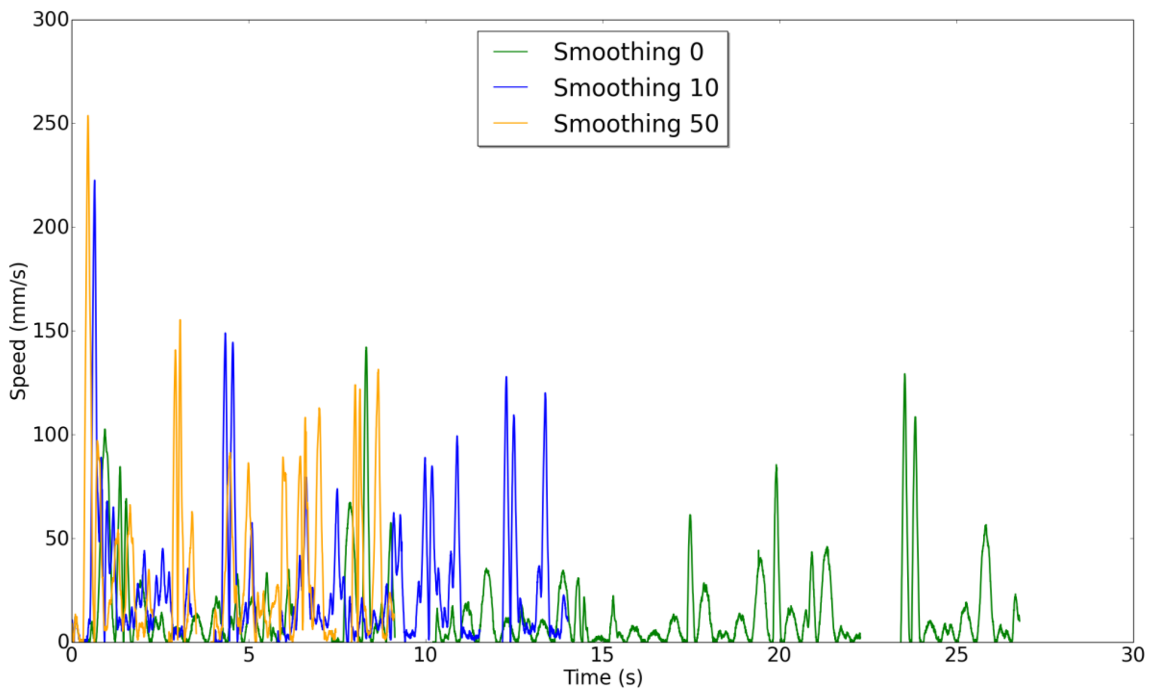

Figure 27: *Robot speed with varying smoothing settings*

Robot smoothing was discovered to have a significant effect on robot speed. This is illustrated in Figure *27*. Average and maximum speeds with different smoothing settings are:

Smoothing setting (%)	Average speed (mm/s)	Maximum speed (mm/s)
0	13.4	142.1
10	25.8	222.5
50	36.5	253.6

It was discovered that usually increasing smoothing from 0% to 10% does not have significant effect on the robot trajectory, but it has a strong effect on robot speed. High smoothing settings might significantly affect the robot final path as well.

5.2.3 Effect of Varying Smoothing

To overcome the problem of drawing sharp edges with high smoothing, it could be attempted to vary the smoothing setting for each sample point. Smoothing setting could be decreased for points that belong to a sharp edge section, whereas smoothing could be set high for points that are on a relatively straight line or within a smooth arc segment. It was discovered that these kinds of sharp edges usually occur in points where the human hand reverses the pen direction, e.g. in the bottom point of letter “r”. These reversal points can be found by considering the local minima of the writing speed trajectory, and smoothing can be decreased in these points.

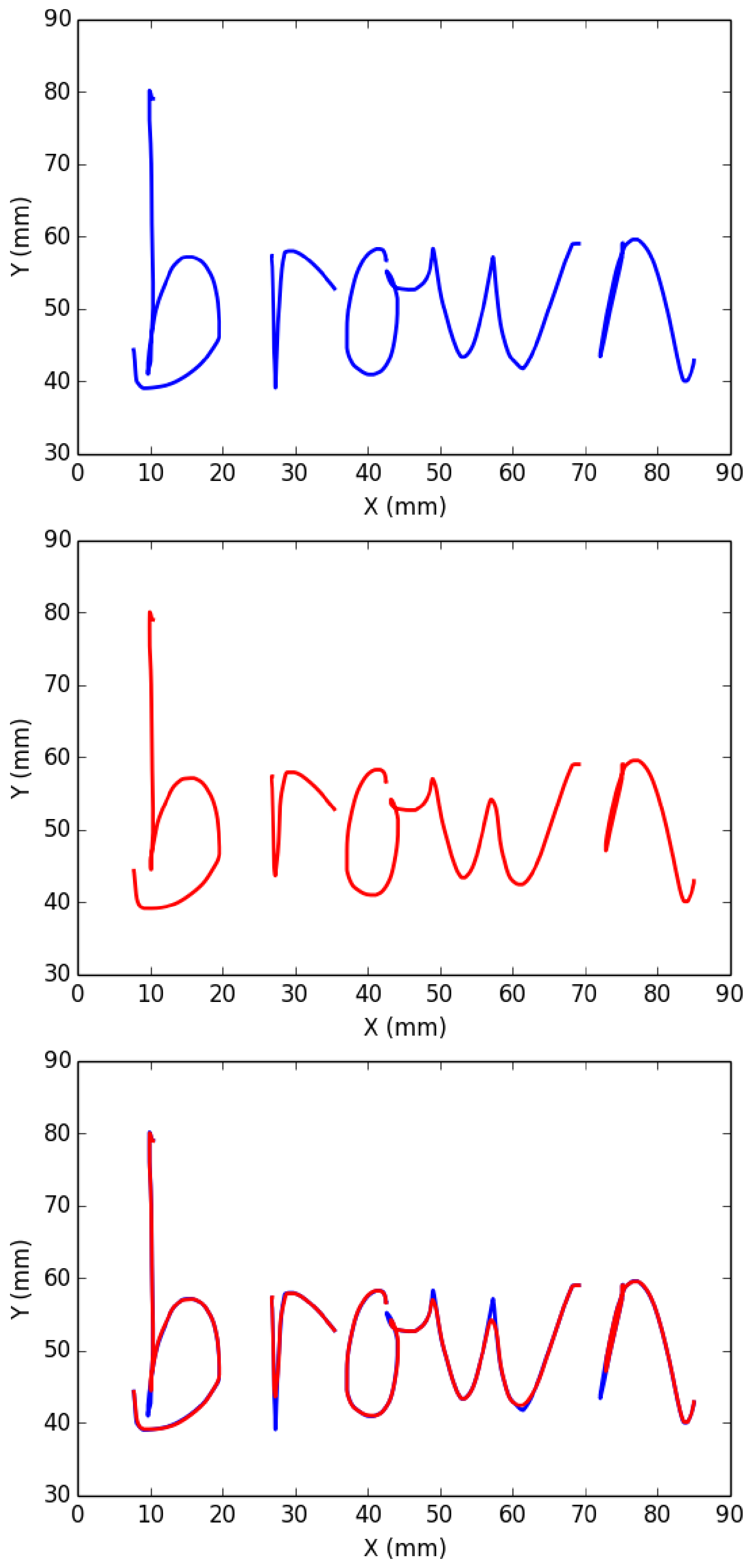

Figure 28: *Effect of decreasing smoothing in speed minima*

Figure *28* illustrates how decreasing the smoothing in the edge points affects the final robot path. Red line corresponds to a constant 50% smoothing setting, whereas the blue line corresponds to 0% smoothing in speed minima points, and 50% otherwise. As it can be seen from the image, varying the smoothing setting enables the robot to draw sharp corners while keeping a high smoothing setting outside the corner points.

5.2.4 Final Results

Results of the final experiments are introduced in this section. These experiments were done with following robot settings:

- speed: 3000mm/s
- smoothing in corner points: 0%
- smoothing outside corner points: 0% / 50% / 100%
- maximum approximation error: 0% / 0.5% / 1% / 10%.

Figure *29* shows an example writing by a test person. Figure *30* shows the same sample replicated with the robot using various settings.

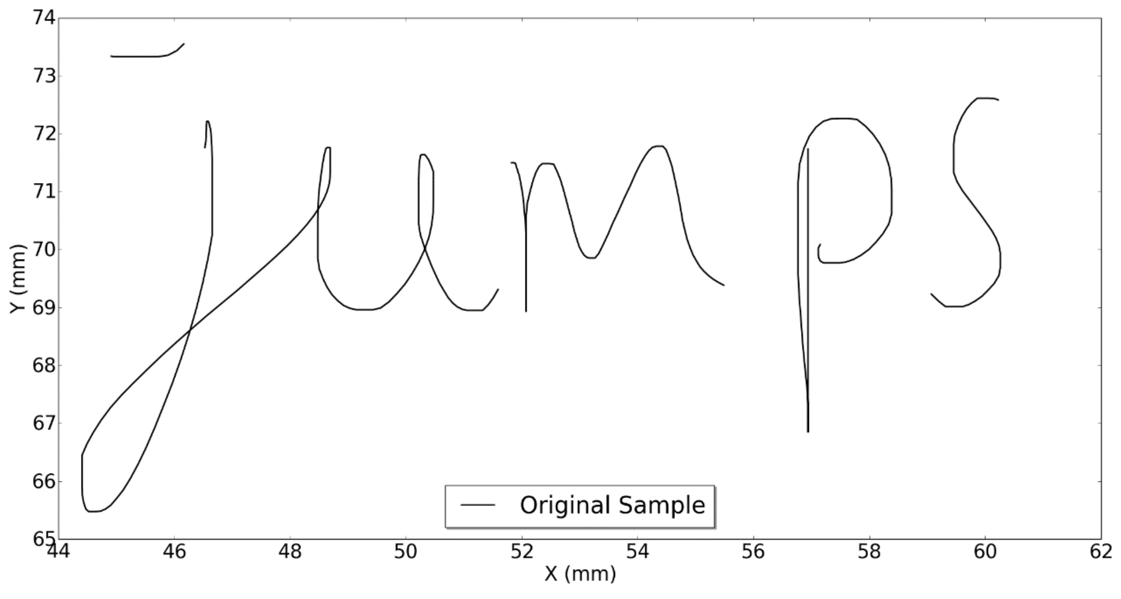

Figure 29: *Original sample written by a test person*

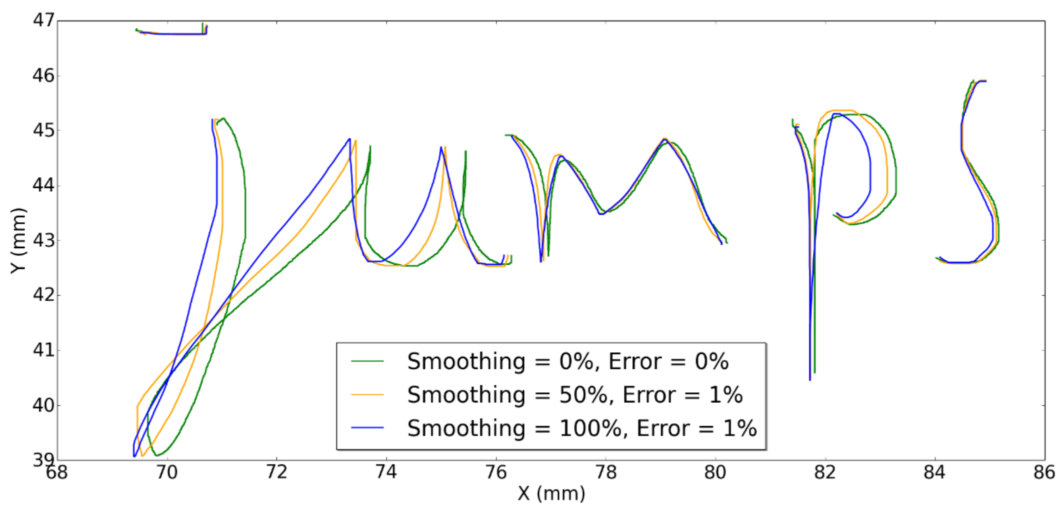

Figure 30: *Sample replicated by the robot*

During these experiments it was discovered that the robot cannot reach realistic speeds when the text is small, such as the text in Figure *30*. Each character in the figure is only a few millimeters wide, which significantly limits the robot acceleration time. Figure *31* shows the speed trajectories for both human and robot writing when small-sized text is utilized.

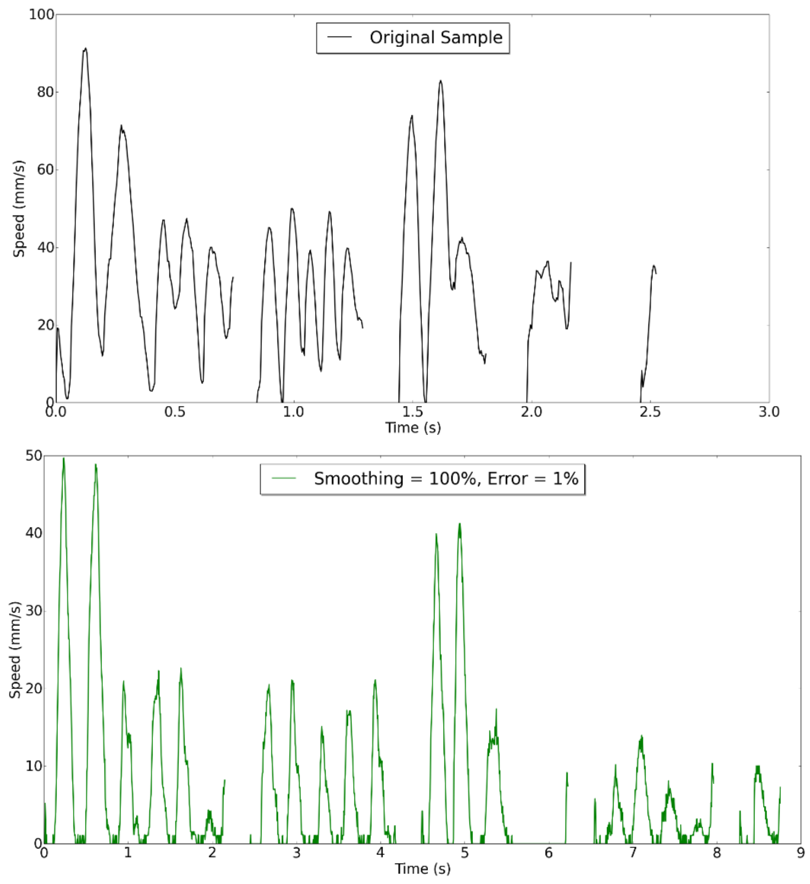

Figure 31: *Writing speed – human vs. robot*

To help the robot to reach higher speeds, experiments were done again with slightly larger writing samples. An example of a larger writing sample by a test person is shown in Figure *32*. Resulting robot path trajectories are shown in figures Figure *33* – Figure *35*.

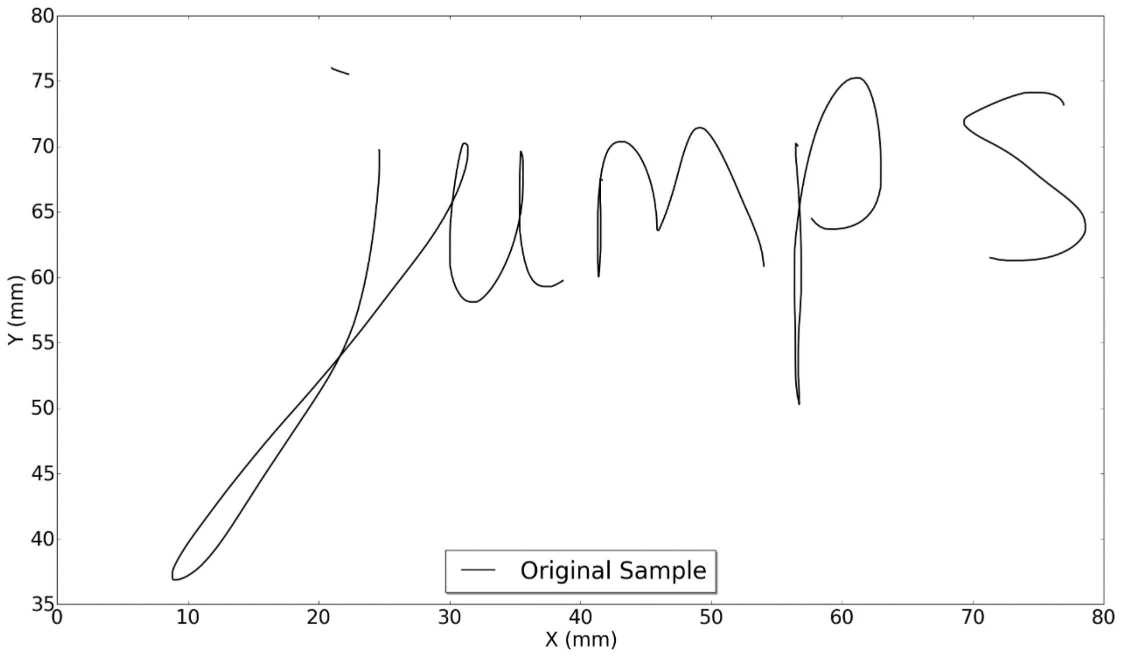

Figure 32: *Original sample written by a test person*

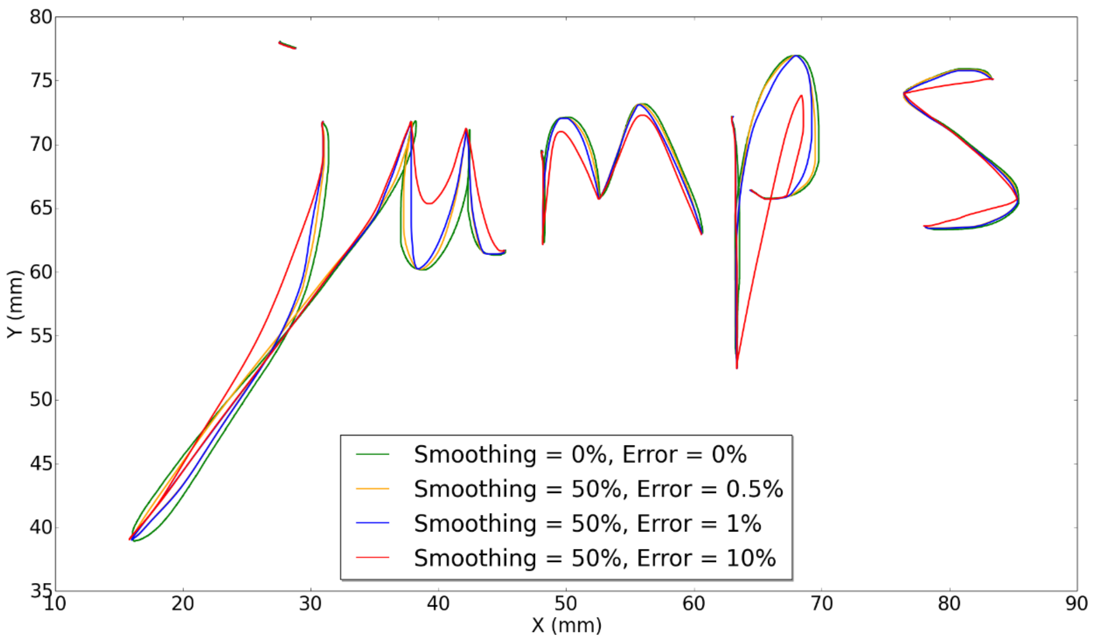

Figure 33: *Sample replicated by the robot (50% smoothing)*

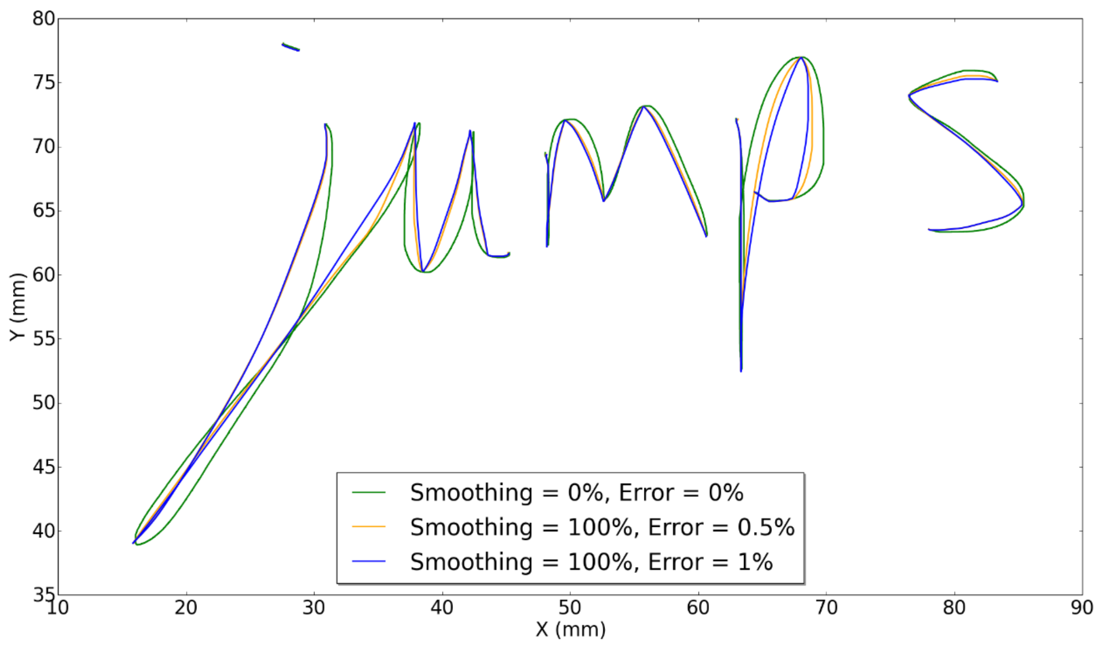

Figure 34: *Sample replicated by the robot (100% smoothing)*

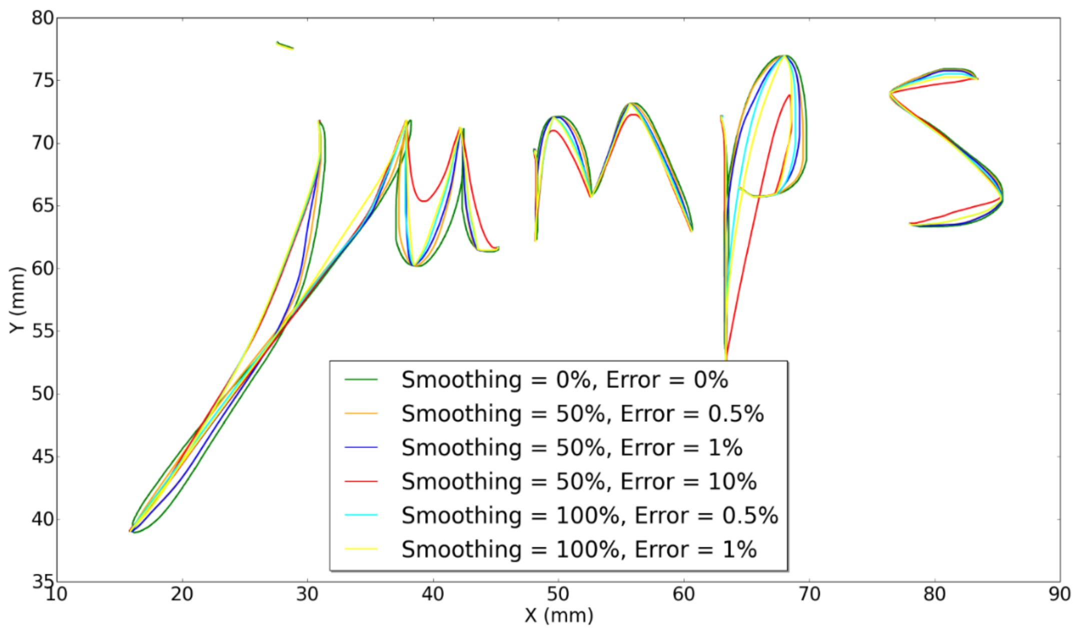

Figure 35: *Sample replicated by the robot (various settings)*

Speed trajectories for these experiments are illustrated in Figure *36*. Black line in the topmost image shows the pen speed of the original human writer, whereas the blue and red lines show the pen speed of the robot.

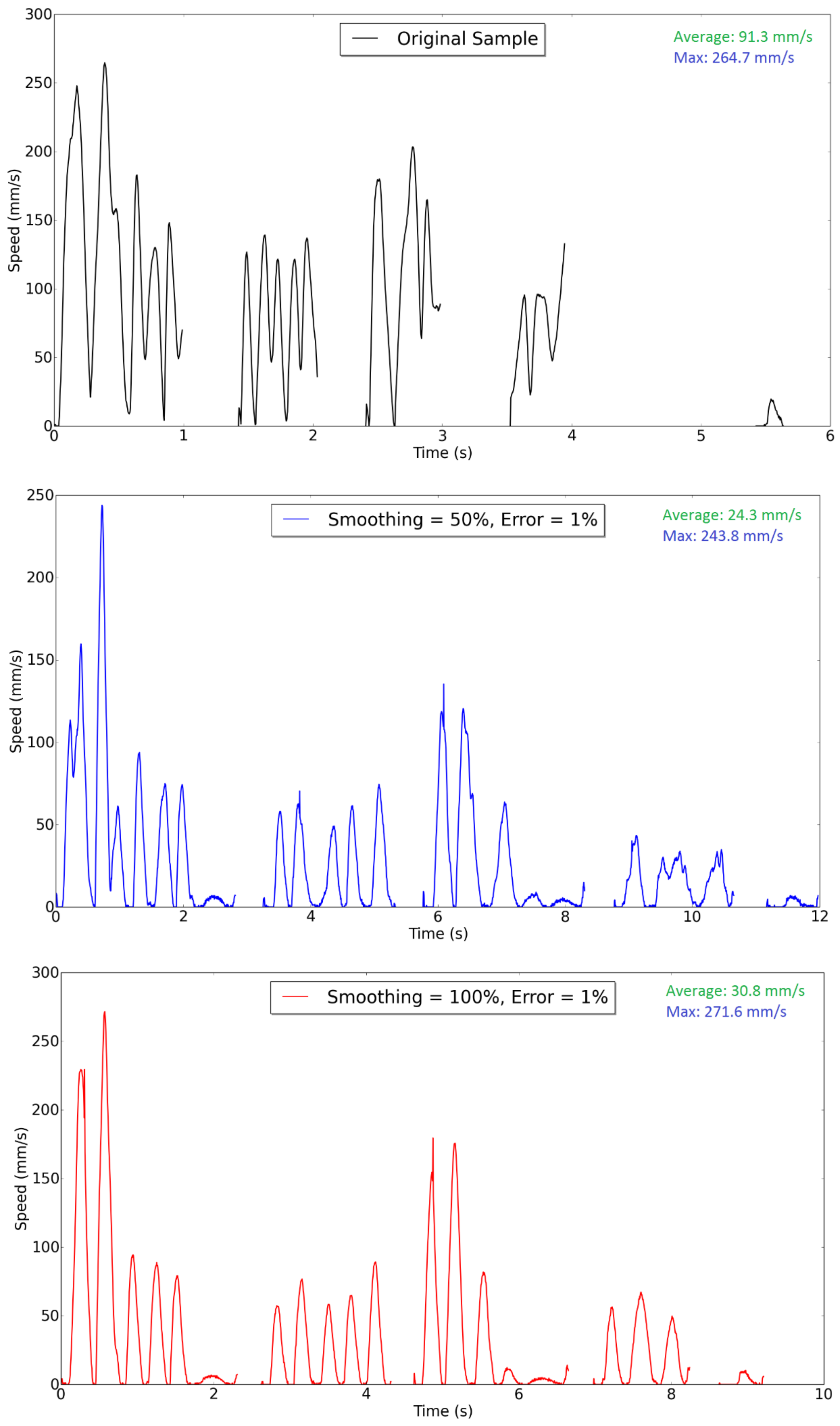

Figure 36: *Writing speed – human vs. robot*

Similar results were obtained for other samples as well. Another writing example is illustrated in figures Figure *37* and Figure *38*.

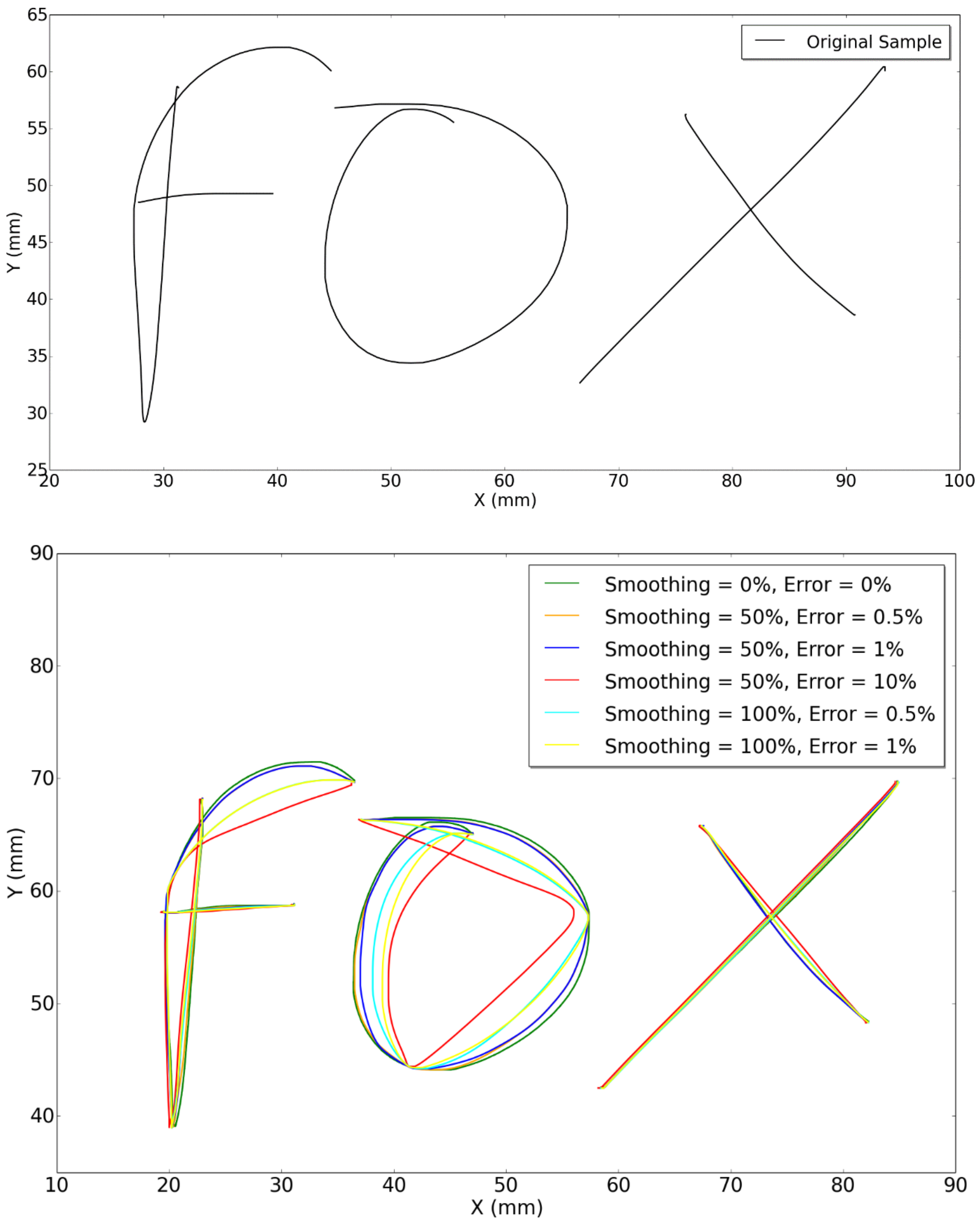

Figure 37: *Writing trajectories – human vs. robot*

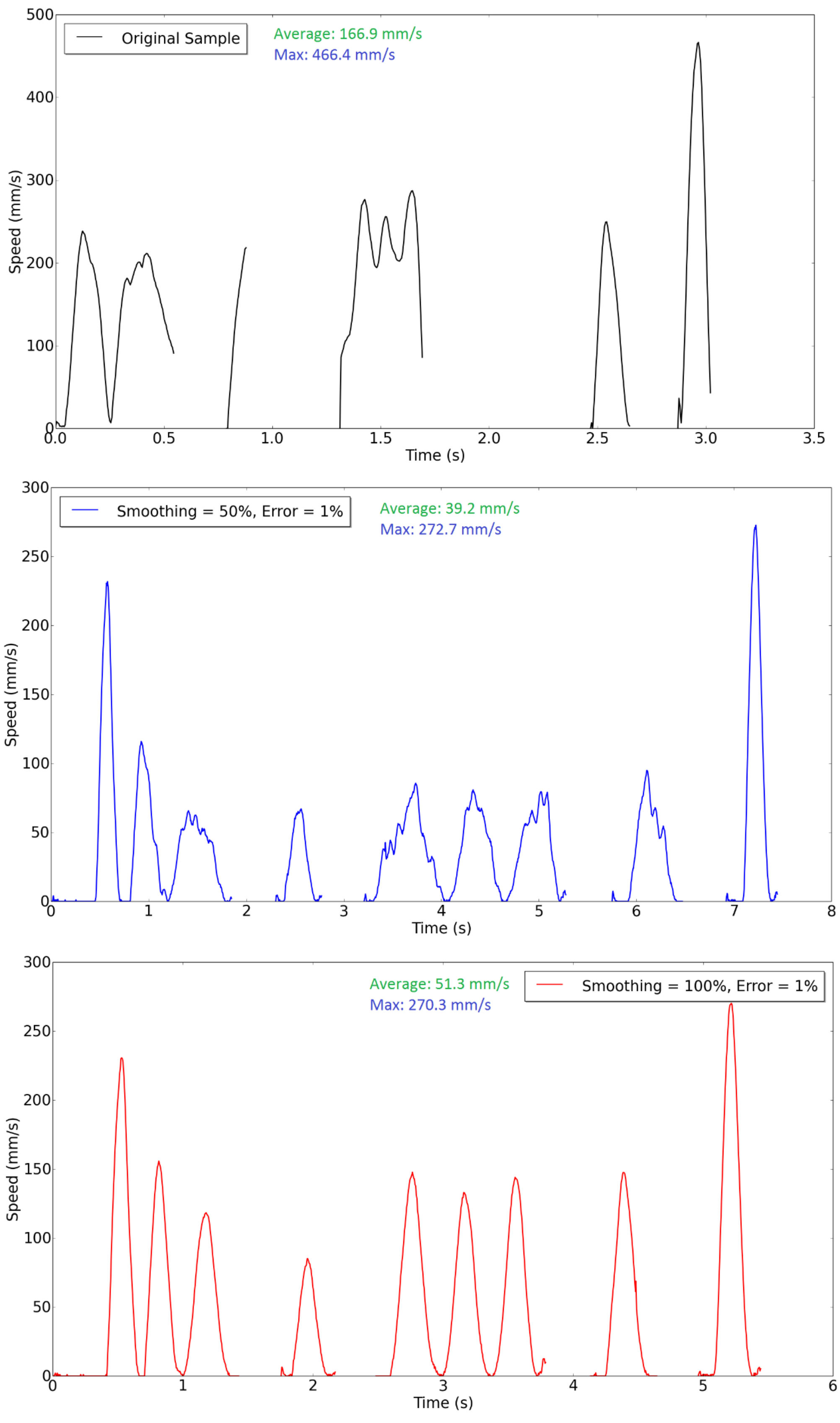

Figure 38: *Writing speeds – human vs. robot*

5.3 Analysis of the Results

Based on the measurement results, the robot is able to replicate all the handwriting samples in a relatively high accuracy. Accuracy of the resulting writing depends on the utilized robot settings, such as smoothing and path approximation. The robot manufacturer specifies a repeatability of ± 0.02 mm for the robot model, which means that the results are also highly repeatable.

As can be seen from the speed trajectories, the robot is also able to reach fairly high maximum speeds during writing. On the other hand, the measurements show that average speeds of the robot are not particularly high. This is partly due to the fact that the robot always completely stops its movement in the end of each stroke before raising the pen from the screen. Human writers, on the other hand, often tend to raise the pen in the end of each stroke while the pen is still in motion. This can be seen for example from Figure *38*. In the topmost image, pen speed never reaches zero at the end of strokes, whereas in the two latter images the speed always falls to zero. This has an effect on the calculated average speeds. To minimize this effect, the robot could be programmed to imitate the human hand at the end of strokes, so that it raises the pen from contact with screen while still moving, and stops it movement above the screen surface.

However, this does not explain all the differences between human and robot speed. For example, in the second last stroke in Figure *38*, the human writer is able to reach a speed of approximately 250mm/s, compared to approximately 150mm/s for the robot. It can be concluded that even with all the means introduced in this section, the robot is not able to match the human writing speed.

To further improve the robot writing speed, the biarc approximation method introduced in section 3.6.5 could be studied more. As the majority of industrial robots support arc movements, this method would potentially generalize well across different robot models. The second method for improvement would involve acquiring better control of the low-level robot hardware. This means that the robot programmer would be able to directly control the currents / voltages that are fed to the robot motors. This would provide much more freedom for implementing the algorithms that control the system. The biggest problem with this approach is that none of the most common commercial robot models offer this kind of deep controllability of the system. Suitable robots could be found by looking at robot models that are specifically made for research purposes.

6. CONCLUSIONS

To develop and produce new touch-based devices that satisfy the ever-growing performance and quality requirements, manufacturers are forced to do a lot of user-interaction testing. Doing this kind of end-to-end testing manually is a very labor-intensive process and prone to producing subjective results. Thus, all major manufacturers have adopted the use of industrial robots for automated touch device testing.

To allow creating comprehensive test cases, the utilized robots should be able to perform all the same interaction gestures as humans are. Of all the most common gestures, natural human handwriting is likely the most problematic gesture to imitate with robots. This problem includes both acquiring a realistic handwriting sample that the robot shall attempt to imitate, and commanding the robot in such a way that the resulted writing is as close to the original as possible, regarding accuracy, speed, acceleration, and other qualities.

This thesis discusses the problem of imitating human handwriting with industrial robots. First part of the work discusses necessary theory behind the problem. First of all, understanding the problem requires basic understanding about industrial robots. Topics that are relevant within the scope of this thesis, including description of basic mechanical structure, kinematics, and motion control of industrial robots, have been discussed in Chapter 2. Based on this theory, selected approaches that would enable the robot to move faster within the handwriting path were introduced in Chapter 4. These approaches include adjusting robot smoothing, approximating the path with fewer sample points, and using joint PTP motion.

Second part of the work describes the final implemented system. The system was used to replicate handwriting samples that were collected earlier to see how the various methods of Chapter 133 would perform in reality. Results of these experiments, along with analysis of the system performance, is provided in Chapter 5. Based on the results, the methods of Chapter 133 do help increasing the robot writing speed by essentially providing a way to adjust the tradeoff between robot accuracy and speed. This is done by adjusting robot smoothing, and the level of path approximation. Higher smoothing of the robot movement leads to higher robot speed, but lower accuracy. On the other hand, approximating the handwriting path with fewer points also increases robot speed, but too much approximation decreases accuracy.

Results show that the current system can replicate human handwriting well in terms of accuracy, but it is not quite capable of imitating the average speed of fast human handwriting. Thus, a system like the one presented in this thesis would be suitable for touch device testing where it is not required that the writing speed is perfectly imitated. The method discussed in Chapter 5.1 was found suitable for acquiring a database of realistic handwriting samples. Alternatively, an existing Internet database could be used as the source for samples. All in all, this project was found to provide good basis for further development around the topic, as well as potential commercial applications in the future.

REFERENCES

[1] Apple Inc., Pencil for iPad Pro, retrieved January 30, 2016, available at: https://www.apple.com/apple-pencil/

[2] OptoFidelity Oy, Touch Panel Performance Test Systems brochure, retrieved January 30, 2016, available at: http://www.optofidelity.com/wp-content/uploads/2013/09/OF_TPPT_general.pdf

[3] OptoFidelity Oy, Watchdog brochure, retrieved January 30, 2016, available at: http://www.optofidelity.com/wp-content/uploads/2013/07/OF_WatchDog_screen.pdf

[4] A. Ng, J. Lepinski, D. Wigdor, S. Sanders, P. Dietz, Designing for Low-Latency Direct-Touch Input, Proceedings of the 25th annual ACM symposium on User interface software and technology, 2012, p. 453-464

[5] R. Plamondon, S. Srihari, On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, 2000

[6] Microsoft Inc., Using the Pro Pen, retrieved January 30, 2016, available at: https://www.microsoft.com/surface/en-us/support/touch-mouse-and-search/pro-pen?os=windows-10

[7] MVTec Software GmbH, Halcon brochure, retrieved January 30, 2016, available at: http://www.halcon.com/halcon/brochures/pdf/halcon-12-brochure-english.pdf

[8] MyScript, MyScript Text, retrieved January 30, 2016, available at: http://myscript.com/technology/

[9] K. Franke, L. Schomaker, M. Köppen, Pen force emulating robotic writing device and its application, IEEE Workshop on Advanced Robotics and its Social Impacts, 2005, p. 36-46

[10] Bond Inc., Bond web page, retrieved January 30, 2016, available at: https://bond.co/

[11] J. Craig, Introduction to Robotics: Mechanics and Control, 3rd edition, Prentice Hall, 2004

[12] J. Craig, Introduction to Robotics: Mechanics and Control, 3rd edition, Prentice Hall, 2004, p. 68 - 102

[13] Introduction to Robotics: Module Trajectory generation and robot programming, FH Darmstadt, 2000

[14] B. Siciliano, O. Khatib, et al., Springer Handbook of Robotics, Springer, 2008

[15] B. Siciliano, O. Khatib, et al., Springer Handbook of Robotics, Springer, 2008, p. 19 – 31

[16] B. Siciliano, O. Khatib, et al., Springer Handbook of Robotics, Springer, 2008, p. 69

[17] Fanuc M-1ia datasheet, retrieved January 31, 2016, available at: http://www.fanucrobotics.com/cmsmedia/datasheets/M-1iA%20Series_11.pdf

[18] Epson G-series robot brochure, retrieved January 31, 2016, available at: http://robots.epson.com/admin/uploads/product_catalog/files/EPSON_G-Series_SCARA_Robots%28RevE%29.pdf#zoom=90

[19] Wikipedia article, Plotter, retrieved January 31, 2016, available at: https://en.wikipedia.org/wiki/Plotter

[20] Wikipedia article, Wacom, retrieved February 6, 2016, available at: https://en.wikipedia.org/wiki/Wacom

[21] Wacom Technology Corp., Wintab Interface Specification 1.4, October 2010, retrieved February 6, 2016, available at: http://www.wacomeng.com/windows/docs/Wintab_v140.htm

[22] University of Bern, IAM On-Line Handwriting Database, retrieved February 6, 2016, available at: http://www.iam.unibe.ch/fki/databases/iam-on-line-handwriting-database

[23] University at Buffalo, CEDAR Online Handwriting Database, retrieved February 6, 2016, available at: https://www.cedar.buffalo.edu/handwriting/HRdatabase.html

[24] A. Del Prete, N. Mansard, O. E. Ramos, O. Stasse, F. Nori, Implementing Torque Control with High-Ratio Gear Boxes and without Joint-Torque Sensors, International Journal of Humanoid Robotics, 2015

[25] P. Heckbert, M. Garland, Survey of Polygonal Surface Simplification Algorithms, Carnegie Mellon University, 1997, p. 4

[26] A. Avdzhieva, D. Aleksov, I. Hristov, N. Shegunov, P. Marinov, Circular arc spline approximation of pointwise curves for use in NC programing, Final Report, ESGI'104, 2014

[27] M. Campbell, Apple invents continuous handwriting input method, magnetically linkable earphones, 2015, retrieved: March 4, 2016, available at: http://applein-sider.com/articles/15/07/09/apple-invents-continuous-handwriting-input-method-magnetically-linkable-earphones

[28] T. Chandrupatla, T. Osler, Planar Biarc Curves – A Geometric View, Rowan University, retrieved March 23, 2016, available at: http://www.rowan.edu/col-leges/csm/departments/math/facultystaff/osler/128%20Pla-nar%20Biarc%20Curves%20by%20Chandrupatla%20and%20Osler%20%20Re-vised.pdf

[29] C. Rocchini, Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/in-dex.php?curid=32858984

[30] G. Maier, Optimal arc spline approximation, Computer Aided Geometric Design. 2014

[31] A. Atmosudiro, M. Keinert, A. Karim, A. Lechler, A. Verl, A. Csizar, Productiv-ity Increase through Joint Space Path Planning for Robot Machining, UKSim-AMSS 8th European Modelling Symposium, 2014