Intuitive Robot Programming by Demonstration

Fredrik Bagge Carlson

Department of Automatic Control
Lund University
Lund, April 2013

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

ISSN 0280–5316
ISRN LUTFD2/TFRT--5918--SE

Printed in Sweden by Media-Tryck.
Lund 2013

Abstract

An intuitive programming method for industrial robots is proposed and investigated. The main goal is to reduce the time required to program an industrial robot without the use of expensive reverse engineering equipment. Using a pen shaped tool and an optical tracking system, the user indicates the desired robot trajectory, whereafter a 3D scanner is used to gain knowledge of the work piece. The indicated path and 3D scan is then fused together in order to automatically generate an executable robot program.

Acknowledgements

I would like to thank Dr Rolf Johansson for showing an interest in my studies and initiating the contact with SIMTech. I would also like to thank my thesis advisor Dr Vuong Ngoc Dung at SIMTech for great support and commitment during my time in Singapore.

Contents

0.1 Glossary

Point cloud A data set of points $(x,y,z) \in \mathbb{R}^3$. Usually representing an object or a surface.

Optical tracking system A system consisting of two or more video cameras capable of tracking a rigid body marked with optical markers. Referred to as the tracker.

Marker An object, preferably spherical in shape, capable of reflecting or emitting infrared light. Referred to as passive or active depending on whether it is reflecting or emitting light.

Path A series of coordinates $(x,y,z) \in \mathbb{R}^3$ and orientations $(roll, pitch, yaw) \in \mathbb{R}^3$ recorded by the tracker. Every point on the path may be associated with an estimate of the direction, or tangent.

Teaching tool A rigid body equipped with three or more markers used by the human operator to teach a path.

Laser scanner A system consisting of one or more video cameras and a laser source capable of emitting laser light in a plane. The laser scanner is capable of generating a point cloud representation of an object.

Surface The point cloud generated by the 3D scanner is referred to as the surface.

Registration The act of finding a rigid transformation, rotation R and translation $P = (x,y,z)$, between two set of points so that corresponding points in the two sets are mapped to the same location.

0.2 Abbreviations

API Application Programming Interface

CAD Computer Aided Design

DOF Degrees Of Freedom

HMLV High-Mix Low-Volume

ICP Iterative Closest Point algorithm

OpenCV Open Computer Vision library

PCA Principal Component Analysis

PCL Point Cloud Library

SME Small and Medium-sized Enterprise

SNR Signal to Noise Ratio

1

Introduction

1.1 Background

Recent developments in industrial robotics has led to cheaper robots, which have become more accessible to small and medium-sized enterprises (SMEs). The traditional way of programming a robot however has not seen any such large scale advancements and still require a significant amount of man-hours, thus being a major part of the costs related to operating an industrial robot. Traditionally, robots are programmed using an *on-line* programming procedure, such as *Lead through* or *Walk through* [Todd 1986]. This requires a robot operator to manually control the robot to a number of poses along the actual trajectory, which will be stored in a robot program. The operator will then specify the speed and type of movement needed to reach the pose. This must be done for all individual robots on the floor, requiring a significant amount of work load. Another way of programming industrial robots made popular by increased absolute accuracy and a demand for more complex trajectories, is *off-line* programming. Off-line programming involves the planning of a robot trajectory using a CAD model of the workpiece. This procedure differs from on-line programming in one key aspect. During on-line programming, the robot will remember the joint configuration of each pose it was controlled to. All the robot has to do in order to replay the trajectory is to move to all remembered poses in the specified way. This can generally be done with very high precision. The robots ability to replay a remembered trajectory is called the robots repeatability, which is usually high. For off-line programming however, the robot must be able to reach a pose specified in some external coordinate system, and must solve for the joint angles that will achieve this pose. From factory, the robot controller uses nominal values for parameters like segment lengths. These values differs slightly between individual robots, and the difference between the actual length of the segments and the once used in the model will result in an error in the final position reached by the robot. This error is dependent on the current joint angles, and is in general non-constant within the work space. The robots ability to reach a point specified in an external coordinate system is termed its absolute accuracy. This can vary from less

than 1mm up to more than 20mm, and will thus introduce difficulties for the off-line programming method, which involves workpiece localization and calibration.

1.2 Problem formulation

Both programming methods described above require a significant time investment to be successful. For a large enterprise operating at high volumes, this may be of low concern. For SMEs however, the situation might be different. SMEs often face High-Mix, Low-Volume (HMLV) orders. To program an industrial robot using on-line programming might be economically unfeasible if the task is to be repeated a low number of times. Furthermore, the CAD model of the work piece may not be available. This might be the case if none ever existed or if the work piece has suffered wear and tear since it was fabricated and the CAD model no longer represents the actual work piece. Without a CAD model, the robot programmer must resort to reverse engineering of the work piece in order to allow off-line programming.

To summarize, many SMEs would benefit greatly from a faster, more cost efficient way of programming industrial robots, which do not require a CAD model of the work piece. A company looking for a way to automate HMLV orders, such as maintenance and finishing tasks, require an economically feasible programming method where a desired solution meets the following requirements

- Low cost implementation.
- No CAD model requirements.
- Sufficient accuracy for contact finishing tasks using force control operation.
- Reduced programming time compared to on-line programming.
- Low operator skill level required.

This work will asses the feasibility of constructing a robot trajectory, consisting of a series of 6 degrees-of-freedom (DOF) points, by capturing the users hand movements as he demonstrates the trajectory. Sub problems visited includes the capturing of input with high accuracy, curve reconstruction, target frame assignment and execution accuracy enhancement. The ultimate goal is to develop a method which will reduce the costs related to the programming and operation of an industrial robot, while maintaining an acceptable level of accuracy for both contact and non-contact finishing tasks, such as spray painting, polishing or deburring.

1.3 Method

The proposed programming method, called Intuitive robot programming by demonstration, introduces an optical tracking system, described in section 2.1, to capture the movements of a human operator. More precisely, the tracking system is used to track an object, referred to as the teaching tool, held by the user. A description of this tool is given in section 4.2. The user indicated trajectory will result in a series of coordinates $\in \mathbb{R}^3$, referred to as the path. Using knowledge of the path, a 3D scan of the area surrounding it will be conducted. The resulting information will be used to gain knowledge of the work piece, such as surface normal, structure and curvature. Using this additional information, each three dimensional target point on the path will be augmented to a six dimensional target frame, specifying both location and the desired orientation of the robot tool as it passes through the point during execution of the trajectory. To reduce the final trajectory following error, an error correction method is proposed and investigated (section 7.2).

Three main tracks can be distinguished and are illustrated in form of Fig. 1.1. The initial track deals with the user input. This involves recording and processing of the data from the optical tracking system (section 4), division of the path into segments and segment classification (section 6.3). The second track deals with the scanning of the work piece (section 5) and processing of the resulting scan (section 5.5). The third track fuses the previous two together in order to produce a complete trajectory. Here, registration of the path and the surface is considered (section 6.4), reconstruction and enhancement of the path is done (section 6.5) and points on the path are equipped with target frames (section 6.6). A summary of the results will be found in section 8 after which the developed Graphical User Interface (GUI) will be presented in section 9. A final discussion is held in section 10.

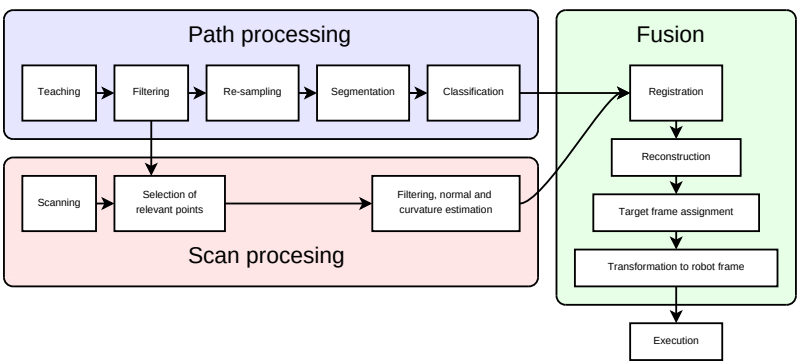


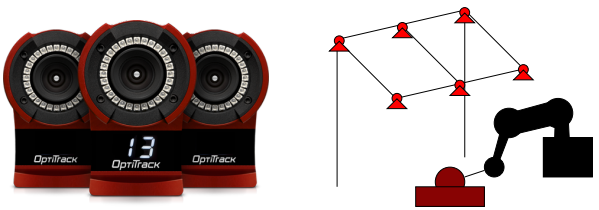
Figure 1.1: Procedure outline

2

Hardware and software

2.1 Optical tracking system

The optical tracking system chosen is the Flex 13 from NaturalPoint[1]. The tracker consists of six cameras (Fig. 2.1(a)) arranged in order to cover a large volume, see Fig. 2.1(b), while not limiting the motions of the robot or the user. A few criteria



(a) Flex 13 cameras (*NaturalPoint 2013*)

(b) Workspace layout

Figure 2.1: The optical tracking system

was considered when the tracking system was chosen.

- Frame rate
- Resolution
- Cost
- Number of cameras
- Programming API

A NaturalPoint tracker was previously used at SIMTech why in-house experience favored a NaturalPoint system. The chosen system has a frame rate of 120FPS,

[1] `http://www.naturalpoint.com/optitrack/products/flex-13/`

which is considered sufficient in order to capture the relatively slow movements of the user. The resolution of 1280×1024px is significantly higher than the system previously used at SIMTech, which already provided acceptable accuracy. The maximum number of cameras that was considered feasible both in terms of cost and practical installation was then chosen in order to maximize the workspace and minimize problems that can arise when the tracked object is blocked from view for one or more cameras. Each camera is equipped with an infrared LED-strobe, providing illumination of the capture volume.

The tracking system is used to track the location of special purpose markers. A marker, described in section 4.2, is a small spherical object coated with reflective material, can be tracked accurately by the tracking system. By grouping a set of markers attached to a rigid body, called a trackable, both the position and orientation of the rigid body can be determined. The minimum amount of markers needed to track a rigid body in 6 DOF is three, provided that at least two cameras see all the markers at the same time. Redundant markers can be used to increase the accuracy and provide robustness with respect to occluded markers.

Fig. 2.2 shows the Tracking Tools interface and the layout of the cameras above the three tracked rigid bodies, the tool frame, the work object and the teaching tool.

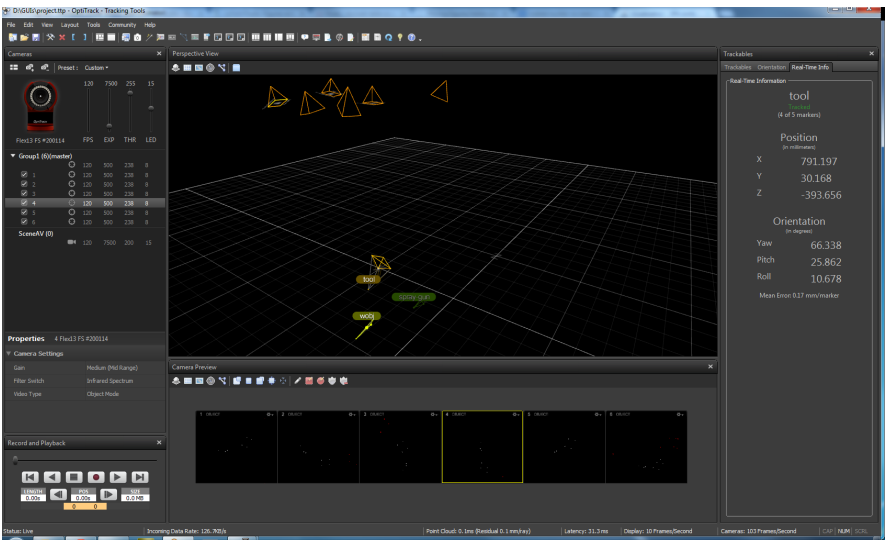


Figure 2.2: Tracking tools software

2.2 Scanner

The 3D scanner used is the Microsoft Kinect. It was originally developed as an alternative input method for the video game console Microsoft XBOX 360, but has been hugely popular in the robotics and computer vision communities for its relatively high performance, together with a very affordable price. The point cloud produced by the Kinect measures $640 \times 480 \approx 300\,000$ points. The minimum working distance of the range sensor is approximately 50cm, which together with the resolution poses an upper limit on the attainable point density. The Kinect is based on a technique called Structure of Light, which involves the projection of a known infrared pattern on the surface in front of the sensor. The known relation between the projector and the sensor can then be used to solve for the depth coordinate of pixels in the image captured by the sensor. This method of 3D scanning is fast (the Kinect can produce output at up to 30Hz) and requires no motion.

As a second alternative, a laser scanner, or laser stripe profiler, has been investigated. The laser scanner provides significantly better resolution and accuracy compared to the Kinect, but was turned down based on the estimated work required to implement a working system. A section outlining the theory of a laser scanner, together with proposed configurations and calibration methods, can be found in appendix B.

2.3 Robot

The robot used for this research is the seven DOF Motoman IA20, shown in Fig. 2.3, together with the controller NX1000. The NX1000 provides a limited programming API which allows direct control of the robot through a PC. Through the API, only six out of the seven DOF is controllable. The maximum payload of the robot is approximately 20kg.

Figure 2.3: Motoman IA20

2.4 Third party software

A number of third party libraries has been used for this research. The most notable are listed below. Aside from the main libraries, a number of standalone functions implemented by individuals have been used and are referenced where mentioned.

Point Cloud Library

> *The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing*[2].

PCL provides a large set of functions for manipulating, transforming, filtering and registering point clouds. PCL is used to interface with the Kinect and to process the resulting point clouds.

Matlab

> *MATLAB® is a high-level language and interactive environment for numerical computation, visualization, and programming*[3].

All processing of the data recorded by the tracker as well as the fusion of the scan and the path is implemented in Matlab. Matlab Builder NE is used to generate a .NET class library which can be called from any program supporting the .NET framework. The computer running the main program is not required to have Matlab installed, instead, it is sufficient to install the Matlab runtime, which is free to distribute. The Matlab runtime will be initialized when the main program starts up, after which Matlab functions can be called as if they were native .NET functions. Parameters passed to and from the Matlab functions are automatically marshaled to the correct Matlab type, transparent to the programmer.

ILNumerics

> *ILNumerics is a high performance math library for applications. It simplifies the implementation of all kinds of numerical algorithms in convenient, familiar syntax - optimized to the speed of C and FORTRAN*[4].

ILNumerics is used for all computations carried out in the main program. The library provides a syntax similar to Matlab, and has basic functionality for 3D plotting, used to visualize frames in the GUI.

[2] `http://pointclouds.org/`
[3] `http://www.mathworks.com/products/matlab/`
[4] `http://ilnumerics.net/`

3

Theory

This chapter will present a brief overview of the most important theoretical concepts used in this work.

3.1 Coordinate frames

A coordinate frame A is for our purposes defined with respect to another frame T by the coordinates of its origin (x, y, z) and the rotation of its three basis vectors with respect to the basis vectors of frame T.

The following coordinate frames will be used frequently and can be seen in Fig. 3.1 on the facing page.

T_T The frame of the optical tracker. This frame will be considered the base frame or global frame. All measurements by the tracker are given in this frame.

T_{RB} The robot base frame. This is the frame used by the robot.

T_S The frame of the 3D scanner, all measurements conducted with the scanner are initially given in this frame.

T_{TF} The robot tool frame. The frame of the last part of the robot.

T_{TCP0} The tool contact point frame, orientation aligned with the tool frame. T_{TCP0} is simply the translation from the tool frame to the contact point of the tool.

T_{TCP1} The tool contact point frame, orientation aligned with the tracker frame. This matrix contains no translation, it will only rotate T_{TCP0} such that it is aligned with the tracker frame. This is the frame assigned by the tracking software when a rigid body is formed by grouping a selection of markers. By the convention chosen, it is essential that the orientation of the trackable is set so that the force acting on the tool when it approaches the surface at desired angle, is aligned with the tracker y-axis, see Fig. 3.1 for reference.

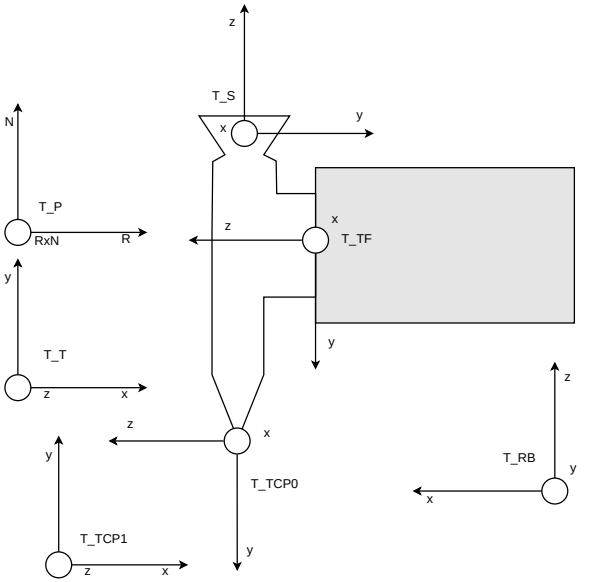


Figure 3.1: Coordinate frames

3.2 Rigid body transformations

A (proper) rigid body transformation consists of a rotation and a translation. When acting on a vector $\mathbf{v}$, the rigid transformation results in the vector

$$T(\mathbf{v}) = R\mathbf{v} + \mathbf{t} \tag{3.1}$$

R is a 3 by 3 orthogonal transformation matrix with $det(R) = 1$ and $\mathbf{t}$ is a 3 by 1 translation vector.

By constructing the block matrix

$$\mathbf{T} = \begin{pmatrix} & \mathbf{R} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and writing the vector $\mathbf{v} = [x, y, z]^\mathsf{T}$ as $\mathbf{v} = [x, y, z, 1]^\mathsf{T}$, the equation 3.1 can be written as

$$T(\mathbf{v}) = T\mathbf{v} \tag{3.2}$$

In the following, all rigid body transformations will be written on the form T_L^U, where U is the frame in which the vector was given before translation and L is the frame in which the resulting vector is expressed. A vector v expressed in a frame A can thus be transformed to the frame B by the transformation $\mathbf{v}_B = T_B^A \mathbf{v}_A$.

3.3 Principal Component Analysis

Through the entire document, we will find ourselves dealing extensively with point sampled surfaces and curves in three dimensions. Often times, it will prove useful to find some representation of the data sets, such that a large fraction of the information is contained within one or two dimensions. Principal Component Analysis (PCA) [Pearson 1901] is a method of finding a vector $\in \mathbb{R}^n$, describing the greatest amount of variance present in the data. This vector will be called the first principal component of the data set. The second principal component will be the direction which describes as much as possible of the variance not described by the first principal component, under the restriction that the two components are orthogonal. A $m \times n$ data set with m points in n dimensions will thus have n principal components, ordered in decreasing order of importance in describing the data set. Fig. 3.2 illustrates the concept for a sample data set.

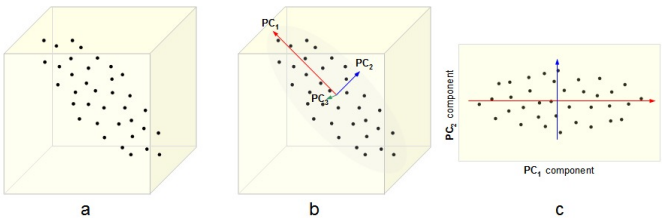


Figure 3.2: Illustration of dimensionality reduction using PCA [Kavraki 2013].

A surface locally exhibits a two dimensional structure. Using PCA to analyze the data points, a new basis will be found such that when the data is described in the new basis, the third component will be close to zero, thus effectively reducing the dimensionality of the data to two dimensions. If the surface was indeed completely flat, the third principal component would explain no variance at all and point straight up from the surface, effectively yielding an estimate for the surface normal at the center of the data set used in the analysis.

Curves in $\mathbb{R}^3$ are also assumed to locally exhibit a two dimensional structure. By expressing the points along the curve in the basis described by the principal components, one can, without loss of any significant amount of information, project the data onto the subspace spanned by the first two components. This dimensionality reduction allows for instance two dimensional analytical functions to be fitted to the data.

3.4 Robot Calibration

The aim of this research is not the traditional tool calibration and robot to tracker calibration. It is however important to shed some light on why this is needed and how it is usually done.

There are two parts to be determined in the relation between two coordinate systems. The location of their origins with respect to each other and the orientation of their axes. A calibration method making use of screw theory, described in [Yang *et al.* 2002], has been implemented by Mr. Lin Zi Jian as a parallel project at SIMTech. The method will simultaneously solve for both T_{RB}^{T} and T_{TF}^{TCP}, based on a number of robot poses measured by the tracker.

Both matrices T_{RB}^{T} and T_{TF}^{TCP} are fixed and will not change during normal operation. However, as mentioned briefly in section 1.1, the robot controller relies on internal parameters for the segment lengths in order to compute the current position of the tool frame through a process called forward kinematics. Normally, the controller will use the nominal values of these parameters. A slight deviation from the nominal values in each of the actual segment lengths will amplify through the kinematic chain and result in a large error at the tool frame. Depending on the pose, this error can range between zero up to as much as several cm. This will cause the robot base coordinate system to be non-linearly distorted with respect to the tracker coordinate system. A simple rigid body transformation matrix can not compensate for this nonlinear distortion, and in order to resolve the problem, a calibration of the robot intrinsic parameters must be carried out.

During traditional online programming of a robot, the robot will be manually moved to points along the trajectory. The robot will then only remember the joint configuration at that point. This joint configuration is a set of seven (for a seven DOF robot) parameters, not related to the Cartesian world coordinate system. Given an industrial robots high repeatability, the pose can be reached again with very high accuracy. The problem of the errors in the kinematic chain only manifests itself as the robot is commanded in an outer Cartesian coordinate space. For an offline programming method like the one considered, this poses a difficulty in attaining a high absolute accuracy. A method of dealing with this problem is proposed in section 7.2.

4

Recording a path

The main idea of intuitive robot programming is for the user to show the robot how to perform a task, as opposed to programming the robot by a teaching pendant. With the teaching pendant, the user must manually move the robot to a number of poses along the intended path and specify the velocity and type of movement needed to reach the pose.

Recording the user indicated path with high accuracy is essential. Next to a number of technical problems related to the recording of the location of optical markers, another issue affects the accuracy of the path; the ability of the user to accurately follow the intended path. When drawing a straight line, most people would use a ruler as support, if not, the line will be imperfect. For practical reasons, using a ruler as support when indicating a path along a free form edge is not possible. How this can be assessed is discussed in section 4.2.

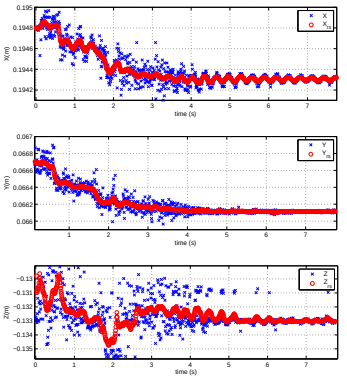
4.1 The OptiTrack Flex 13

Here, a brief discussion is provided on the functions and performance of the OptiTrack Flex 13 tracking system.

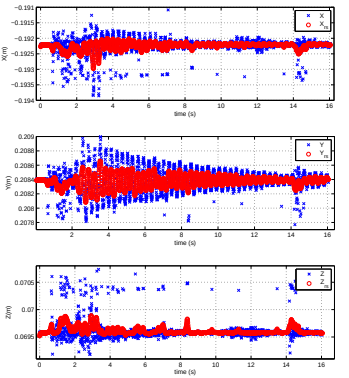
Accuracy

The claimed accuracy of the OptiTrack tracking system is less than 1mm in good conditions, meaning a small workspace, good calibration result and full visibility of all markers. For our purpose, these conditions can not be met. Our workspace is larger than $10m^3$ and markers are often obstructed by the robot, the user, the workpiece or by other markers. Apart from this, an industrial setting contains several vibration sources, such as the robot, the user and other machinery. The impact of such disturbances are shown in Fig. 4.1.

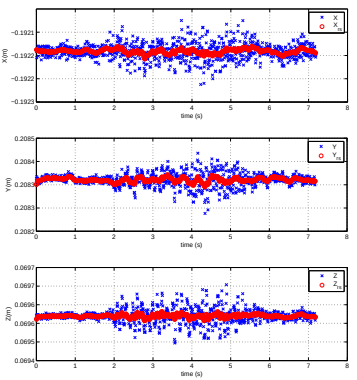
Since two cameras are sufficient to obtain the 3D coordinates of a point, adding more cameras to the setup has a positive influence on accuracy, due to added redundancy in the measurements. More cameras, with different viewpoints, will also handle occlusion of markers better.



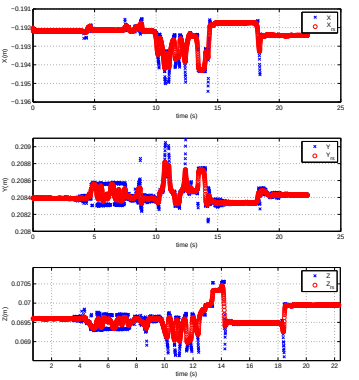
(a) Vibrating workpiece. The workpiece was moved manually on the ground.



(b) Vibrating camera frame. Vibration induced by tapping the frame lightly with a finger.



(c) Vibrating robot. Vibration induced by tapping the robot by hand, starting at t =2s.



(d) Moving and obstructing robot. The robot was moved around using the servo. The servo was turned on and off several times and the view of the object was obstructed for different cameras during the path.

Figure 4.1: Impact of various disturbances on the recorded location of an object. The object was stationary during all tests except for case 4.1(a). The graphs show both raw and filtered measurements.

The camera frame design used for the experiments was known before construction to be sub optimal, and can easily be stabilized. The design was chosen with flexibility and camera arrangement evaluation in mind. However, even this weak design proves sufficiently rigid for our purposes, despite the fact that it is coupled to the robot. The largest source of error is the obstruction of markers as seen in Fig. 4.1(d) and the vibration of the workpiece in Fig. 4.1(a). The magnitude of the vibrations induced when moving the workpiece was largely exaggerated and are unlikely to occur in a real scenario. The obstruction of the markers however is a potentially serious issue. Due to the nature of the task, there is no feasible arrangement of markers or cameras that can guarantee perfect tracking performance for all workpieces and poses. By posing restrictions on the angles and orientations in which the user can teach, or on the volume of the workspace, the setup can be customized to fit the particular task at hand. The final setup will always be a balance between cost and flexibility as well as between accuracy and workspace volume.

Relative accuracy Assessment of the accuracy of the optical tracker has previously been done at SIMTech. For confirmation, a simple experiment is conducted as follows.

A rigid body of cylindrical shape with 1cm radius was equipped with three markers made out of reflective material wrapped around the cylinder. The ring-shaped markers were approximately 4mm wide. The rigid body was positioned 1500mm from the tracker and so that the cylinder normal was pointing in the direction of the cameras and then moved 100mm along the tracker frame x-axis. At both endpoints 160 samples were collected. The experiments were then reproduced with the rigid body tilted so that the cylinder axis was almost pointing in the cameras direction, so as to allow the cameras to see the circular markers. The rigid body was once again moved 100mm along the tracker frame x-axis. The results indicate that the relative accuracy is indeed sub-mm and that the angle has no practical influence in the accuracy, see Table 4.1.

	Standard deviation in mm	
	First position	Second position
Normal facing camera	0.0072	0.0223
Axis facing camera	0.0068	0.0208
Mean distance moved in mm	First run	99.8626
	Second run	99.6853
Difference in movement in mm	0.1772	

Table 4.1: Test results from experiments on movement of rigid body.

Tracking tools API

The output from the OptiTrack system is by default the coordinates of the tracked object, given in a right handed coordinate system with origin in the center of the tracker. This can however be changed by specifying a ground plane. When a ground plane is set, the output from the tracker is with respect to the ground frame as chosen by the user. The output is in the form of a vector with the seven elements $[x, y, z, qx, qy, qz, qw]$, where the last four elements form a quaternion, specifying the current orientation of the object.

The tracking tools software allows the user to choose whether the output should be given in a right or left handed coordinate system. The setting is however not propagated to the quaternions given by the programming API, which are always given in a left handed coordinate system. To compensate for this, the given output from the tracker is transformed into its right handed representation by negating the z and qw-components.

To put the output in the form of a transformation matrix, a translation vector is constructed as

$$\mathbf{P} = \begin{pmatrix} x \\ y \\ -z \end{pmatrix}$$

The upper left hand corner of the transformation matrix $\mathbf{T}$, describing the orientation, is constructed from quaternions $q = [a, b, c, d] = [-qw, qx, qy, qz]$ with the formula[1]

$$\mathbf{R} = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc+2ad & 2bd-2ca \\ 2bc-2ad & a^2-b^2+c^2-d^2 & 2cd+2ab \\ 2bd+2ac & 2cd-2ab & a^2-b^2-c^2+d^2 \end{pmatrix}$$

which yields the final transformation matrix

$$\mathbf{T} = \begin{pmatrix} & \mathbf{R} & & \mathbf{P} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[1] http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

Tracking issues

A number of factors influencing the performance of the optical tracking system have been identified and are described below.

Occlusion The problem occurs when one marker occludes another one or one or many markers are occluded by the work object or the teaching device itself. The effect of the problem can be reduced by

- Adding more cameras to the setup or adding more markers to the trackable.
- Arranging the markers so that occlusion is unlikely to occur.
- A mirror can be used to allow existing cameras to look from different angles. This requires development of a new tracking algorithm.
- Using a movable camera mounted on the robot.

Among the suggested alternatives, the two first are elaborated on further in section 4.2.

Symmetry When all visible markers are co-linear or exhibit some form of symmetry, the orientation of the trackable can not be determined and the output is corrupted. This causes sudden high amplitude changes in the coordinates reported by the tracker. Careful arrangement of the markers are needed in order to avoid this problem.

Noise Shiny areas of the teaching device, the work object or any other object within the tracker field of view, may lead to false positives. This may corrupt the output from the tracker. Use of active markers can reduce the amount of IR-light in the scene and thus reducing the number of false positives.

Smallest possible spacing between markers During tests with NaturalPoint active led markers, the smallest possible spacing between two LEDs that allowed them to be recognized as two distinct markers at normal working distance was 10mm. When the circles seen by the tracker overlap, they can no longer be distinguished from each other. The following modifications increase the circle diameter:

- Increasing the intensity of the active marker.
- Increasing the exposure time.
- Lowering the threshold.

This poses a constraint on the marker arrangements possible on the teaching tool.

Markers partially visible The tracking algorithm determine the location of a marker by calculating the center of mass. Thus, a partially occluded marker will have its center of mass shifted from the true center towards the still visible part, see discussion in section 4.2 and Fig. 4.2 on the next page. This effect causes a systematic error in the reported position.

4.2 Teaching tool

The optimal tool for teaching may differ from the tool normally used to do the task. Research on tool shapes for hammering [Schoenmarklin and Marras 1989] indicate that the shape of the tool has little or no significant impact on the performance, i.e. accuracy. These findings are of course not generalizable to all applications, they are however seen as an indication that the teaching tool can be designed with focus on usability and comfort. Findings also indicates that reliability, functionality and ease of use are considered more important by tool users compared to weight and tool size [Kuijt-Evers *et al.* 2004]. More findings on tool ergonomics can be found in appendix A.

A tool most people are familiar with is a standard pen. It is small, light, highly maneuverable and allows indication of flat or concave surfaces with high accuracy. Convex or sharp features however provide a challenge for the normal pen as it easily slips on a low friction surface. The lack of support for the hand operating the pen also reduces accuracy significantly.

To allow greater accuracy when indicating sharp features, a convex tool tip may be used. To reduce the risk of slipping, the tool tip may be coated with a high friction material such as rubber.

Markers

A marker is an object which either emits or reflects IR-light, visible to the tracking system. When using passive markers, the tracking system uses a strobing IR-light to illuminate the work space, whereas when active markers are used, the markers themselves emits the necessary light. Both approaches have some drawbacks which are discussed below.

Passive markers

- The reflectivity is reduced as the markers surface gets soiled.
- Markers that reflect in all direction are bulky.
- Passive markers are relying on external IR-illumination which results in a low SNR as other reflective surfaces may me treated as markers.

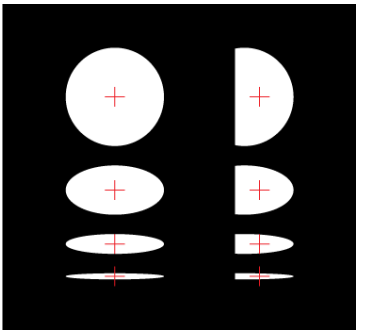


Figure 4.2: The location of a marker is defined as the center of mass [Seth 2012].

Active markers

- To be considered a marker, the shape of the light source has to be fairly circular as seen from the tracker, see Fig. 4.2. A ring shaped marker for instance is considered ”less ideal” by NaturalPoint [Birch 2012]. Constructing a spherical active marker that emits light uniformly in all directions is a challenge.
- Active markers require a power source.
- The light emitted by the marker can be reflected in shiny surfaces nearby, causing the tracker to see ghost markers.
- A teaching tool using active markers is more expensive to manufacture and it is harder to replace an active marker than a passive.

Marker arrangement

Ideally, the tip of the teaching tool is tracked as a separate marker. This allows a direct, high accuracy measurement of the contact point between the teaching tool and the surface. A marker at the tip is however easily occluded by the tool itself, the work object or the user, rendering the tip invisible. In a situation like this, an arrangement of a minimum of three other markers visible to at least two cameras can be used to calculate the position of the tip. An arrangement of four markers, where at most one marker is allowed to be occluded at any given time, for any given camera, is desirable. It is assumed that a marker located at the top of the teaching tool is always visible to all cameras. An example of such arrangement is shown in Fig. 4.3(a). This arrangement has been found to give consistently good results. No two distances between markers are the same, minimizing the risk of the tracking algorithm loosing track of the orientation.

(a) Illustration of marker arrangement

(b) Photo of 3D printed teaching tool prototype with active markers

(c) Illustration of intended usage

Figure 4.3: Teaching tool

Power source

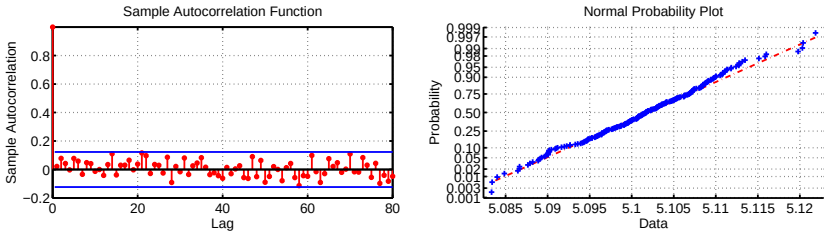
Since the teaching tool hosts active LED markers and plans exists on incorporating wireless communication, it requires an internal power source. Two alternative power sources have been considered and table 4.2 highlights some of the benefits and drawbacks of a super capacitor and a battery. Deciding factors when a power source was chosen was the low specific energy of the super capacitor and the requirement of a charge circuit for the lithium-based battery, together with the high availability and convenient form factor of the nickel-based battery.

	Super capacitor	Lithium-based Battery	Nickel-based Battery
Benefits	Fast charge (10s) High specific power Long lifetime	High specific energy	High number of cycles Removable **High availability**
Drawbacks	**Low specific energy** Linear discharge Low voltage	**Requires charge circuit** Slow charge	Needs periodic discharge High self discharge

Table 4.2: Benefits and drawbacks of available power sources [*Supercapacitor* 2012]. Deciding factors highlighted with bold face.

4.3 Filtering

By studying the output from the tracker when the trackables are at rest, it can be shown that the measurement noise acting on the system (**w** in equation 4.1) is approximately white and Gaussian distributed with zero mean. See Fig. 4.4. This,



(a) Noise auto correlation function with 95% confidence bounds. According to the findings, the noise is with 95% confidence white.

(b) Normplot of noise, suggesting a Gaussian distribution.

Figure 4.4: The noise acting on the system can considered white and Gaussian.

together with the linear dynamics of the system, suggests the use of a Kalman filter for smoothing and velocity estimation. Filtering is done before the measurements is transformed to the robot frame. A Kalman filter based on a simple state space model is used to estimate the true position and velocity. Filtering on the measurements is done separately for the coordinates x, y, z. The state-space model is given in discrete time as

$$\mathbf{x_{i+1}} = \mathbf{F}\mathbf{x} + \mathbf{w} \tag{4.1}$$

$$z = \mathbf{H}\mathbf{x} + v \tag{4.2}$$

$$\begin{pmatrix} x_{i+1} \\ v_{i+1} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ v_i \end{pmatrix} + \mathbf{w} \tag{4.3}$$

$$z = \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{x} + v \tag{4.4}$$

Outlier rejection

The output from the tracker may in some cases suffer from severe outliers. This is true for both orientation and position measurements, see Fig. 4.5. A measurement is considered an outlier if it is very far from the predicted value. It will then simply be replaced by the predicted value. Furthermore, the tracking algorithm will generate a not-a-number (NaN) if the trackable is not found in the frame. These values will be treated as a normal outlier and replaced by the predicted location at that time instance.

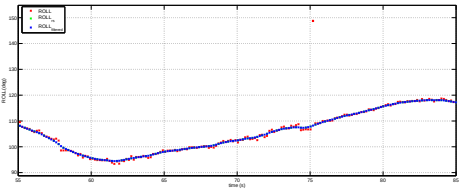


Figure 4.5: The measurement time series exhibits outliers. Here at $t \approx 75$s.

Kalman Smoothing

When state estimation is done offline, the minimum variance estimate is obtained from Kalman smoothing, as opposed to Kalman filtering. The Kalman smoothing algorithm uses knowledge of the measurements $z_1..z_N$ whereas the Kalman filter only uses measurements up to, and including, z_k. Use of the Kalman smoothing algorithm does not only improve the variance of the estimate, it also eliminates the phase lag. The filtering is carried out in two passes, where the first pass uses the normal Kalman filtering, and the second pass is done backwards on the filtered measurements. The algorithm used is described in detail in [Rauch *et al.* 1965].

In practical implementation of the Kalman filter, one often encounters problems with numerical instability. If the process noise covariance matrix is small, numerical round of errors can cause the positive definite process noise covariance matrix P to be indefinite, resulting in divergence of the algorithm. A solution to this is to save the matrix in the form $P = SS^T$ where S may be obtained by Cholesky factorization [Thornton 1976]. This representation prevents the matrix diagonal elements from becoming negative and the matrix from becoming asymmetric.

5

Scanning the work object

Conventionally, the CAD model of the work piece is used to program a robot offline. Due to the nature of the tasks an SME might be facing in a high mix, low volume operation, a CAD model is not always available. The lack of a CAD model may stem from the fact that none has ever existed, the object has suffered wear and is not in the same condition it was when it was new, or the CAD model is not supplied by the work piece manufacturer. Creating an accurate CAD model is time consuming and costly and often unfeasible for very low volume operation. From now on, no CAD model is assumed to exist.

5.1 Scanner placement

When the location of the scanner is chosen, a few things must be considered. In order to minimize the constraints on where to locate the work piece while conducting the teaching, the scanner is mounted on the robot end effector, see Fig. 5.1. This allows the scanner to move and capture the work piece surface from multiple angles, ensuring that no relevant sub surface is missed. Another feature of a mobile scanner is that the distance between the scan surface and the scanner can be controlled. Since both scan accuracy and resolution decreases with distance as $\propto 1/d^2$, minimizing the distance between the scanner and the surface is essential for a high quality scan result. Furthermore, all optical lenses

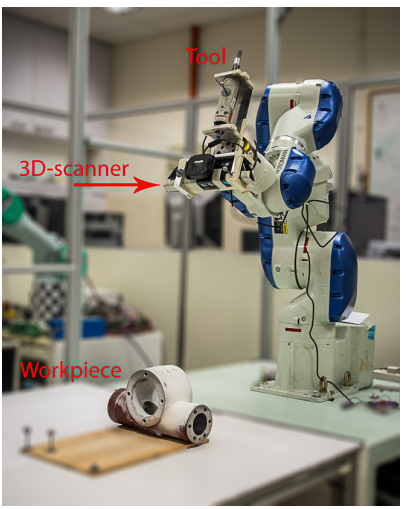


Figure 5.1: End effector assembly

suffers from various kinds of distortion.
The distortion is typically heavier close to the edges of the image plane. The ability to relocate the scanner automatically opens up the possibility to use only the center of the scan, thus cropping the areas suffering from the heaviest distortion.

Other possible arrangements include keeping the scanner fixed and moving the work piece in between scans, for instance by using a turn table. This solution would be cumbersome if the work piece is large or heavy and impossible if it was static, such as a wall. It would also be significantly harder to automate.

A third possibility is to let the scanner be hand held. This is by definition a manual process and the result would be dependent on the user operating the scanner.

Since the coordinates of all scans are given in the scanner frame, the location and orientation of this frame must be known in order to transform the point cloud to the tracker frame. This is possible to achieve by tracking the scanner using the optical tracking system. Since every measurement of the location of a trackable is associated with a small amount of error, this transformation will always be inexact. The most serious type of error is in the measured orientation. Since the scanned surface is more than 50cm away from the scanner, a small error in the orientation will through the lever effect amplify to a large error at the distance where the workpiece is located. To enhance the accuracy with which the orientation is estimated, it is vital to configure the markers with great separation, such that the distance between the markers is in the same order of magnitude as the distance between the markers and the scanned surface. The error can also be reduced by introducing redundant markers in the trackable that represents the scanner position. The redundancy will have an averaging effect on the position and orientation measurement thus canceling some of the error in an individual marker.

5.2 Finding scan poses

When the path is recorded, suitable target poses must be found in order to capture all relevant surface information. This is a research topic in itself that involves calculating the optimal viewpoints based on robot reachability and information from previous scans. A simplified approach is adopted where the workpiece is assumed to be oriented in such a way that a small number of scan poses are sufficient to capture the surface. To generate a target frame for the scanner, the vector

$$\mathbf{R}_b = \mathbf{P}_T^{RB} - \mathbf{p} \tag{5.1}$$

is generated. It is simply the vector between a suitable point on the path $\mathbf{p}$, such as the path center of mass or the center of a segment, and the location of the robot in the tracker frame. The vector $\mathbf{N}$ is the normal of the path at the point $\mathbf{p}$, obtained either from a previous scan or as the third principal component of the path. A target

frame for the scanner is generated as

$$T_T^S = \begin{pmatrix} \text{-}\mathbf{R}_b \times \mathbf{N} & \mathbf{R}_b & \text{-}\mathbf{N} & \begin{matrix} 0 \\ \text{offset} \\ 0 \end{matrix} \\ 0 \quad 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.2}$$

where the offset is the desired distance between the path segment and the scanner, which for the Kinect is $\approx$ 50cm. The target frame is then then transformed to a target for the tool frame in the robot base coordinate system as

$$T_{RB}^{TF} = T_{RB}^{T} \, T_T^S \, \left(T_{TF}^S\right)^{-1} \tag{5.3}$$

The equation 5.2 is constructed so that the vectors involved forms a transformation matrix between the tracker frame and the desired target scanner frame.

5.3 Putting the pieces together

When the robot has moved to a given scan pose, a measurement of the scanners position is made by the tracker. This way, no information from the robot, which is assumed to be less accurate than the information from the tracker, is used. When the first scan is obtained, it is transformed to the tracker frame for fusion with the recorded path. All subsequent scans are first transformed to the frame of the initial scan in order to avoid later problems arising from uncertainties in the calibrated matrices relating the scanner to the tracker frame. The transformation between the scanner frame and the tracker frame is given by

$$T_T^S = T_T^{Wnew} \left(T_T^{Wold}\right)^{-1} T_T^{TCP10} \, (T_{TCP0}^{TCP1})^{-1} \, (T_{TF}^{TCP0})^{-1} \, T_{TF}^S \, (T_T^{TCP10})^{-1} \, T_T^{TCP1i} \tag{5.4}$$

Since the scanner is rigidly attached to the tool frame and the tool, a measurement of the tool position (TCP1) gives the position of the scanner through the transformation 5.4

5.4 Calibrating the Kinect

In order to calibrate the transformation T_{TF}^S, an initial guess is required. The scanner is rigidly attached to the tool frame and a reasonable transformation matrix can be obtained through inspection and some basic measurements. In our case, the initial guess is as simple as

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -110 \\ 0 & -1 & 0 & 50 \end{pmatrix}$$

This initial guess is unlikely to be sufficiently accurate, unless the CAD model for the scanner and its mount is known. By scanning a suitable object, on which a few distinct features are indicated and recorded by the tracker, a refinement of the guessed transformation matrix can be obtained. Aligning data from the tracker to the scanned point cloud is discussed further in section 6. The result of the alignment is a transformation matrix T_{ICP}, which transforms the scan from its original position to better fit the indicated features. The desired transformation however is one that can be applied to T_{TF}^{S} in order to improve upon it. The equation to transform a scan from the scanner frame to the tracker frame, equation 5.4, can be simplified to

$$T_{T}^{S} = T_{T}^{TF}\, T_{TF}^{S} \tag{5.5}$$

where T_{T}^{TF} is a composite of matrices applied in front of T_{TF}^{S}. The transformation T_{ICP} is then multiplied from the left, to improve the location of the already transformed scan. The desired matrix, T_B, that improves the initial guess is then given by

$$T_{ICP}\, T_{T}^{TF}\, T_{TF}^{S} = T_{T}^{TF}\, T_B\, T_{TF}^{S} \tag{5.6}$$

$$T_B = (T_{T}^{TF})^{-1}\, T_{ICP}\, T_{T}^{TF} \tag{5.7}$$

$$(T_{TF}^{S})_{new} = T_B\, T_{TF}^{S} \tag{5.8}$$

Equation 5.8 forms the new and improved calibration matrix.

5.5 Point cloud processing

When all scans are recorded and saved, the composite scan is processed for information such as normals and curvature, which will be used to enhance the path and supply it with suitable orientation information.

Outlier removal

Each scan from the Kinect contains $640 \times 480 \approx 300\,000$ points. This large number of points requires a substantial amount of time to process, why some method of reducing the amount of data is desired. Since the path is already recorded at the time of scanning, the information recorded can be used to sort out and remove all points in the scan located far from the path.

The data recorded by the scanner is also corrupted by noise due to specular reflections and quantization errors et.c. The effects caused by specular reflection can cause severe outliers in the data set which must be handled before further processing. The outlier rejection is carried out in two steps. First, a conditional removal filter is applied. All points that does not have a specified number of neighboring points within a certain radius is considered outliers and removed[1]. The next step is

[1] `http://pointclouds.org/documentation/tutorials/remove_outliers.php#remove-outliers`

a statistical outlier removal filter. During this step, the mean distance in the neighborhood of all points is calculated. All points for which the mean distance is greater than the global mean plus some specified constant times the global standard deviation will be considered outliers and removed[2].

$$(M_{neighborhood} > M_{global} + c \cdot \sigma_{global},\ c \in \mathbb{R}) \rightarrow outlier$$

The outliers that were removed previous to applying the statistical outlier rejection filter would increase the variance estimate, thus masking modest outliers which would result in a less effective filtering.

Filtering and normal estimation

The point cloud obtained from the scanner is corrupted by noise from a number of sources. One major error source is the misalignment occurring when different scans are combined using tracker measurements of the scanner position. Other error sources are intrinsic to the Kinect and give rise to both random white noise and correlated noise, described in [Khoshelham 2011].

The noisy output from the Kinect is filtered through a moving least squares filter[3] [Pauly *et al.* 2002], which fits a surface to a local neighborhood of each point and projects the point onto the surface. The fitted surface is also used to calculate the local surface normal. Parameters for the filter include the radius within which to look for neighbors when fitting the surface. A large radius will suppress noise more while also reducing sharpness of features such as edges. The normals estimated by the filter are ambiguous in the sense that the filter is unaware of which side of the surface is the outside, and the normals are therefore not guaranteed to be facing outwards. To solve this, the location of the origin of the scanner is saved and all points for which the dot product between the normal and the vector between the point and the origin are smaller than zero, the normal is flipped.

Other techniques for normal estimation exists. However, since the MLS-filter is applied for other reasons than normal estimation, no extra time consuming computations must be carried out when using this method.

Curvature estimation

Information such as curvature direction and magnitude can be estimated using PCA [Rusu 2010]. The curvature information will help determine which points in the cloud that belong to an edge or crease. An edge point is characterized by a high curvature. Knowing which points belong to an edge will prove useful for later fusion of the scan and the path.

[2] `http://pointclouds.org/documentation/tutorials/statistical_outlier.php#statistical-outlier-removal`

[3] `http://pointclouds.org/documentation/tutorials/resampling.php`

6

Fusion of scan and tracker data

When the scan is obtained and processed, the information about the workpiece geometry can be used to augment the path. This chapter will describe the steps taken to align the two data sources, reconstruct the path and assign target frames along the path.

6.1 Procedure outline

Different approaches for fusion of the recorded data sets are needed since the nature of the application may vary. Edge deburring and surface spray painting requires different information to be extracted from the data sets. This work will focus mainly on contact tasks with a hard tool. Roughly, the procedure is outlined below

1. Ensure rough initial alignment of surface point cloud and path. This should be satisfied if the calibration between the scanner and the tracker is sufficiently accurate. Alignment is then improved using software algorithms.

2. Divide the recorded paths into segments and classify each segment.

3. Detect and classify surface types along user indicated path.

4. Enhance the structure of the user indicated path based on the structure in the surface.

5. Construct the robot trajectory.

6.2 Re-sampling

The path is sampled at a fixed time interval. This is good for estimation of the velocity of the users hand movements. The dependence on the users hand velocity, however, means that points are not equidistant in space. In order to improve the performance of algorithms used to process and enhance the path, both in terms of processing time and output quality, a re-sampling is conducted. Except for the sequence of the recorded points, no temporal information is used in subsequent algorithms, therefore, re-sampling done in the spatial domain is possible.

In order to re-sample the path so that points are equidistant in space, a naive approach algorithm is implemented. The algorithm assumes that the new sampling distance is sufficiently small in order to preserve all desired detail. The Kalman-filter described in section 4.3 will ensure that no aliasing will occur as a result of the re-sampling. From the first recorded point, p_0, the algorithm searches through subsequent points until the distance between p_0 and the currently investigated point, p_i, is greater than the sampling distance. The new point recorded in the re-sampled path is then constructed by a linear interpolation between p_i and p_{i-1} as

$$r = (p_i - p_0) - d \tag{6.1}$$

$$p_{rs} = p_{i-1}r + p_i(1-r) \tag{6.2}$$

See Fig. 6.1. After a re-sampled point p_{rs} is constructed p_0 is set to p_{rs}. This simple algorithm will ensure that re-sampled points are spaced at an interval d, under the assumption that the path is a straight line from p_0 to p_i. This is a good approximation if d is small.

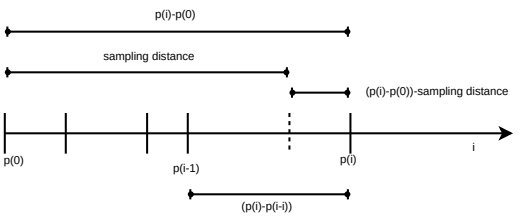


Figure 6.1: Resampling in space domain.

Practical experience indicates that the number of recorded points can be reduced by a factor 3-5 without sacrificing detail. This large reduction will reduce processing time in for instance nearest neighbor searches by roughly the same factor. Nearest neighbor searches between the path and the surface are the most time consuming task when the fusion, described in chapter 6, is carried out.

As can be seen in Fig. 6.2, the algorithm results in a distance between two consecutive points very close to the desired sampling distance.

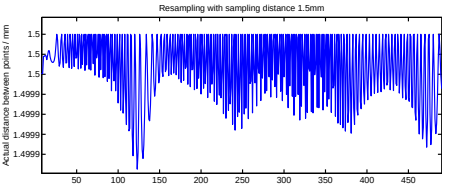


Figure 6.2: Re-sampling result using 1.5mm sampling distance. The plot shows the distance between two consecutive points for a re-sampled path.

6.3 Path segmentation and classification

In order to further process the path, an important first step is to determine what type of path the user has indicated and how many segments it contains. The process of enhancing a straight segment differs somewhat from the task of enhancing a curved one.

Segmentation algorithm

Two different approaches to path segmentation is proposed and evaluated. One based on Principal Component Analysis and one based on differentiation.

PCA approach Based on the assumption that the path is locally two dimensional, principal component analysis (PCA) [Pearson 1901] is used to transform the coordinates of the data points to the principal component space, where the third principal component is assumed to be of negligible magnitude. The user specifies the radius within which the algorithm uses points to conduct the PCA. The PCA will reveal how much of the variance in each neighborhood is explained by the individual principal components, where the first component explains the greatest amount of variance. For a straight line, all variance will be explained by the first component, whereas for a curved path lying in a plane, the variance explained by the first component will be

$$\text{Fraction of Variance Explained: } FVE = \frac{V_{pc_1}}{V_{pc_1} + V_{pc_2}} < 1 \tag{6.3}$$

Negative peaks in the FVE indicates potential breakpoints in the path. The specified radius will influence the depth of the peaks in the FVE, allowing the user control over the outcome.

If the path truly was located in a plane, the variance explained by the third principal component would be zero. This scenario is however unlikely both due to noise in the recording and involuntary hand movements by the user. The variance described by the third component is therefore considered as noise and the denominator in the expression for the FVE considers only the contribution from the first two components.

Differentiation approach Based on a simple differentiation of the path $d_i = p_{i+1} - p_1$, one can locate break points in the path by calculating the dot product between two consecutive directions, $d_i^{\mathsf{T}} d_{i+1}$. Negative peaks in this dot product lower than some threshold indicates the presence of a break point.

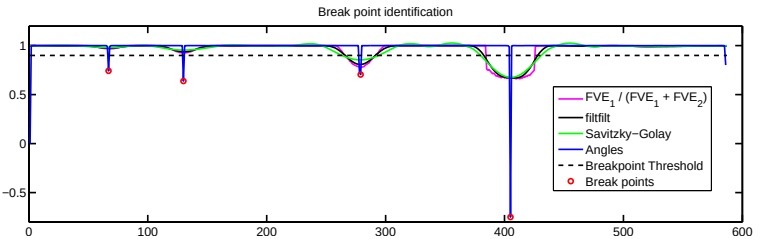


Figure 6.3: Result of break point identification using different approaches. The result of the PCA-approach is shown filtered using both a moving average low-pass filter and a Savitzky-Golay filter.

Experimental results As can be seen in Fig. 6.3, the differentiation approach (blue line) is a stronger indicator of a break point in the path, it is however not dependent on the user specified radius. The differentiation approach's superior performance in detecting even subtle break points is considered more important, why this approach will be adopted.

Classification

Two classes of path segments are considered, straight and curved. To determine the class of a segment, two approaches are investigated.

Fraction of Variance Explained-approach The FVE, described in section 6.3, equation 6.3, can be used as an indicator to whether a path is straight or curved. If the fraction of variance explained by the second principal component, FVE_2, is greater than some threshold, the path is considered curved, otherwise straight. Once again, FVE_3 is considered a result of noise and disregarded.

Function fitting approach This approach will fit two analytical functions to the segment under consideration, a straight line and a circle. The fit is then analyzed and the norm of the residuals is used as an indicator of segment class. The idea is that if the segment is curved, the norm of the residuals should be much higher for the first degree polynomial than for the circle. For a straight line however, the difference in fit is negligible. If the segment is very complicated, none of the functions will fit very well and the segment will be treated as curved.

Experimental results Experience indicates that both approaches do well in most cases. There are however cases when one method fails to indicate a straight segment

while the other one succeeds and vice versa. None of the methods fail to properly classify a curved segment. This is solved by using both classifiers and considering a segment straight whenever any of the methods indicate a straight segment. This combined approach, while admittedly inelegant, has been found to have a high success rate, where success is defined as the output of the algorithm being the same as the kind of segment intended.

6.4 Registration

Registration is the act of finding a rigid transformation, rotation $R \in SO(3)$ and translation $P = (x, y, z)$, between two set of points so that corresponding points in the two sets are mapped to the same location. Several successful algorithms already exists for registration of two partially overlapping point clouds. A large subset of these try to estimate key features in the data which are unique and can be matched between the two point clouds, see for instance [Johnson and Hebert 1997]. Another category of algorithms tries to minimize some distance measure between all points in the two data sets, a widely used example is the ICP algorithm described in section 6.4 [Besl and McKay 1992].

Our problem is fundamentally different from the problems treated in the above mentioned works. A point belonging to a surface may be associated with two principal directions and two corresponding principal curvatures, i.e. a tangent plane spanned by the two principal directions. This allows for instance normal estimation. A point on a path however may only be associated with one principal curvature and direction, i.e. a tangent line. All methods using for instance normal estimates or features calculated from two principal directions will therefore fail when registration of a path is considered. Other problems related to the registration of a path and a surface is elaborated on in section 6.4.

Iterative Closest Point, ICP

The Iterative Closest Point algorithm is widely used for registration of point data sets and many variants of and extensions to the original algorithm exits. The following description of the ICP algorithm follows the one in [Besl and McKay 1992].

The distance from a point $\mathbf{p}$ to a data set X is given by $\min_{\mathbf{x}\in X} \|\mathbf{x} - \mathbf{p}\|$. Given a target data set X with N_x points and a input data set P with N_p the optimization objective function to be minimized is

$$f(\mathbf{T}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|\mathbf{x}_i - \mathbf{T}\mathbf{p}_i\|^2 \tag{6.4}$$

where $\mathbf{T}\mathbf{p}_i$ may be expressed as $\mathbf{R}(\mathbf{T}_R)\mathbf{p}_i - \mathbf{T}_T$. $\mathbf{T}_R = [q_0, q_1, q_2, q_3]^\mathsf{T}$ is a quaternion vector with corresponding rotation matrix $\mathbf{R}(\mathbf{T}_R)$ and $\mathbf{T}_T$ is a translation vector, $\mathbf{T} = [\mathbf{T}_R|\mathbf{T}_T]$. The optimization seeks to minimize the sum of square distances to

the target data set X from all points in the data set P, by finding the optimal rigid transformation matrix $\mathbf{T}$.

For the data sets X and P, the cross covariance $\mathbf{C}_{px}$ and respective center of mass μ_X, μ_P is computed. From the matrix

$$\mathbf{A} = \mathbf{C}_{px} - \mathbf{C}_{px}^{\mathsf{T}}$$

a vector $\Delta = [\mathbf{A}_{23}\ \mathbf{A}_{31}\ \mathbf{A}_{12}]^{\mathsf{T}}$ is constructed. The optimal rotation $\mathbf{T}_R$ is the given as the eigenvector corresponding to the maximum eigenvalue of

$$\mathbf{Q}(\mathbf{C}_{px}) = \begin{pmatrix} \mathrm{tr}(\mathbf{C}_{px}) & \Delta^{\mathsf{T}} \\ \Delta & \mathbf{C}_{px} + \mathbf{C}_{px}^{\mathsf{T}} - \mathrm{tr}(\mathbf{C}_{px}) I_3 \end{pmatrix}$$

The optimal translation is given by

$$\mathbf{T}_T = \mu_x - \mathbf{R}(\mathbf{T}_R)\mu_p$$

For the proof that these transformations indeed are optimal, the reader is referred to [Horn 1987]. The discussion in [Horn 1987] starts with a proof of the optimal rotation when perfect measurements of three points in two coordinate systems are to be aligned. The proof is then expanded to a least squares solution, where the sum of squared residuals are minimized, and further expanded to include any number of points. The key point is the matrix $\mathbf{Q}$, which contains all the information needed to find the least squares solution.

At each iteration step, the set of closest points and the $\mathbf{Q}$ matrix above is calculated, after which the transformation is applied. The algorithm is terminated when the difference in the objective function value between two consecutive iterations is lower than some threshold.

The reader may refer to [Besl and McKay 1992] for analysis of convergence, where the author provides a theorem of convergence to a local minimum. To ensure convergence to a global minimum, a number of different initial transforms is tested and the one resulting in the lowest value of the objective function is chosen as initial condition. This is a time consuming approach and if global convergence is to be ensured, a large number of initial transforms must be considered.

The algorithm, as presented in [Besl and McKay 1992], is able to successfully register two sets of points together. The points may be sampled from surfaces, curves, triangles or line segments. Registration of a surface and a curve however may result in an ambiguous result. Consider for example the case depicted in Fig. 6.4. The recorded path will be well aligned to the outside of the cylinder. However, any placement of the circular path along the side of the cylinder is equally optimal.

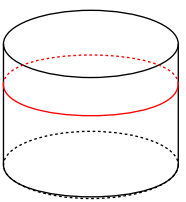


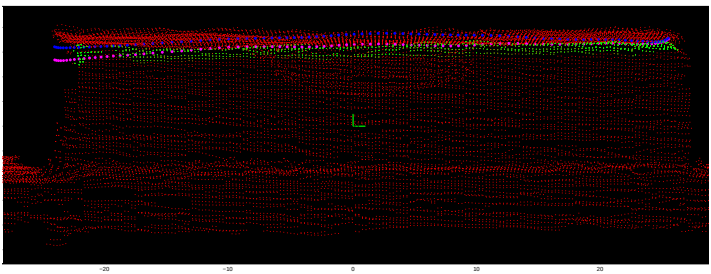
Figure 6.4

In [Feldmar *et al.* 1995], the authors have presented a way to successfully register projections of 3D surfaces with

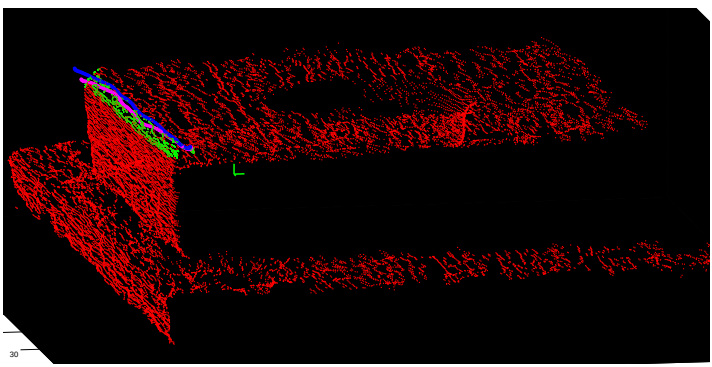
2D curves. In this case, the curves are the edges or contours in 2D projections of the surface and the registration aims to find the rigid transformation that aligns these projections of the surface with a given set of 2D contours.

Evaluation of ICP algorithm

When using the ICP algorithm to align a path to a surface, the outcome is highly dependent on the ability to extract the feature points one wish to align the path to. Below, experimental results are presented in form of Fig. 6.5. The scenario depicted is registration of a recorded path and surface containing an edge, acquired from the David laser scanner.



(a) XZ-view



(b) Tilted-view

Figure 6.5: Registration of path and edge using ICP. Blue paths are roughly aligned, purple paths are aligned to the green edge points using ICP.

As can be seen in the figures, the recorded path is not straight. This might be due to noise in the tracking system, but the main source of noise comes from the users shaky hand motion. The alignment, as can be seen in Fig. 6.5(a), does not follow the edge perfectly.

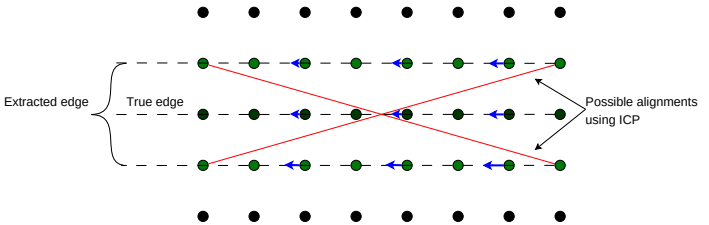


Figure 6.6: Illustration of problem when using ICP for registration of path and edge. If the edge is not perfectly extracted, several alignments of a path might be possible. The directions, indicated by the blue arrows, however, are the same for all points close to the edge.

This procedure requires knowledge of which points on the surface that belongs to the edge. For this purpose, algorithm 6.4.1 is implemented.

Estimate principal curvature for all points on surface.
$curv\_sorted = \text{sort}(curvature)$; ▷ Sort values in ascending direction.
$threshold = curv\_sorted(\text{round}(\text{length}(curvature) \cdot 0.85))$; ▷ Select 15% of the points based on highest curvature.

Algorithm 6.4.1 Pseudo code for extracting an edge from a surface along a path.

Rotational alignment of straight lines

When two straight lines are to be aligned, there is one DOF, corresponding to the rotation around the line axis, that cannot be determined. The ICP algorithm will indeed align the lines perfectly, but no guarantee is given for how the last DOF is treated. To resolve this, the following algorithm is implemented.

The direction of the individual lines is determined by their first principal component. This vector will be called $\mathbf{d}_1$ and $\mathbf{d}_2$, for line one and line two respectively. To find the rotation matrix that aligns the two lines, a vector orthogonal to both $\mathbf{d}_1$ and $\mathbf{d}_2$ is created as $\mathbf{k} = \mathbf{d}_1 \times \mathbf{d}_2$. This vector corresponds to the axis of rotation. The angle of rotation is given by the formulation for a vector dot product

$$\mathbf{d}_1 \cdot \mathbf{d}_2 = \|\mathbf{d}_1\| \, \|\mathbf{d}_2\| \cos\alpha \tag{6.5}$$

which when solved for α yields

$$\alpha = \arccos \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{\|\mathbf{d}_1\| \, \|\mathbf{d}_2\|} \tag{6.6}$$

The rotation matrix is then constructed from Rodrigues' rotation formula[1]

$$\mathbf{R} = \mathbf{I} + \sin\theta[\mathbf{k}]_\times + (1-\cos\theta)(\mathbf{k}\mathbf{k}^\mathsf{T} - \mathbf{I}) \tag{6.7}$$

where $[\mathbf{k}]_\times$ denotes the cross product matrix form of k, such that

$$[\mathbf{k}]_\times \mathbf{v} = \mathbf{k} \times \mathbf{v} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \mathbf{v} \tag{6.8}$$

the translation between the lines is obtained by the vector between the respective lines center of mass.

Transformation to direction space

To remedy the problem with the ICP algorithm, depicted in Fig. 6.6, a new registration method is investigated.

The recorded path consists of a series of coordinates in $\mathbb{R}^3$. Based on the sequence of coordinates, every point on the path can be associated with a direction estimate. These direction estimates can be plotted together with the direction estimates from the corresponding surface patches, see Fig. 6.7(b). This, of course, requires directions in the surface to be defined and known. This mapping from point coordinates to directions will be called a transformation from location space to direction space.

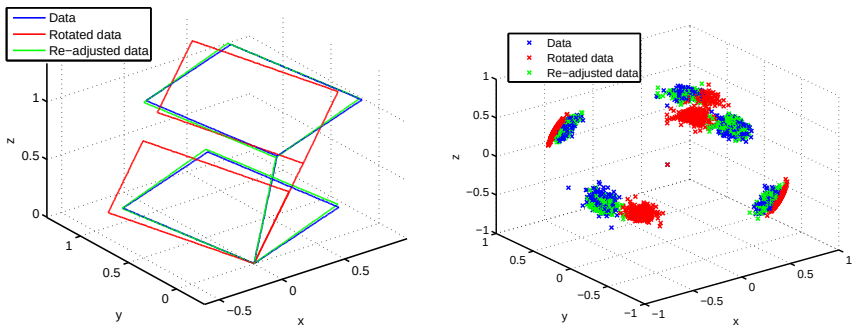
A direction in a point on the surface is for our purposes defined as the vector cross product between the normal and the curvature principal direction at the point, ($\mathbf{d}_s = \mathbf{n} \times \mathbf{c}$). On a planar surface, the described direction estimate is unlikely to be exactly zero, but since the curvature is low, these points can easily be disregarded. A scanned surface is likely to contain several different directions. Therefore it is essential to extract directions only from parts of the surface that corresponds to the path.

The set of estimated directions is likely to contain a significant amount of outliers, raising the need for pre-processing with outlier removal. A simple algorithm is implemented where all points for which the distance to the n:th closest point is greater than a multiple of the standard deviation among all distances between a point and its n:th nearest neighbor is removed. This can be interpreted as all points that does not have n or more neighbors within the radius $r = c\sigma, c \in \mathbb{R}^+$ are considered outliers.

This procedure is invariant to both translations and scale factors between the two data sets to be aligned, and can thus only align them rotationally. To realize this, consider two squares in $\mathbb{R}^2$ with sides a and $2a$ respectively. Both share one corner point and the direction of the diagonal from that point. Even though the two squares are on a different scale, all sides are parallel. All direction estimates will

[1] http://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula

therefore be equal. Consider next the small square centered in the large square, corresponding to a translation of the square from the previous case. This does not change the fact that all sides are parallel and thus mapped to the same points in the direction space.



(a) Simulated data. The second data set (red) is a rigid rotation of the first (blue). Both data sets are corrupted with uncorrelated gaussian noise. The third data set (green) is aligned using the transformation acquired from ICP.

(b) The first two data sets (blue, red) represent directions estimated from data in (a). The third data set (green) is an aligned version of the second, using ICP.

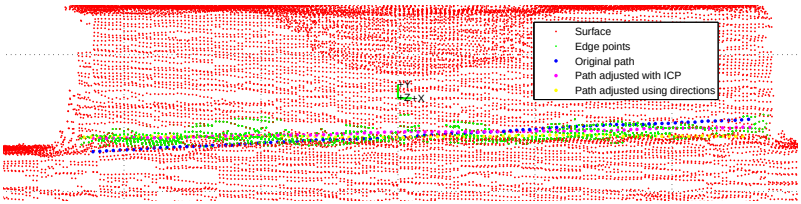
Figure 6.7: Procedure for rotation alignment of path and surface.

Existing registration algorithms such as i.e. ICP [Besl and McKay 1992] work very well to align two point clouds together. For registration of a path and a surface they fail however. In Fig. 6.7(b), the direction estimates from a synthetic data set is shown. Direction estimates from a surface point cloud and a path are more similar in nature to the normal input of the ICP algorithm and can easily be aligned using such existing registration algorithms.

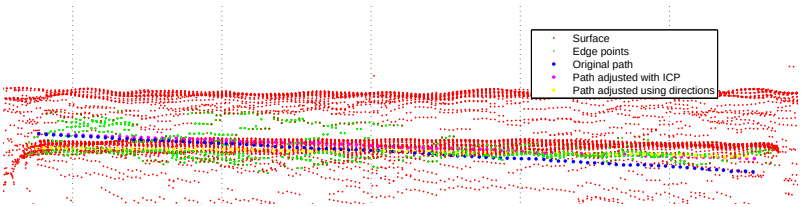
Experimental results

The data sets used below are synthetic, unless otherwise noted. The surface data set was corrupted with gaussian white noise and the path was corrupted with colored low frequency noise to simulate unsteadiness in the users hand movements. The standard deviation of the noise added was 1% of the model scale. Based on experience from using the david laser scanner and input from the tracker, these noise levels can be considered realistic. After creation of the data sets, an MLS-filter was applied and normals and curvature was calculated using PCL. Further processing was done in Matlab. For some data sets, the mean squared error (MSE) was calculated based on the correct path and the path obtained through registration.

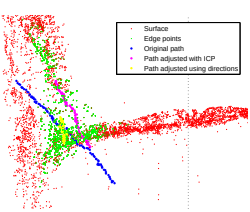
$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left\| p_i^{aligned} - p_i^{optimal} \right\|^2$$



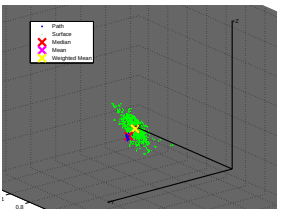
(a) XY-view



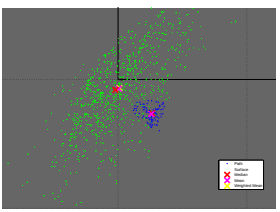
(b) XZ-view



(c) YZ-view



(d) Directions Overview



(e) Directions Close up

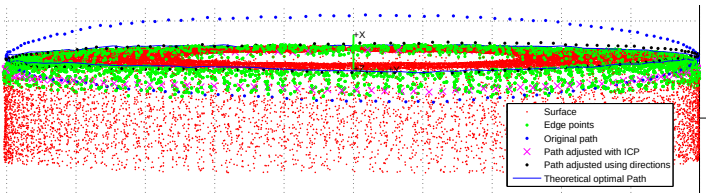
Figure 6.8: Test with real surface and simulated straight path. The task is to align the path to a crease in the scanned surface.

Fig. 6.8 shows the result of applying the algorithm to a crease in a real data set, obtained from the david laser scanner (non synthetic). In particular Fig. 6.8(c) illustrates the benefit of using the direction space method. Figures 6.8(d)-6.8(e) depicts the direction estimates for the edge points in the surface (green) and the path (blue).

In figures 6.9-6.10, results are shown after running the algorithm in a scenario where a circular path is to be aligned with the edge of a cylinder. For comparison, the result of running the ICP algorithm using extracted edge points, is shown. The estimated directions is shown in Fig. 6.10. For the direction approach, the error metric was $MSE_d = 0.57$ and for the ICP algorithm with edge points in location space $MSE_l = 0.59$.

Figures 6.11-6.12 shows the result of applying the algorithm to two edges of a sythetic cube. For the direction approach the error metric was $MSE_d = 0.26$ and for the ICP algorithm with edge points in location space $MSE_l = 1.57$.

Figures 6.14-6.13 shows the result of applying the algorithm to a free form



(a)

Figure 6.9: Test with circular edge.

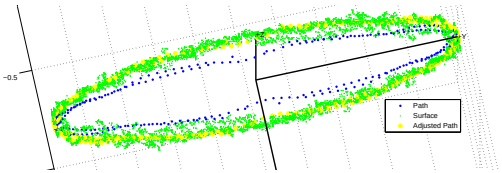
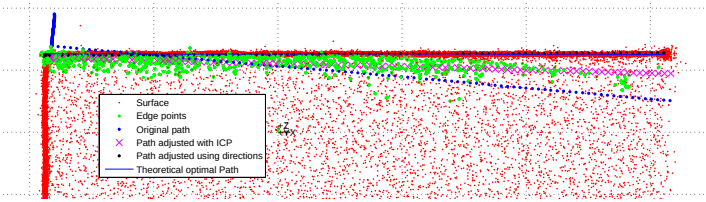
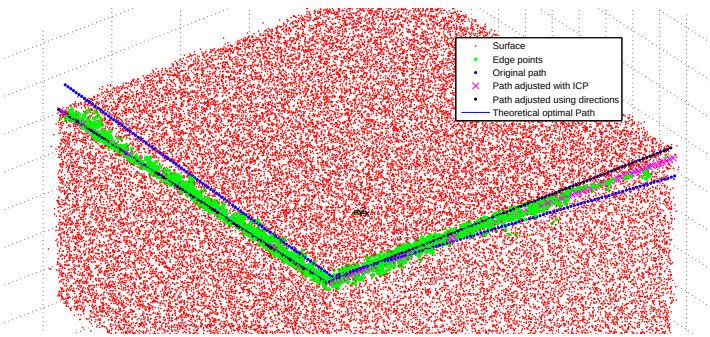


Figure 6.10: Estimated directions in path and surface used for registration of cylinder and circular edge.



(a) XZ-view



(b) Tilted view

Figure 6.11: Test with two perpendicular edges.

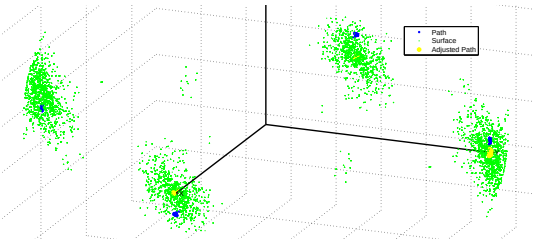


Figure 6.12: Estimated directions in path and surface used for registration of cube and two perpendicular edges. Adjusted path directions are aligned using modified ICP.

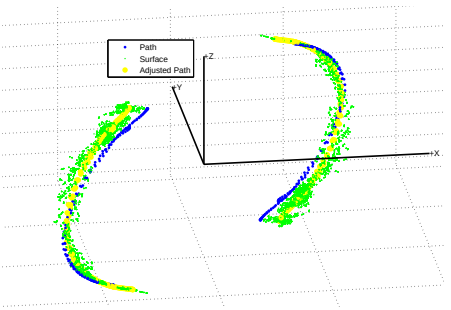
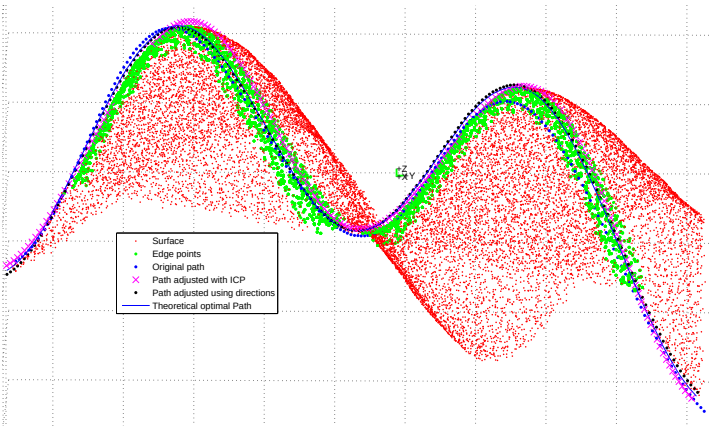


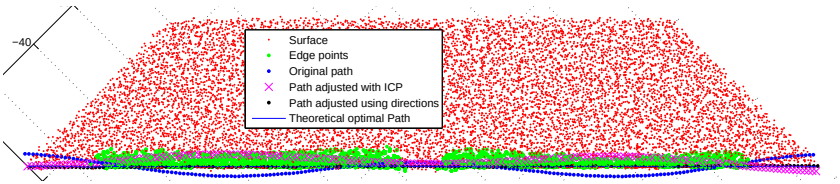
Figure 6.13: Estimated directions in path and surface used for registration of free form edge. Adjusted path directions are aligned using modified ICP.

Data set	Noise level	MSE_d	MSE_l
Cylinder	1%	0.57	0.59
Cube	1%	0.26	1.57
Free form surface	2%	0.45	1.22
Free form surface	5%	1.18	2.17

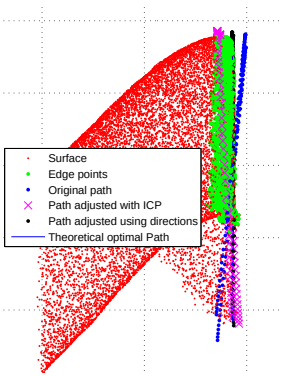
Table 6.1: Calculated error metrics for different data sets.



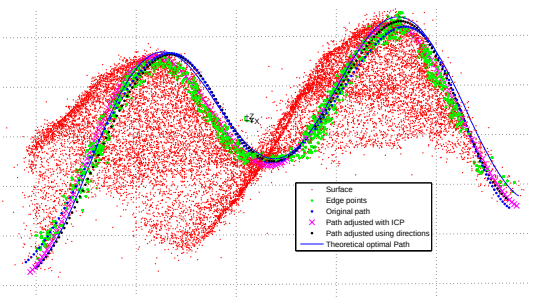
(a) 1% noise



(b) 1% noise



(c) 1% noise



(d) 5% noise

Figure 6.14: Test with free form edge.

edge. Different noise levels was used to corrupt the surface and the error metrics calculated are presented together with the rest of the error metrics in Table 6.1.

Although results are positive for both synthetic data with added noise and real data from the David laser scanner, experience shows that results using the Kinect scanner are less robust. The Kinect scanner exhibits significantly more noise and less point density than the David laser scanner and no improvement to the registration can be made using the direction space method. Since the direction space algorithm works only on edges and creases and not on flat surfaces, it was decided that the more general ICP algorithm alone was sufficient.

6.5 Path reconstruction

The user indicated path is prone to errors due to for instance hand shake and mistakes made by the user. A major benefit of having a 3D scan of the work object surface is the possibility of enhancing or reconstructing the recorded path based on the structure of the scanned surface.

After pre-processing, outlined in algorithm 6.5.1, the path is assumed to consist of a single, smooth segment with equidistant points. Reconstruction starts by a choice of a parameter r called the reconstruction radius. For each point p_i on the path, neighboring points within distance r will be used used in the reconstruction of p_i. The set of points within the reconstruction radius will be termed the path reconstruction neighborhood N_P. Each point will also have a set of nearest neighbors in the surface. If a large fraction of the nearest neighboring surface points to the current path segment is considered to be edge points (points with high curvature), the path segment is considered to be following an edge. Nearby edge points will then be chosen as the surface reconstruction neighborhood N_S. If no edge is present in the surface, N_S will consist of the k nearest neighbors in the surface to all points $\in N_P$.

Re-sample path to ensure points are equidistant;
Divide path into segments if sharp features are present;
Classify segments as straight lines, circles or curved paths;
for all *Segments* **do**
  if *segment* is straight line or circle **then**
    continue; ▷ These cases are handled separately.
  end if
  Determine if *segment* follows edge based on surface curvature;
  for all points p in *segment* **do**
    Establish N_P and N_S;
  end for
end for

Algorithm 6.5.1 Pre-processing outline

Using the established N_P and N_S, the reconstruction will proceed as follows. A PCA will be carried out on all points in N_P and N_S respectively. The analysis will yield two orthonormal matrices C_P and $C_E \in \mathbb{R}^{3\times3}$, whose columns correspond to the principal components of the respective data sets. The PCA will through the matrices C_P and C_E transform the data points p_i to an orthogonal principal component space $\hat{p}_i$ as

$$\hat{p}_i = C^{-1}(p_1 - \mu) \tag{6.9}$$

where μ is the center of mass for the data set currently analyzed.

The transformed points are then orthogonally projected to the subspace spanned by the first two principal components by setting the third coordinate to zero. Under the assumption that the data is planar, the third principal component holds no relevant information and only noise is lost in the projection.

A polynomial of low degree is then fitted to the projected points. For a single data set, a reconstructed point $\tilde{p}_i$ is formed by the projection of $\hat{p}_i$ onto a fitted polynomial as

$$\begin{aligned} f &= \text{Polynomial fitted to points } \in N \\ \tilde{p}_i &= (\tilde{p}_i^x, \tilde{p}_i^y, \tilde{p}_i^z) \\ &= \mu + C^x \hat{p}_i^x + C^y f(\hat{p}_i^x) \end{aligned} \tag{6.10}$$

where the super-script $p^x \in \mathbb{R}^1$ denotes the first component of $p = (p^x, p^y, p^z)^\mathsf{T}$ and $C^x \in \mathbb{R}^{3\times 1}$ denotes the first column of $C = (C^x\ C^y\ C^z)$. The operation

$$p_i = C\hat{p}_i + \mu \tag{6.11}$$

transforms a point in principal component space back to Cartesian space. The term

$$C^y f(\hat{p}_i^x) \tag{6.12}$$

forms the second component of the reconstructed point by evaluating the polynomial and transforming the result back to the original space.

To allow the use of both information from the path and the surface in the reconstruction, equation (6.10) is extended to

$$\begin{aligned} \tilde{p}_i &= (\tilde{p}_i^x, \tilde{p}_i^y, \tilde{p}_i^z) \\ &= \mu_P + C_P^x \hat{p}_i^x + \\ &+ C_P^y f_P(\hat{p}_i^x)\alpha\beta + C_P^y f_S(\hat{p}_i^x)(1-\alpha)\beta \end{aligned} \tag{6.13}$$

where μ_P is the center of mass of the points $\in N_P$. The parameter $\alpha \in [0,1]$ is a weight that balances the influence of information from the path and the surface and $\beta \in [0,1]$ determines the total amount of reconstruction. The procedure is illustrated in Fig. 6.15.

Notice how separate principal component analysis are done on the two data sets. This is motivated by the desire to use only structural information in the surface cloud, without using any absolute location or directional information. This way, the impact of a slight misalignment of the to data sets are reduced. A reconstructed point is however transformed back to the original space of the path using only the principal components of the path, C_P.

By choosing the parameter α close to one, little or no structural information from the surface is used, and all reconstruction is done from information in the path. If α is chosen closer to zero, only information from the surface is used in the reconstruction. The reconstruction procedure is summarized in algorithm 6.5.2.

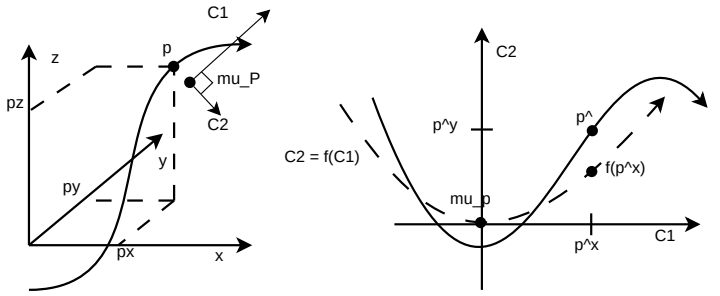


Figure 6.15: Path in Cartesian space 6.15(a) and in principal component space, together with a fitted polynomial 6.15(b). The point $\hat{p}$ will be projected onto the polynomial to form the reconstructed point $\tilde{p} = f(\hat{p}^x)$

---

for all Points p in *segment* **do**
  $C_P, C_S \leftarrow$ Perform PCA on N_P and N_S separately;
  $\hat{N}_P, \hat{N}_S \leftarrow$ Transform N_P, N_S using C_P^{-1}, C_S^{-1};
  Discard third component of transformed points;
  $f_P, f_S \leftarrow$ Fit two, low-order polynomials to two-dimensional points;
  Project $\hat{p}$ onto f_P and f_S;
  Balance influence of projected points using α;
  $\tilde{p} \leftarrow$ Transform projected points back to original path space using C_P;
end for

Algorithm 6.5.2 Reconstruction algorithm summary

To asses the rationality of the proposed reconstruction equation, equation (6.13), consider the case depicted in Fig. 6.16(a). The path depicted exhibits structure not present in the surface. The reconstruction radius is chosen so small that the fitted polynomial (blue) will render the reconstructed point almost equal to the original point. The surface however is of high quality. When the polynomial fitted to the surface (red) is used in the frame of the path's principal components, it is no longer a good fit. When a path point is projected to the surface polynomial it will therefore render a reconstructed point far from the original one, more consistent with the structure in the surface. Projecting the path point onto the surface polynomial in this case will have a smoothing effect, reducing the unwanted structure in the path. For both reconstruction radii shown, the smoothing effect will be greater using the surface polynomial than using the path polynomial.

In Fig. 6.16(b), the surface exhibits structure not present in the path. The center point on the path will be orthogonally projected to a point on the surface polyno-

mial, thus introducing to the path, some of the structure present in the surface. By using a higher degree polynomial, a larger reconstruction radius can be used while maintaining a good fit on features in the surface like the one in Fig. 6.16(b). Using a larger r in this case would shift the center of mass closer to the straight part of the surface, while the higher degree polynomial would be able to closely follow the deep feature. The reconstructed points in this case would closer resemble the structure in the surface. For the application considered however, the case depicted in Fig. 6.16(a), where the path exhibits unwanted structure, is more likely to occur. For this smoothing task, a lower degree polynomial will allow the amount of smoothing to be dependent on the chosen reconstruction radius.

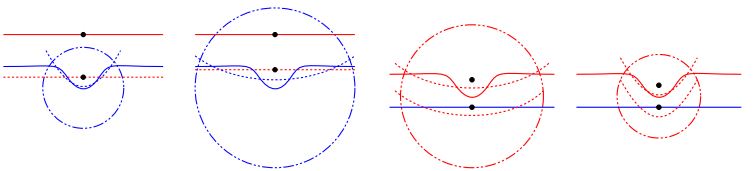


Figure 6.16: Surface (red) and path (blue) together with fitted polynomials (dashed). The center point of the respective data sets within the reconstruction radius (circle) is shown as a black dot. Both cases are depicted with two different reconstruction radii.

Straight lines For a straight line segment, the procedure above simplifies to the fitting of a straight line to the entire segment. This is done in the same way as a higher degree polynomial was fitted in equation 6.10.

Circular segments Paths following a circular arc is common in industrial robotics. An option has been implemented to allow the user to choose whether or not curved segments in the path is to be reconstructed to perfect circular arcs. The reconstruction in this case will use the Taubin method [Taubin 1991] for fitting a circle to the entire curved segment[2]. All points are then projected onto this circle by

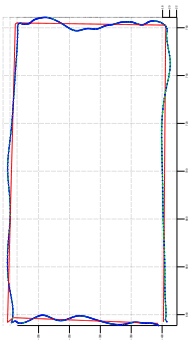
$$\theta = \arctan \frac{\hat{p}^y - \hat{y}_{circle}}{\hat{p}^x - \hat{x}_{circle}}$$
$$X = \begin{pmatrix} \hat{x}_{circle} + r_{circle}\cos(\theta) \\ \hat{y}_{circle} + r_{circle}\sin(\theta) \end{pmatrix}$$
$$\tilde{p} = \mu_P + C_P^{x,y} X;$$

[2]Actual implementation by Nikolai Chernov (13 Jan 2009) `http://www.mathworks.com/matlabcentral/fileexchange/22678`

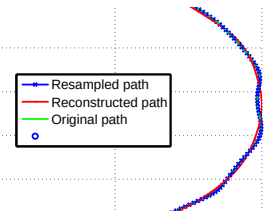
The Taubin method was chosen for its robustness and ability to yield a good fit even when only a small arc of the circle is observed. Experimental results are shown in Fig. 6.17(c).

Experimental results

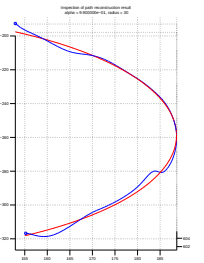
Results for reconstruction of straight segments, curved segments and circular arcs are presented in Fig. 6.17. More results are shown in the article treating the subject submitted in the appendix.



(a) Path was indicated by hand, following the edge of a 250 × 120mm cardboard box. The original path clearly illustrates the difficulty of indicating an edge.



(b) Reconstructed curved path. Path follows a circular edge with about 150mm diameter. No assumption of circular shape was made and the polynomial reconstruction was used.



(c) Reconstructed circular arc. Path follows a circular edge with about 150mm diameter. The path was assumed to consist of a circular arc and circular reconstruction was used.

Figure 6.17: Reconstructed paths (red) together with the original recorded paths (blue). In all cases, the parameter α in equation 6.13 is set to 1, meaning no surface information is used.

6.6 Determining a target frame

A target frame is located in every point on the path. It determines how the tool is to be oriented when that point is passed. In other words, the tool contact point frame, TCP, will coincide with the target frame for every point on the path.

In order to determine the desired orientation of the tool, information about the surface normal in the vicinity of a point is extracted from the point cloud. The surface normal will determine one axis of the target frame, this is however not enough

to fully determine the orientation of the robot end effector. The surface normal determines only two degrees of freedom. The last degree of freedom corresponds to a rotation around the normal.

There are a number of possible ways to choose this last parameter and, depending on the task at hand, two different methods of locking the rotation around the normal is provided.

Non-contact tasks

For a non-contact task, the last degree of freedom is assumed to be arbitrary. This means that it can be chosen in a way that ensures that the pose is reachable based on the current location of the work object and the robot.

For every path segment, the center of mass is calculated μ. From this point, the vector pointing towards the origin of the robot base frame is calculated $R_0 = \mu - p_{robot}$. The normal N together with R_0 determines the second axis as $RxN = R_0 \times N$. This new vector will then together with the normal determine the third and last axis as $R_1 = N \times RxN$. Constructing the target frame is now just a matter of arranging the three vectors as the columns of a 3 by 3 matrix so that each vector corresponds to the desired axis of the TCP frame, while making sure the vectors are normalized, the procedure is described in more detail in section 5.2 on page 35.

Contact task

For a contact task, the orientation around the normal is in general not ambiguous. In this case, the normal N, the path direction D and their cross product $N \times D$ are used to specify the target frame. This will ensure that the same side of the tool is always facing the object. For the target frame to be reachable, $-D$ will sometimes be used instead of D, depending on the work objects location in relation to the robot.

The vectors are treated the same way as for a non-contact task. If the normal N and the direction D are not perfectly perpendicular, the vector D will be fixed. The vector N will then be reconstructed by a double cross product as for the vector R_1 above to ensure that the obtained rotation matrix is orthogonal.

7

Executing the path

The path has now been processed and enhanced using information obtained from a 3D scan of the workpiece. It is now ready for execution. This section describes how the path is transformed to the appropriate coordinate frame and executed. A method will also be developed to allow for compensations of rotations and translations introduced by erroneous calibration and errors in the robot link parameters used in the forward kinematics model.

7.1 Coordinate frame transformation

So far, all processing has been done in the tracker frame. In order to execute the path using the robot, the path must be transformed to the robot frame. Furthermore, the commands given to the robot specifies the position of the tool frame. Thus, all target frames must be transformed to the tool frame. A target frame before transformation specifies the location of TCP1 in the tracker frame, after the transformation, a pose for the tool frame is obtained in the robot base frame. The transformation is given by

$$T_{RB}^{TF} = T_{RB}^{T}\, T_{T}^{Wnew} \left(T_{T}^{Wold}\right)^{-1} T_{T}^{TCP1}\, T_{TCP1}^{TCP0} \left(T_{TF}^{TCP0}\right)^{-1} \tag{7.1}$$

If the work object has been moved between the time of recording and the time of execution, the path must be adjusted accordingly. This is handled by recording and storing the location of the work object when the path is recorded. At the time of execution, a new measurement of the work object location is done, and the path is adjusted by the transformation $T_{T}^{Wnew} \left(T_{T}^{Wold}\right)^{-1}$ in equation 7.1. This allows the user to teach the path in a convenient location and then moving the work object to a pose more suitable for the robot to reach during execution.

7.2 Local improvement of absolute position

As mentioned before, a problem with the transformation matrix T_{RB}^{T} being valid only locally exists due to the errors in the kinematic model used by the robot controller. There are proper ways of dealing with this by calibrating the kinematic model. The calibration procedure is very sophisticated and complicated and usually done by the robot manufacturer. Many SMEs may lack the ability to implement such calibration. A simple way of significantly raise the accuracy of the executed trajectory locally is proposed.

An executed path is, when recorded by the tracker, essentially a series of measurements of the matrix T_T^{TCP1}. When the path is executed, T_T^{TCP1} is the product of

$$T_T^{TCP} = T_T^{RB} T_{RB}^{TF} T_{TF}^{TCP} \tag{7.2}$$

where T_{RB}^{TF} is the desired location as specified by equation 7.1. Due to errors in all three matrices in equation 7.2, the recorded execution will deviate from the desired path.

The matrix T_{TF}^{TCP} contains a translation from the tool frame to the tool contact point, which is typically in the range of 10-40cm. The matrix T_T^{RB} however, contains a translation from the robot base frame to the tracker frame, which can be in the order of several meters. The ratio of the norms of the respective matrices is in our case

$$\frac{\left\| T_T^{RB} \right\|}{\left\| T_{TF}^{TCP} \right\|} \approx 5 \tag{7.3}$$

This fact causes errors in matrix T_T^{RB} to be amplified much more than an error in T_{TF}^{TCP}. Fig. 7.1 shows that the errors in the two matrices are typically the same during late iterations of the calibration algorithm. A method will therefore be developed to find a matrix which when left multiplying the desired location T_{RB}^{TF}, compensates for the error introduced by T_T^{RB}.

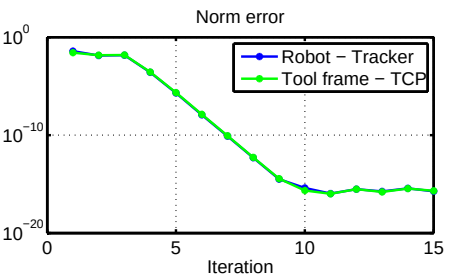


Figure 7.1: Error norm during calibration

Proposed approach

When the path is recorded, processed and everything is set for execution, a test execution is made. During this test run, the path is executed with a small offset, so as to not collide with the work piece. The executed motion is recorded using the tracker and compared to the desired motion. This will yield two paths, which can be aligned using ICP in the general case, and the method using Rodrigues' rotation formula, described in section 6.4, if the paths are single straight lines. The transformation matrix obtained through the alignment can be applied to the path before it is executed in order to compensate for the error that was made during the test run. Equation 7.1 is extended as

$$T_{RB}^{TF} = T_{RB}^{T}\, T_{Calib}\, T_{T}^{Wnew} \left(T_{T}^{Wold}\right)^{-1} T_{T}^{TCP1}\, T_{TCP1}^{TCP0} \left(T_{TF}^{TCP0}\right)^{-1} \tag{7.4}$$

where T_{Calib} is the transformation matrix obtained through alignment of the executed and desired paths.

Fig. 7.2 illustrates an example where the executed path was misaligned with the desired path, both in terms of rotation and translation. The execution of the adjusted path is considerably closer to the desired path compared to the initial execution.

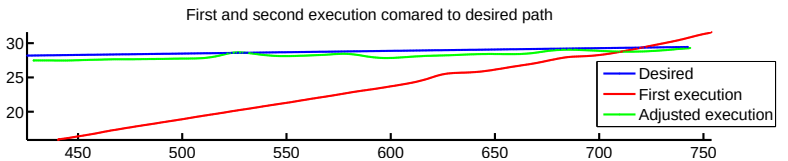


Figure 7.2: Executed path is recorded and compared to the desired path. A transformation matrix is obtained by aligning the two paths. Note the difference in scale between the two axes.

More experimental results are presented in section 8.

8

Results

This chapter will present experimental results in form of a case study. In the end, a brief discussion of error sources is held and a short summary of the computation time associated with the used algorithms is given.

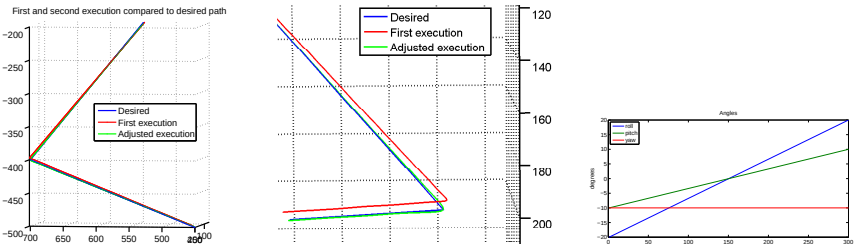
8.1 Accuracy

In order to evaluate the accuracy with which a path can be recorded and executed, the ground truth must be known. In our case, no external measurement system is available and the optical tracking system will be used to evaluate the performance of the entire system. Naturally, using a system to evaluate itself is a flawed approach. The findings presented in this chapter can thus only indicate how well the tracker agrees with the executed trajectories. Results must also be coupled with visual assessments where the performance is compared to the perceived ground truth. Errors presented below will in general represent trajectory tracking errors, unless otherwise noted. The trajectory tracking error is defined as the point wise orthogonal distance from a recorded point to the desired trajectory.

Local calibration adjustment

To verify the effectiveness of the local calibration improvement approach, a test path, depicted in Fig. 8.1, is constructed. The path is synthetically created and contains translations in all three directions and rotations around two axes (Fig. 8.1(c)). The combined length of the two segments measures 590mm. The robot is commanded to execute the path while the location of the tool frame is recorded using the optical tracking system. After an initial execution, the local calibration improvement, described in section 7.2, is applied to enhance the accuracy. A second execution is recorded and both recordings are compared to the desired target path.

Fig. 8.2 shows the distance to the desired path for each recorded point on the executed paths. The distance is calculated as the orthogonal distance to the line between the two closest points on the desired path. The maximum calculated trajec-



(a) Overview (b) Close up view of corner (c) Target orientation along test path

Figure 8.1: Path used for accuracy evaluation

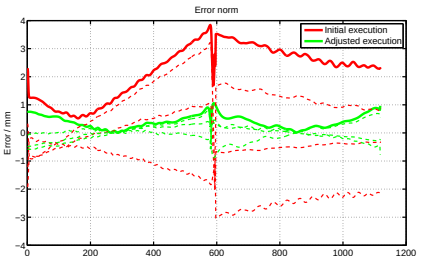


Figure 8.2: Comparison of errors between initial and adjusted execution. Norms are shown with thick solid lines, component wise errors in dashed lines.

tory following error was reduced from 3.8mm to 1.1mm after the adjustment was applied. The mean error decreased from 2.2mm to 0.4mm.

The wavy pattern in the error curves is caused by the specific robot used. Fig. 8.3 illustrates noise present in the measurements of the y-coordinate in the recording of an executed motion. In this case, the robot stops moving after $t = 36$s. The noise is caused by vibrations in the robot, which can be verified by simply listening to and observing the robot while it moves. These vibrations will of course influence the calculated error metric and put a restriction on how accurately a trajectory can be played back.

For the double segment test path, two lines were fitted to the recorded points. The orthogonal distance from each point to the corresponding fitted line was then calculated. The results are depicted in Fig. 8.4 and summarized in table 8.1.

Case study

This section will visualize the entire procedure related to processing and executing of a recorded path. The path will be following the top of the work piece shown in Fig. 8.5, and will thus consist of a circular arc.

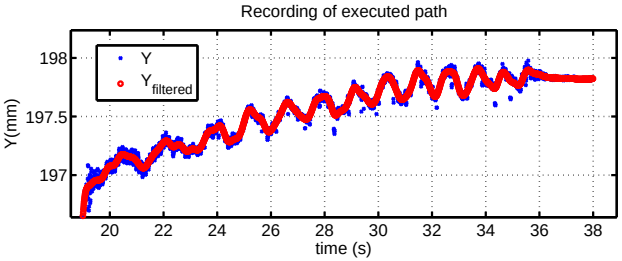


Figure 8.3: Recording of executed motion. Vibrations in the robot are clearly visible in the recorded location.

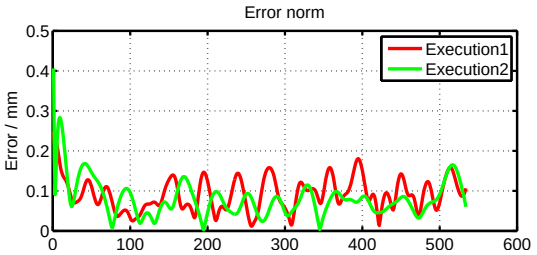


Figure 8.4: Error norms

	Segment 1	Segment 2
mean(e) (mm)	0.09	0.08
max(e) (mm)	0.25	0.41
std(e) (mm)	0.04	0.05

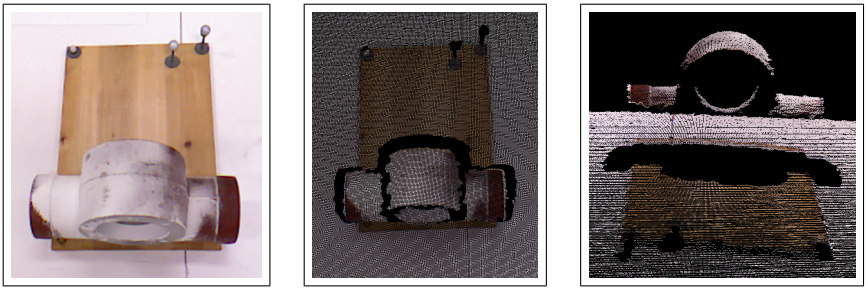
Table 8.1: Errors caused by vibrations

The work piece is painted white in order to improve the scanners ability to capture the surface and it is placed on a mobile platform with markers attached. This will allow the user to move the work piece any time during the process without having to re-teach the path at the new location.

When the path is recorded, a scan will be obtained from the scanner. The path and the scan are aligned and the path will be enhanced with normals estimated from the surface. The result is shown in Fig. 8.6.

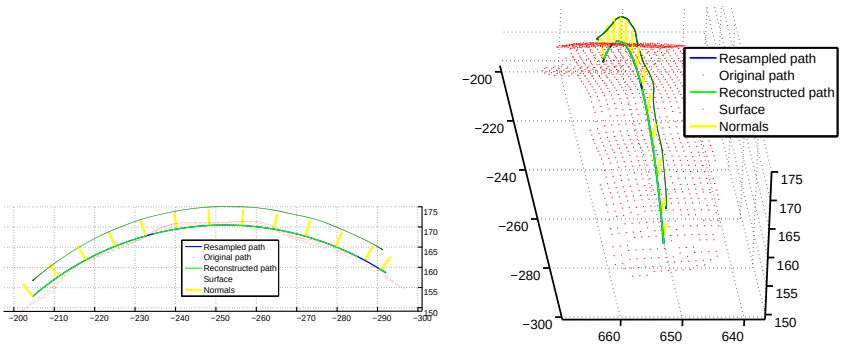
The path will then be reconstructed using the reconstruction method described in section 6.5. The result is shown in Fig. 8.7. The figure clearly illustrates the errors typically introduced by the users hand movements.

When the path is reconstructed and aligned with the surface, target frames will be assigned for each point along the path. The target frames will determine the



(a) 2D view (b) 3D view from scanner (c) 2D view from side

Figure 8.5: Workpiece



(a) (b)

Figure 8.6: Reconstructed path shown on top of surface scan. Estimated normals are shown along the path.

location and orientation of the robot end effector needed to reach the point in the desired configuration. Fig. 8.8 illustrates the orientation in Euler angles obtained from the estimated normals in the surface and directions in the path.

The path is now be ready for execution on the robot. In Fig. 8.9, the path has been executed once and adjusted using the method described in section 7.2. The errors associated with the first and second executions are shown in Fig. 8.10.

The maximum error in this case was reduced from 3.2mm to 1.0mm after application of adjustment. The mean error decreased from 1.4mm to 0.4mm.

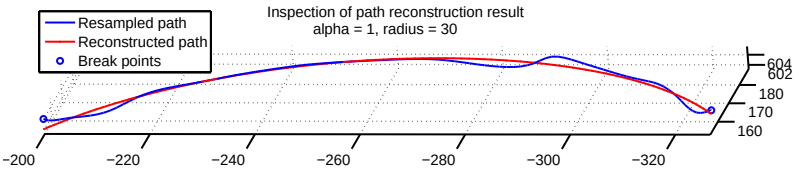


Figure 8.7: Recorded circular arc and reconstruction result

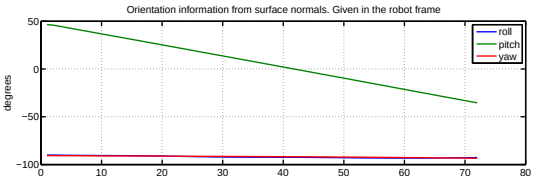


Figure 8.8: Orientations assigned to target frames along the path.

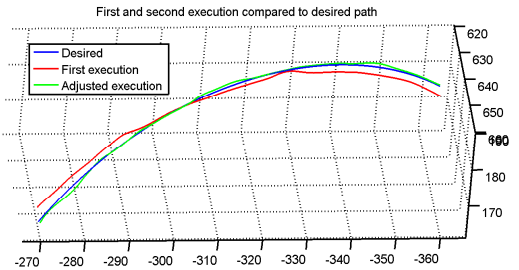


Figure 8.9: Execution of circular arc path

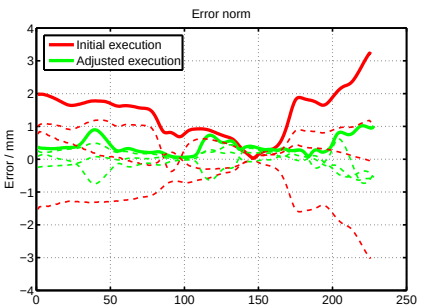


Figure 8.10: Comparison of errors between initial and adjusted execution for circular arc path. Norms are shown with thick solid lines, component wise errors in dashed lines.

Error sources

This section will list the main sources of errors identified, their size and how they affect the result.

Measurement of workpiece location Each time a path is recorded, a measurement of the workpiece location is done. This will allow the operator to move the workpiece at any time without have to re-teach the path. The measurement is however coupled to a potential error which will introduce a rigid translation and rotation to the executed trajectory compared to the desired path. This has been solved by making sure that the optical markers associated with the workpiece are spaced sufficiently far apart and that no occlusion occurs during measurement.

User induced errors during teaching The user induced error will of course be dependent on the individual user. Furthermore, the pose of the user during teaching as well as the nature of the taught path will influence the users ability to indicate a path with high accuracy. This has been dealt with by recording the position of the workpiece at the time of teaching. Any movement of the workpiece subsequent to indication of the path can thus be compensated for, allowing the user to teach the path at a pose he or she finds most convenient. Further work may address this issue by investigating different tool tip designs. A teaching tool with interchangeable tips, suited for indication of different features, may aid the user in the teaching process.

To measure the magnitude of typical user induced errors, simple straight paths have been indicated and the variation around a line fitted to the recorded path has been treated as the error. The maximum magnitude of user induced errors have been found to be in the range of 2-5mm with a standard deviation of 0.5-1mm.

Calibration error Errors in the calibration of the robot and the tracker stems partly from errors in the measurement of the tool frame location during calibration. It is however also tightly coupled to errors in parameters in the robot kinematic chain. For a small work space volume, errors up to 5mm have been identified after calibration. For larger volumes, errors exceeding 20mm has been noticed. Whether these errors stem from the conducted calibration or from the kinematic model of the robot has not been determined.

Errors in robot kinematic chain This error will manifest itself as a nonlinear distortion of the robot base coordinate system. It can therefore not be compensated for by a linear transformation and must be solved by calibrating the internal parameters of the robot. Locally, for a sufficiently short trajectory, the error can be approximated by a linear transformation $\in SE(3)$, which in section 7.2 has been lumped together with the error in T_{RB}^{T}. This composite error has been compensated for by finding a rotation and translation that aligns a recording of an executed trajectory with the desired path. Refer to section 7.2 on page 61 for experimental results.

Orientation errors When a target frame is specified, information about path tangent and estimated normal direction in the surface of the workpiece is used. After

path reconstruction, the path is considered to be the ground truth for the desired path. The two degrees of freedom specified by the path direction in a point will thus be considered error free. The last degree of freedom is specified using the estimated normal direction of the workpiece. This estimate is subject to a wide number of errors:

- Error in the calibration between the scanner and the tool frame.
- Measurement error when scanner location is obtained.
- Scanner lens distortion.
- Scanner noise (described in [Khoshelham 2011]).
- Error in algorithmic alignment of scan and path.

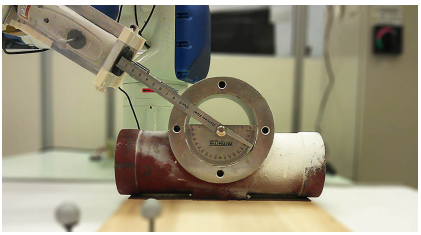


Figure 8.11: Visual assessment of angular error.

The ground truth for orientation depends on the specific workpiece considered and its location. This have not been measurable by any means available, and orientation accuracy has therefor been assessed visually, refer to Fig. 8.11, where the tool is verified to be approximately aligned with the surface normal. The angular accuracy has been determined to be sufficient for the tasks the intuitive programming method is aimed for [Lim and Tao 2010], such as finishing tasks normally carried out by humans.

8.2 Computation time

The computational time associated with execution of the implemented algorithms is naturally dependent on the length of the indicated path. Furthermore, the user may choose the radius used for filtering of the surface scan and the distance from the path within which 3D scan data is kept. Typical times measured on a machine equipped with 3GB RAM and an Intel Core i7 processor are listed below

- Filtering of recorded path: < 2s
- Obtaining 3D-scan and sorting out relevant parts: 12s

- Filtering of surface scan: 2-120s, typically < 10s for a path length of 20cm and 10mm filtering radius.
- Curvature estimation: < 10s
- Fusion of scan and path: < 10s
- Entire procedure: typically 30s

9

Software

In the spirit of the title of this document, extensive work has been done to ensure that the graphical user interface is intuitive and easy to use. Furthermore, a framework independent of the hardware used is developed. This to ensure that a large number of robots, trackers and scanners can be used.

9.1 Software structure

Several peripheral hardware components such as a robot, tracker and scanner, demands customized code in order to connect to and interface with them. Since no standard regarding how this interfacing is done exists, an interface for each class of hardware is written. These interfaces ensures that all components connected to the system follows a known standard. When a new component is to be connected, an implementation of the proper interface is required. The class implementing the interface is compiled to a .dll-file which the user selects when a connection to the component is established. This way, hardware from different vendors can be treated the same way by the main program, and all hardware specific code is written in the interface implementation. The interfaces are written in C♯, which is a managed language. Often programming APIs are written in unmanaged C++, raising the need for a managed wrapper. This wrapper can be written in either C♯, C++ or any other language that supports the Common Language Runtime[1]. As an example, the interface for a robot specifies that all implementations must implement methods for connecting and disconnecting, moving the robot to specified location, reading the current robot location and turn the servo on and off. It is then up to the individual implementations to carry this out in a way suitable for the hardware. When connecting to the robot, the user will select the file which contains the implementation for the robot of choice. Fig. 9.1 illustrates the software structure in form of a class diagram.

[1] `http://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.100).aspx`

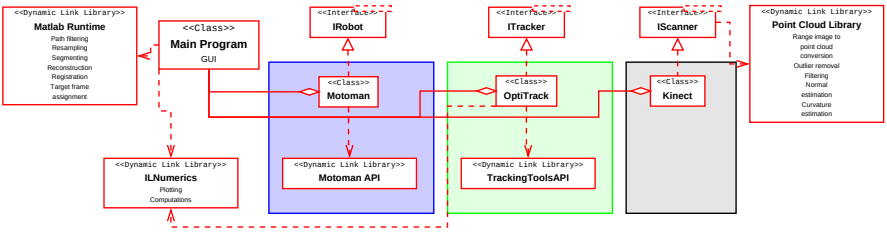


Figure 9.1: Class diagram. The three colored boxes indicate hardware specific components loaded by the Main Program when the user connects to the corresponding hardware.

9.2 Graphical User Interface

The main graphical user interface (GUI) is written in C♯ and uses the Windows Presentation Foundation framework[2]. The main design idea of the GUI is to guide the user through the whole process with a series of steps following each other. In order to minimize visual clutter, only the controls related to the current step is shown. This way, the user will easily find what he is looking for and there will be no confusion as to which functions and controls are available for the current task. Default choices are provided where numerical input is required, and care has been taken in the design of the algorithms so that the effect of changing the parameters are easy to understand.

After processing is done, the user can inspect the path in a 3D plot, together with all tool frame targets. This will aid in the positioning of the work object in relation to the robot. The GUI is depicted in Fig. 9.2.

[2] `http://msdn.microsoft.com/en-us/library/ms754130(v=vs.100).aspx`

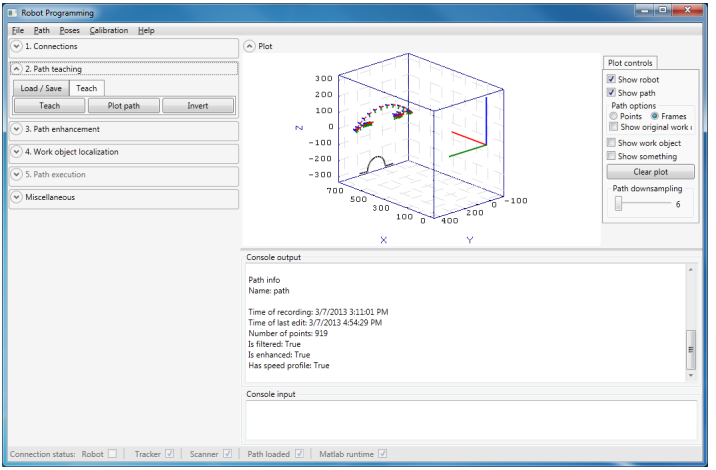


Figure 9.2: Graphical User Interface

10

Discussion and conclusion

With no prior experience in the fields of robotics and computer vision, the first time spent at SIMTech served as an introduction to the theory and techniques that were going to play an important roll during the course of the work. Starting with a fairly blank piece of paper, many ideas have been sketched, discussed, tried out and discarded. This has required an extensive amount of hands-on work and programming, much more so than expected. Especially shortcomings of the optical tracking system has led to long discussions and many experiments to make the teaching phase robust enough to be usable.

Starting with the application of spray painting in mind, early findings indicated that the accuracy needed for contact operation using force control was within reach. With this new goal, the journey towards millimeter accuracy started. It soon became apparent that the most significant error source was not related to the technical equipment. Instead, the user induced error using our current teaching tool proved to be by far the greatest source of error. There are two approaches to reducing this error, post processing of the recorded data and hardware design that better assists the user during the teaching. Designing the teaching tool tip so that the desired feature are easily indicated would eliminate the problem at its source. The possibility to experiment with hardware designs was initially limited, but a 3D printer was ordered, promising fast and easy prototyping. Shipping delays unfortunately led to a very late arrival of the 3D printer, and no such designs were tried out. The second approach however, post processing and reconstruction of the recorded data, proved very successful. Together with interchangeable tool tips, I'm confident that results can be improved upon further.

The use of 3D scanners like the Kinect introduced the need for point cloud processing. PCL provides a vast array of powerful algorithms for this purpose. Being a work in progress however, PCL also comes with a fair bit of bugs that required a lot of time to resolve for a novice C++ programmer like myself. The interaction between unmanaged and managed code needed to interface with various programming APIs also provided many challenges.

Ultimately, the initial goal of demonstrating the usefulness of the intuitive programming method on spray painting was fulfilled. Also the later established goal

of enabling contact operation using force control was fulfilled with a maximum trajectory following error of around 1mm. The intuitive teaching method has a low operator skill requirement and a robot program can be constructed and executed within minutes, making it suitable for HMLV operations.

Future work to further increase the Technology Readiness Level (TRL) of the developed product involves a number of key areas. An update to the 3D scanning hardware would allow greater accuracy in normal and curvature estimations and the proposed curve reconstruction method would allow better reconstruction of the path using a higher quality scan. Interchangeable tool tips for the teaching tool would assist the user in the indication of challenging features such as edges. Better tool tips together with a high accuracy mode, where the user indicates single points without moving the teaching tool, would further enhance the accuracy of the taught path.

A

Teaching tool ergonomics

A.1 Effects of handle angle and work orientation on hammering: Wrist motion and hammering performance.

In case of hammering, bending the tool and not the wrist caused less stress on muscles [Schoenmarklin and Marras 1989]. No significant effect on performance was found. However, hammering orientation had significant effect on performance. Wall hammering performance was significantly lower than bench hammering performance.

A.2 Identifying factors of comfort in using hand tools

See table A.1 on the facing page. [Kuijt-Evers *et al.* 2004]

A.3 Optimal handle angle of the fencing foil for improved performance.

Improperly designed hand tools and sports equipment contribute to undesired injuries and accidents [Lin 2004]. The idea of bending the tool, not the wrist, has been applied to sports equipment. According to Bennett's idea, the design of an ideal handle angle should be in the range of 14 degrees to 24 degrees. Thus design of the handle angle in the sport of fencing is also important. A well-designed handle angle could not only reduce ulnar deviation to avoid wrist injury but also enhance performance. An experiment with several different handle angles was conducted to analyze the effect on performance. Analysis showed an angle of 18 degrees to 21 degrees provided best overall performance in fencing.

	Descriptor	Mean ranks		Descriptor	Mean ranks
1	Reliable	14.68	21	No irritation	19.20
2	Functional	15.45	22	Handle shape	19.27
3	Good fit in hand	15.67	23	Sharpness	19.27
4	Easy in use	15.99	24	Pleasurable	19.63
5	Force exerted from tool	16.10	25	No inflamed skin	19.69
6	No blisters	16.33	26	No slippery handle	19.81
7	Safe	16.39	27	Relaxed working posture	20.48
8	No pain	16.51	28	No sore muscles	20.50
9	Handle feels comfortable	16.59	29	Weight of tool	22.32
10	No peak pressures on hand	16.65	30	Handle size	22.50
11	High quality tool	16.82	31	No sweaty hands	22.53
12	High product quality	17.20	32	Easy to take along	22.73
13	Task performance	17.20	33	No pressure on hand	22.76
14	No body part discomfort	17.44	34	Roughness of handle surface	22.85
15	Lack of tactile feeling	17.70			
16	Friction between hand and handle	18.16	35	Handle does not feel clammy	23.30
17	No muscle cramp	18.40	36	Handle hardness	23.66
18	Low handgrip force supply	18.43	37	Solid design	32.56
19	No numbness in fingers	18.73	38	Professional looks	33.97
20	Comfortable working posture	18.80	39	Styling	36.03
			40	Nice color	37.69

Table A.1: Ranking of descriptors based on mean ranks. [Kuijt-Evers *et al.* 2004]

B

Laser Scanner

In an early stage of the project, an alternative 3D scanning technology was evaluated. A laser scanner, or laser stripe profiler, works by projecting a laser line onto the scene. The plane of the laser light is known in the coordinate system of a camera observing the scene, allowing 3D reconstruction of points along the line using triangulation. The section will describe such system, and discuss some alternative ways of incorporating it into the system developed in previous sections.

B.1 Theory, Depth Calculation by Triangulation.

If the laser light plane in the camera frame is known, the depth of any point on the intersection between the plane and the scanned surface can be calculated as follows. The point lies both in the laser light plane and on the line between the pixel capturing the light and the center of the camera lens. The coordinates for the point in the camera frame can be calculated by calculating the intersection of the laser plane and the line.

$$\Pi_l : (\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{n} = 0 \quad \text{The laser plane equation} \tag{B.1}$$

$$\lambda : \mathbf{p} = d\mathbf{l} + \mathbf{l}_0 \quad \text{The line equation} \tag{B.2}$$

$$0 = (d\mathbf{l} + \mathbf{l}_0 - \mathbf{p}_0)\mathbf{n} \quad \text{Substitution of (B.2) into (B.1)}$$

$$0 = d\mathbf{l}\mathbf{n} + (\mathbf{l}_0 - \mathbf{p}_0)\mathbf{n}$$

$$d = \frac{(\mathbf{p}_0 - \mathbf{l}_0)\mathbf{n}}{\mathbf{l}\mathbf{n}} \quad \text{Solving for } d \tag{B.3}$$

A vector in the line direction is given by

$$\mathbf{l} = \frac{(-x_c, -y_c, f)}{\|(-x_c, -y_c, f)\|}$$

which is the line between the pixel and the lens center. A point on the line is given by the coordinates of the pixel on the sensor $\mathbf{l}_0 = (x_c, y_c, 0)$. A point on the plane $\mathbf{p}_0$ can be chosen as any point obtained through the calibration of the laser light plane. $\mathbf{n}$ is the laser plane normal. With these parameters, d can be calculated from equation B.3 and the distance from the pixel to the lens center from

$$d_f = \sqrt{x_c^2 + y_c^2 + f^2}$$

which in turn yields the real world coordinates

$$z = \frac{d}{d_f} \cdot f \qquad x = \frac{(z-f)x_c}{f} \qquad y = \frac{(z-f)y_c}{f}$$

This follows from the geometry in Fig. B.1.

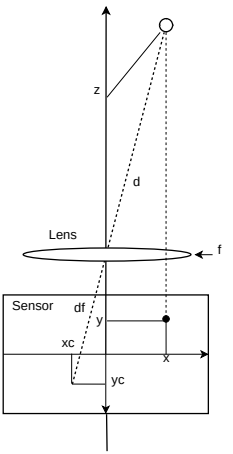


Figure B.1: Coordinate conversion.

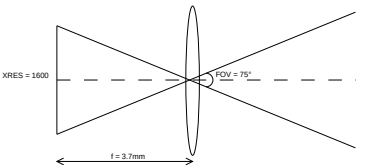


Figure B.2: Conversion between pixel coordinates and meters.

The algorithm relies on a number of camera specific parameters. The parameters for the Logitech Quickcam Pro 9000 is given in table B.1 on the next page. To convert from pixel coordinates to real world camera coordinates, the following formula is derived from the geometry in Fig. B.2

$$\frac{\frac{XRES}{2}}{f} = \tan\left(\frac{FOV}{2}\right)$$

The conversion between pixel coordinates, x_p, and camera frame coordinates, x_c are then given by

$$x_c = x_p \frac{2f\tan\left(\frac{FOV}{2}\right)}{XRES}$$

During a scan, the position of the robot end effector is known. To obtain the equation for the laser light plane in the camera frame a calibration must be done. This is discussed further in section B.3.

Horizontal field of view (FOV)	75°
Horizontal resolution ($XRES$)	1600px
Focal length (f)	3.7mm

Table B.1: Camera parameters for the Logitech Quickcam Pro 9000

B.2 Commercial solutions

Several commercial solutions for conducting 3D laser scans exists. One low-cost alternative is the David laser scanner. David Laser scanner starter kit[1] is considered to conduct a 3D surface scan along the indicated path. The provided software is relying on a camera calibration corner for the scan to work. The calibration corner consists of two flat surfaces mounted perpendicular to each other to form the corner in which the item to be scanned is placed. During the scan, the laser line must be visible on the calibration corner on both sides of the object at all times. This is how the software calculates the laser light plane. However, if the laser light plane is known, this calibration corner is not needed. David laser scanner software has limited support for scanning without the calibration corner where a pre-calibration is done before the real scan. This imposes a number of constraints on the scan procedure. The laser movement must be identical in path and velocity during the calibration run and the scan and the calibration corner must be present during the calibration so that the laser light plane at all time instants can be calculated and remembered. There is no existing support for predetermined laser light planes.

B.3 Calibrating the laser plane parameters

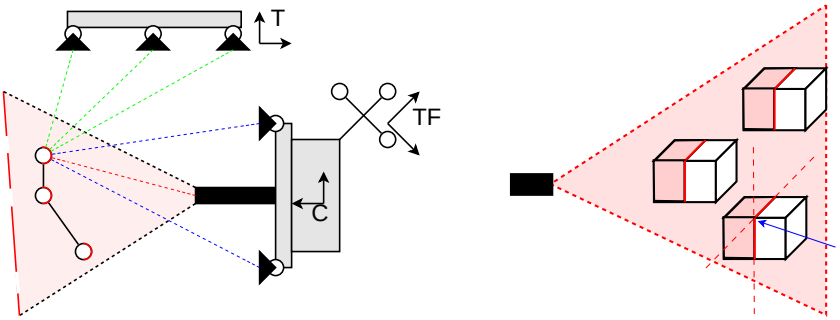
For a laser scanning system to be useful, the plane equation of the laser light must be known with respect to the camera frame. The following proposed approaches both use a laser scanner with a stereo camera system to determine the equation of the laser plane.

Marker approach

Placing three or more markers in the laser plane will allow for a direct measurement of this plane, see Fig. B.3(a) on the facing page. The markers can also be used to relate the coordinate systems of the laser scanner and the tracker.

This approach requires careful manual alignment of the markers. An error analysis is presented in section B.3 on the next page

[1] `http://shop.david-vision-systems.de/product_info.php/language/en/info/p84_DAVID-Starter-Kit.html`



(a) Marker calibration approach. Laser scanner using a stereo camera system and a calibration object. The calibration object allows the laser plane to be captured by the laser scanner camera system and the tracker at the same time.

(b) Line intersection calibration approach. The intersection points of the laser lines (indicated by the arrow) can be calculated and from correspondence between the two images, the depth can be calculated.

Figure B.3: Laser scanner setup.

Line intersection approach

When shining the laser stripe on an object with several distinct edges, a number of lines will be visible in the image captured by the laser scanner's stereo camera system, see Fig. B.3(b). These lines together with their intersection points are all lying in the laser light plane. Using Hough transform, parameters of these lines can be estimated. By matching the intersection points of these lines between the two images, the depth of the intersection points can be calculated. With three or more points which are not co-linear, the laser light plane can be calculated. The calculated intersection points can be obtained with higher accuracy than a point shaped object like a marker.

This approach requires robust line fitting as well as a suitable calibration object featuring two planar surfaces intersecting at an appropriate angle.

Error analysis of marker approach

Consider an independent alignment error with 0.5mm standard deviation, $dy \in \mathcal{N}(0, 0.5mm)$ along the y-axis for all markers as seen in Fig. B.4. The magnitude of the error is dependent on the spacing between the top and bottom markers (p_2, p_3), and the marker in the origin (p_1). Results are given in the form of Fig. B.5(a) on the following page.

Error analysis of Line intersection approach

While shining the line laser on an edge as depicted in Fig. B.3(b) and B.6(a), several frames of the scene were recorded by the camera. Each frame was processed with

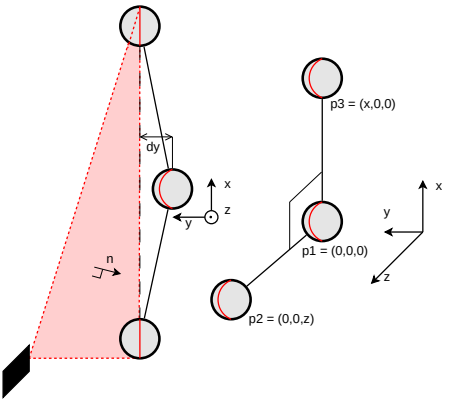
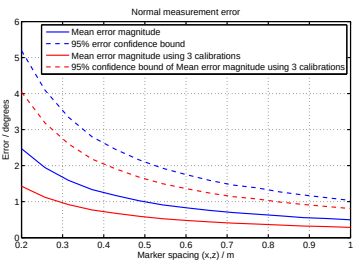
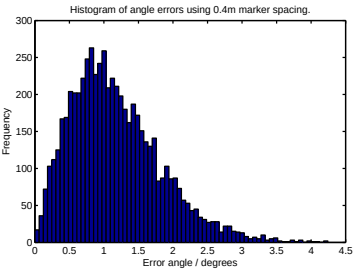


Figure B.4: Calibration object. A manually aligned calibration object will have some alignment error (dy).



(a) Normal error curve assuming an independent alignment error $dy \in \mathcal{N}(0, 0.5mm)$ for all markers in Fig. B.4.



(b) Histogram of magnitude of angle errors using 0.4m marker spacing.

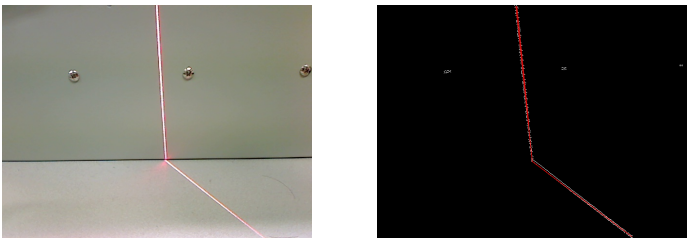
Figure B.5: Angle error analysis.

the following algorithms, in order

1. Thresholding
2. Canny edge detection[2]
3. Probabilistic Hough Line Transform[3]

[2] `http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html#canny-detector`

[3] `http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html#hough-lines`



(a) Original image.

(b) Detected edges (white) and lines (red).

Figure B.6: Example frame from analysis of Line intersection approach.

The detected lines were exported to Matlab and intersection points between all lines in a single frame was calculated. This was done for all frames. The resulting intersection points, after outlier removal is shown in form of a histogram, see Fig. B.7.

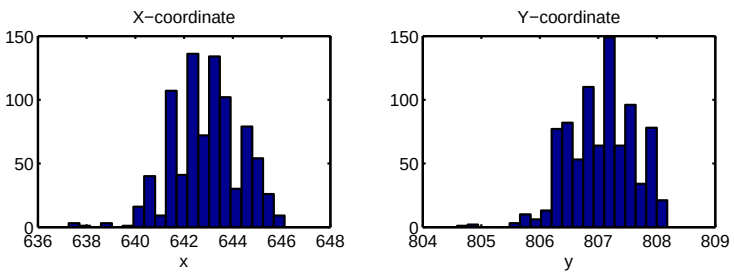


Figure B.7: Distribution of calculated pixel coordinates of intersection points. The standard pixel deviation of the estimated intersection points are $S_x = 1.40$px, $S_y = 0.57$px. A sample size of $n = 200$ frames of resolution 1600×1200px from each camera was used for the analysis. Assuming normally distributed errors, the standard deviation of the mean coordinate estimate is $S/\sqrt{n} \Rightarrow S(m_x) = 0.10$px, $S(m_y) = 0.04$px

By the nature of the Canny edge detection algorithm, two lines located nearby each other is detected. For better results, more robust line detection is necessary. However, the standard deviation of the mean coordinates is inversely proportional to the square root of the number of frames. Since the laser is mounted on the robot, obtaining a large number of frames and thereby reducing the variance of the estimate is possible.

Improved line detection, more samples, higher resolution cameras and use of only top voted lines are examples of measures that can be taken in order to improve the calibration result.

B.4 Comparison between laser scanner and Kinect

The Kinect has a minimum working distance of about 50cm. For small objects, this imposes a serious limitation on the maximum resolution obtainable. The laser scanner can be placed as close as the camera can focus (closer than 25cm). A simple experiment showed that the point density obtained when a small (10cm) object was scanned differed by at least one order of magnitude in favor of the laser scanner.

No comparison regarding accuracy has been made.

Polynomial Reconstruction of 3D sampled Curves Using Auxiliary Surface Data

Fredrik Bagge Carlson*
Lund University
Faculty of Engineering
PO Box 118
SE22100 Lund Sweden
ael07fba@student.lu.se

Ngoc Dung Vuong
Singapore Institute of
Manufacturing Technology
Mechatronics Group
71 Nanyang Drive, Singapore
ndvuong@SIMTech.a-star.edu.sg

Rolf Johansson
Lund University
Dept Automatic Control
PO Box 118
SE22100 Lund Sweden
Rolf.Johansson@control.lth.se

***Abstract*— This paper proposes a method for structural enhancement of a 3D sampled curve. The curve is assumed to be organized, but corrupted with low frequency noise. The proposed method approaches the notion of curve reconstruction in a novel way, where information about the structure in a scanned surface is used to reconstruct the curve. Principal Component Analysis is carried out on successive neighborhoods along the curve to estimate reduced dimensionality spaces, which allows polynomial reconstruction. The effectiveness of the proposed method is verified by both simulations and experiments.**

***Index Terms*— Polynomial reconstruction, 3D sampled curve, point cloud**

I. INTRODUCTION

Industrial robots have traditionally been used for pick and place tasks, where absolute accuracy performance is not crucial. In recent years however, industrial manipulators have been introduced to contact tasks and machining, where they can offer a more flexible, lower cost alternative to CNC machines [1] [2]. Traditionally, programming of industrial robots have been done using the conventional *online* programming approaches such as *Lead through* or *Walk through* [3]. This method requires an operator to manually control the robot to a number of poses along the desired trajectory, which will then be remembered and repeated. In the context of Small and Medium-sized Enterprises (SMEs), high-mix low-volume operations where the robot program is executed a low number of times, is common. The workload related to online programming will, in such operations, amount to a significant part of the total cost related to operating the industrial robot. This often makes online programming economically infeasible, thus raising the need for new, rapid, programming methods.

Offline programming is an alternative programming method that involves planning of a robot trajectory using a CAD model of the work piece. The work piece CAD model however may not always be accessible in practice, especially if the work piece has been used or modified since construction. To enable offline programming, the operator may resort to reverse engineering of the work piece which is both time consuming and costly. Nevertheless, the field of reverse engineering of industrial work pieces has seen increasing popularity along with the introduction of the offline programming methodology. The availability of high performance 3D scanners from companies such as GOM[1] and Leica[2] has further increased the interest in techniques for CAD model construction based on point cloud data. This has for example been considered in [4], where the authors used several range images obtained from different viewpoints to reconstruct a complete model of a work piece.

An important part of a workpiece model is natural features, such as edges. Several authors have considered reconstruction of point sampled surfaces using feature reconstruction. In [5], the authors reconstructed features from stereo photos and in [6], the author used a Bayesian approach to reconstruct curves from a single photo, utilizing prior model knowledge. In [7] [8], the authors reconstructed curves based on a surface point clouds alone. In [9], the author used a moving least-squares technique to obtain a thinner version of an unorganized point cloud representing a curve sampled under heavy noise and in [10], knowledge of a kinematic process used to generate a surface were used to reconstruct the surface itself and features within it.

As can be seen, the market is still open for a rapid robot programming method that requires no prior knowledge of the work piece geometry and does not rely on perfect work piece reconstruction, which is based on costly and time consuming reverse engineering. In this work, we introduce the need for reconstruction of a sampled curve in an application of intuitive robot programming, aimed at increasing the productivity of SMEs. In the application considered, a user indicates a robot path using a device, which location in space can be accurately tracked. The user indicated path is assumed to be following a surface with unknown structure and is prone to errors due to hand shake and mistakes made by the user, reducing the accuracy and raising the need for

*Work supported by the project C12-R-006 at Singapore Institute of Manufacturing Technology

[1] http://www.gom.com/metrology-systems/3d-scanner.html
[2] http://hds.leica-geosystems.com/en/Leica-ScanStation-C10_79411.htm

curve reconstruction.

After recording, the user indicated path is augmented using information from a 3D scan of the workpiece. The point cloud representing the surface of the work piece will contain structural information, which can be exploited in order to reconstruct the recorded path and correct errors caused by the user.

The area of curve reconstruction is a well explored field also outside the field of work piece reconstruction. Savitzky and Golay pioneered the field of polynomial reconstruction of curves [11]. Their method fits a polynomial to points within some distance from a center point. The center point is then projected onto the fitted function. This approach applies to two-dimensional data and does unfortunately not extend directly to higher dimensions.

To proceed to three dimensions, a straightforward approach is to project the sampled points onto an appropriate two-dimensional subspace, where an analytical function can be fitted. This has been explored in [12], where the authors use Principal Component Analysis (PCA) to locally estimate the tangent vector of a point sampled curve. A piece wise linear curve is then obtained using the estimated tangents. The data sets under consideration were unorganized noisy point clouds and no information regarding the sequence in which the points occur was available.

As can be seen, the above mentioned works concentrate either on reconstructing curves from unorganized point clouds, or on reconstructing curves that are believed to be present in a point cloud, such as an edge. Reconstruction of a curve related to a point cloud, but not belonging to it, have not been treated. To deal with this problem, a new method is proposed. The proposed approach considers the surface point cloud as an auxiliary data set, used as a mean to enhance the structure of a curve which is sharing features with it, in a way that may not have been considered before. A novel approach is therefore investigated, where no CAD model reconstruction is done and only relevant parts of the the surface, from a small neighborhood around the curve, is used. This results in an algorithm suitable for reconstruction of noisy curves indicated by hand, aimed at execution by industrial robots for manufacturing and finishing tasks.

The curve and the surface point cloud are dissimilar in nature, but related to each other. In the considered application, no restriction is put on what kind of surface feature the curve is following. A reconstruction algorithm must thus be able to handle a large variety of cases. For high flexibility, the 3D scanner is mobile, which introduces uncertainty in the calibration between the tracking system and the scanner. This fact increases the difficulty of the exploitation of the auxiliary data.

The paper is organized as follows. Initially, a brief review of theory related to the proposed solution is presented in Sec. II, followed by the proposed approach in Sec. III. Simulations and experimental results are presented in Sec. IV and V, followed by an ending discussion.

II. Principal Component Analysis - A Review

Principal Component Analysis (PCA), originally introduced by Karl Pearson [13], is often used as a way to reduce the dimensionality of high dimensional data. A dimensionality reduction may enable easier visualization of the data or a lower complexity representation. It will be used here as a way to allow analytical functions on the form

$$f : \mathbb{R} \to \mathbb{R} \tag{1}$$

to be fitted to the originally three-dimensional data.

To allow dimensionality reduction, PCA finds a vector $\in \mathbb{R}^n$, along which the data exhibit the greatest amount of variance. This vector will be called the first principal component of the data set. The second principal component will be the direction which describes as much as possible of the variance that is not described by the first principal component, under the restriction that the two components are orthogonal. Refer to Fig. 1 for an illustration of the concept. The figure illustrates how the small third principal component can be neglected in order to obtain a lower-dimensional representation of the data.

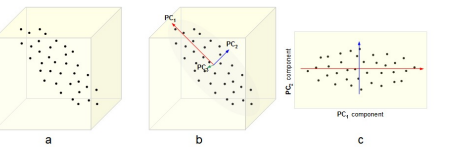


Fig. 1. Illustration of dimensionality reduction using PCA [14]

Formally, the PCA finds the eigenvalue, eigenvector pairs (σ, V) of the covariance Σ matrix of the data such that

$$\Sigma V = \sigma V \tag{2}$$

The eigenvector V_X corresponding to the greatest eigenvalue σ_x will be the first principal component.

III. PROPOSED APPROACH

As described in Sec. I, an application of intuitive robot programming raises the need for structural enhancement of a three-dimensional, point sampled curve. In this section, a method for polynomial reconstruction of a curve using a related point sampled surface is developed. The curve is assumed to be organized, corrupted with low frequency noise and following the surface of a workpiece with unknown geometry.

After pre-processing, outlined in algorithm 1, the path is assumed to consist of a single, smooth segment with equidistant points. Reconstruction now starts by a choice

of a parameter r called the reconstruction radius. For each point p_i on the path, neighboring points within distance r will be used used in the reconstruction of p_i. The set of points within the reconstruction radius will be termed the path reconstruction neighborhood N_P. Each point will also have a set of nearest neighbors in the surface. If a large fraction of the nearest neighboring surface points to the current path segment is considered to be edge points (points with high curvature), the path segment is considered to be following an edge. Nearby edge points will then be chosen as the surface reconstruction neighborhood N_S. If no edge is present in the surface, N_S will consist of the k nearest neighbors in the surface to all points $\in N_P$.

Algorithm 1 Pre-processing outline

```
Re-sample path to ensure points are equidistant;
Divide path into segments if sharp features are present;
Classify segments as straight lines, circles or curved paths;
for all Segments do
    if segment is straight line or circle then
        continue; ▷ These cases are handled separately.
    end if
    Determine if segment follows edge based on surface curvature;
    for all points p in segment do
        Establish N_P and N_S;
    end for
end for
```

Using the established N_P and N_S, the reconstruction will proceed as follows. A PCA will be carried out on all points in N_P and N_S respectively. The analysis will yield two orthonormal matrices C_P and $C_E \in \mathbb{R}^{3\times3}$, whose columns correspond to the principal components of the respective data set. The PCA will through the matrices C_P and C_E transform the data points p_i to an orthogonal principal component space $\hat{p}_i$ as

$$\hat{p}_i = C^{-1}(p_1 - \mu) \tag{3}$$

where μ is the center of mass for the data set currently analyzed.

The transformed points are then orthogonally projected to the subspace spanned by the first two principal components by setting the third coordinate to zero. Under the assumption that the data is planar, the third principal component holds no relevant information and only noise is lost in the projection.

A polynomial of low degree is fitted to the projected points. For a single data set, a reconstructed point $\tilde{p}_i$ is formed by the projection of $\hat{p}_i$ onto a fitted polynomial as

$$\begin{aligned} f &= \text{Polynomial fitted to points } \in N \\ \tilde{p}_i &= (\tilde{p}_i^x, \tilde{p}_i^y, \tilde{p}_i^z) \\ &= \mu + C^x\hat{p}_i^x + C^y f(\hat{p}_i^x) \end{aligned} \tag{4}$$

where the super-script $p^x \in \mathbb{R}^1$ denotes the first component of $p = (p^x, p^y, p^z)^\mathsf{T}$ and $C^x \in \mathbb{R}^{3\times1}$ denotes the first column of $C = (C^x\ C^y\ C^z)$. The operation

$$p_i = C\hat{p}_i + \mu \tag{5}$$

transforms a point in principal component space back to Cartesian space. The term

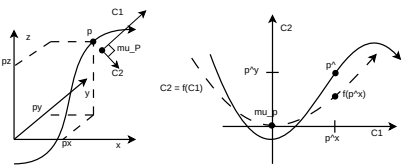
$$C^y f(\hat{p}_i^x) \tag{6}$$

forms the second component of the reconstructed point by evaluating the polynomial and transforming the result back to the original space.

To allow the use of both information from the path and the surface in the reconstruction, eq (4) is extended to

$$\begin{aligned} \tilde{p}_i &= (\tilde{p}_i^x, \tilde{p}_i^y, \tilde{p}_i^z) \\ &= \mu_P + C_P^x\hat{p}_i^x + \\ &+ C_P^y f_P(\hat{p}_i^x)\alpha\beta + C_P^y f_S(\hat{p}_i^x)(1-\alpha)\beta \end{aligned} \tag{7}$$

where μ_P is the center of mass of the points $\in N_P$. The parameter $\alpha \in [0, 1]$ is a weight that balances the influence of information from the path and the surface and $\beta \in [0, 1]$ determines the total amount of reconstruction. The procedure is illustrated in Fig. 2.



(a) (b)

Fig. 2. Path in Cartesian space 2(a) and in principal component space, together with a fitted polynomial 2(b). The point $\hat{p}$ will be projected onto the polynomial to form the reconstructed point $\tilde{p} = f(\hat{p}^x)$

Notice how separate principal component analysis are done on the two data sets. This is motivated by the desire to use only structural information in the surface cloud, without using any absolute location or directional information. This way, the impact of a slight misalignment of the to data sets are reduced. A reconstructed point is however transformed back to the original space of the path using only the principal components of the path C_P.

By choosing the parameter α close to one, little or no structural information from the surface is used, and all reconstruction is done from information in the path. If α is chosen closer to zero, only information from the surface is used in the reconstruction. The reconstruction procedure is summarized in Alg. 2.

To asses the rationality of the proposed reconstruction equation, Eq. (7), consider the case depicted in Fig. 3(a). The path depicted exhibits structure not present in the

Algorithm 2 Reconstruction algorithm summary

for all Points p in *segment* **do**
  $C_P, C_S \leftarrow$ Perform PCA on N_P and N_S separately;
  $\hat{N}_P$, $\hat{N}_S \leftarrow$ Transform N_P, N_S using C_P^{-1}, C_S^{-1};
  Discard third component of transformed points;
  $f_P, f_S \leftarrow$ Fit two, low-order polynomials to two-dimensional points;
  Project $\hat{p}$ onto f_P and f_S;
  Balance influence of projected points using α;
  $\tilde{p} \leftarrow$ Transform projected points back to original path space using C_P;
end for

surface. The reconstruction radius is chosen so small that the fitted polynomial (blue) will render the reconstructed point almost equal to the original point. The surface however is of high quality. When the polynomial fitted to the surface (red) is used in the frame of the path's principal components, it is no longer a good fit. When a path point is projected to the surface polynomial it will therefore render a reconstructed point far from the original one, more consistent with the structure in the surface. Projecting the path point onto the surface polynomial in this case will have a smoothing effect, reducing the unwanted structure in the path. For both reconstruction radii shown, the smoothing effect will be greater using the surface polynomial than using the path polynomial.

In Fig. 3(b), the surface exhibits structure not present in the path. The center point on the path will be orthogonally projected to a point on the surface polynomial, thus introducing to the path, some of the structure present in the surface. By using a higher degree polynomial, a larger reconstruction radius can be used while maintaining a good fit on features in the surface like the one in Fig. 3(b). Using a larger r in this case would shift the center of mass closer to the straight part of the surface, while the higher degree polynomial would be able to closely follow the deep feature. The reconstructed points in this case would closer resemble the structure in the surface. For the application considered however, the case depicted in Fig. 3(a), where the path exhibits unwanted structure, is more likely to occur. For this smoothing task, a lower degree polynomial will allow the amount of smoothing to be dependent on the chosen reconstruction radius.

IV. Simulation

In this section, the functionality of the proposed algorithm is verified using artificial data sets. The synthetic data were corrupted with noise to simulate the output from real systems, such as low frequency noise induced by the users hand motions and noise in the 3D scanner generating the surface point cloud.

Initially, the intended usage of the algorithm is illustrated. Fig. 4 shows a simulated point cloud surface (red)

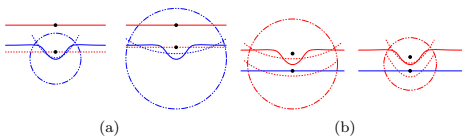


Fig. 3. Surface (red) and path (blue) together with fitted polynomials (dashed). The center point of the respective data sets within the reconstruction radius (circle) is shown as a black dot. Both cases are depicted with two different reconstruction radii.

with a noisy path (blue) following an edge in the surface. The simulation compares the output of the algorithm using both pure polynomial smoothing of the path, ($\alpha = 1$), and reconstruction using edge information ($\alpha = 0$). Fig. 4(a) indicates that involving edge information yields a better result compared to pure polynomial smoothing of the path. This verifies that reconstruction using structural information from the auxiliary data increases the performance of the algorithm, provided that the surface cloud used is of high quality (low noise content). For this experiment, colored noise was added to the path before reconstruction to allow comparison between the reconstruction result and a noise free path. The noisy path was aligned to the edge points of the surface point cloud using the Iterative Closest Point algorithm. This resembles a real scenario where alignment between the two data sets may be inaccurate. The surface cloud was synthetically constructed and corrupted with noise with standard deviation of 2% of the model scale. Moving Least-Squares filtering and curvature estimation was done using Point Cloud Library[15].

A simulated path allows a point wise error to be calculated between the reconstructed path and the original path before it was corrupted by noise. Fig. 5 shows a residual plot for an experiment using the same configuration as when the result in Fig. 4(b) was produced. The errors produced by the formula

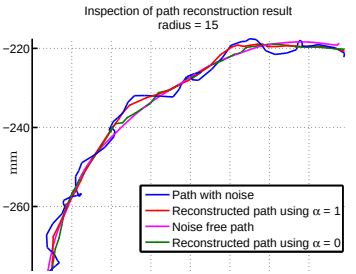
$$e = \sum_{i=1}^{N} \left\| p_i^{desired} - p_i \right\| \tag{8}$$

was $e_0 = 13.0$ for the reconstructed path using $\alpha = 0$, $e_1 = 16.8$ for the reconstructed path using $\alpha = 1$, and $e_n = 21.9$ before reconstruction. The absolute magnitude of these numbers are of secondary interest, the ratio however indicates that the average error decreased after reconstruction.

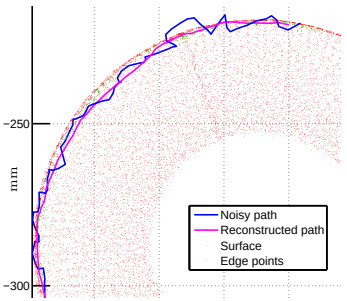
V. Experimental results

To verify the functionality of the proposed approach on real world data, an experimental result using a surface point cloud obtained from the Kinect[3] sensor is shown in Fig. 6. The Kinect sensor is a low cost 3D scanner which is used to prove the concept of the proposed intuitive robot programming approach. The Kinect scanner suffers

[3] `www.xbox.com/kinect`



(a) Results are shown using $\alpha = 1$ and $\alpha = 0$.



(b) The result using $\alpha = 0$ shown together with the point cloud used for reconstruction.

Fig. 4. Reconstructed curved path.

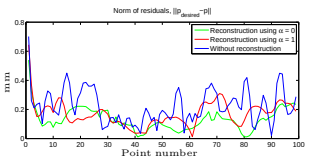


Fig. 5. Norm of residuals for the experiment shown in figure 4(b)

from a correlated noise, described in [16]. When used at it's minimum working distance, this noise has approximately the same magnitude as the typical errors in the path caused by the user, with a standard deviation of approximately 1mm. The simulated results shown earlier indicates that the performance of the algorithm will improve with better quality surface data. Still, Fig. 6 shows that the data from this very low cost scanner can be used for correction of errors in the user indicated path.

The path recorded for these experiments is recorded using the OptiTrack Flex 13 optical tracking system[4]. The recorded path shows typical errors related to indicating a path by hand. Experience has shown that indication of an edge by hand typically introduces a maximum error of 2-5mm and a standard deviation of 0.5-1mm.

The path shown is supposed to follow a flat surface. In the center of the segment, the user has made a significant error, causing the path to rise above the surface. If the surface is used in the reconstruction ($\alpha = 0$), the reconstructed path will be following the surface more closely, indicated by the green path. This result indicates that *low cost* equipment such as the Kinect is indeed useful for reconstruction for manufacturing purposes. An application where the curve under consideration have errors of greater magnitude than the ones presented here, may thus benefit greatly from structural enhancement using such low cost equipment.

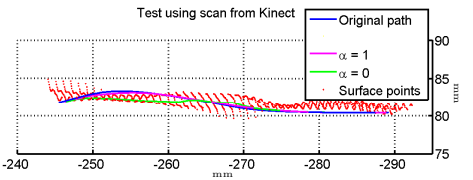


Fig. 6. Test results using a scan from the Kinect sensor and path recorded using OptiTrack Flex 13 optical tracking system.

If no surface information is available, the proposed method can be used using only information from the path. Fig. 7 illustrates a result where a curved path is reconstructed without the use of surface information. If no auxiliary data is available, a large reconstruction radius may be used to enhance the structure of the path using pure polynomial smoothing, adapted to three dimensions. This indicates that the proposed method is useful even for cases where the auxiliary data is absent or of poor quality.

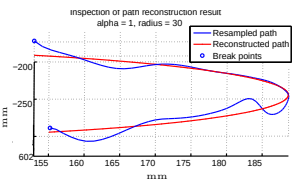
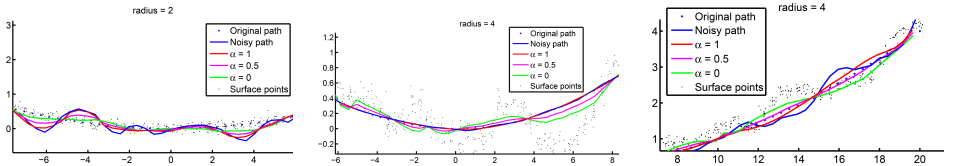


Fig. 7. Reconstruction of a path using no surface information. Path recorded using OptiTrack Flex 13 optical tracking system.

VI. Discussion

In Sec. IV and V, it has been shown that a large noise content in the surface compared to the path, reduces the usefulness of the surface scan for reconstruction purposes. If the surface used to enhance the path is of low quality, the reconstruction is likely to be better using only information in the path. Refer to Fig. 8 for a comparison between cases where varying quality of both path and surface is used. The figure indicates that

[4] http://www.naturalpoint.com/optitrack/products/flex-13/



(a) High quality surface and low quality path

(b) Low quality surface and high quality path

(c) Both surface and path are of low quality

Fig. 8. Results using different levels of α are shown for varying quality of path and surface.

the parameter α can be used to balance the amount of information used from the different data sources. If the surface is of higher quality than the path, a value of α close to 0 yields a better reconstruction, if the path however is of higher quality, a value of α closer to 1 is more likely to yield a good result. $\alpha \approx 0.5$ can be used if both the surface and the path is noisy. The noise in the two data sets are then likely to cancel out and yield a better average. All data sets are constructed from a second order polynomial with added colored noise to produce sets of varying quality. The results indicates that the tunable parameter α can be used to balance the relative confidence in quality between the two data sets.

The errors typically introduced by the user in an unprocessed path, in particular the high maximum error, makes the indicated path unsuitable as trajectory for an industrial robot for our purposes. After reconstruction, a path can typically be indicated and executed with a maximum error of 1mm, making it suitable for finishing tasks such as polishing and deburring using force control [17].

VII. CONCLUSIONS

An algorithm has been proposed which can significantly improve the structure of a point sampled curve in three dimensions, making it more suitable for execution on an industrial robot. This is achieved using a novel approach, where structural similarities in a related point cloud surface is exploited. The alignment between the surface and the curve may be assumed to be imperfect and the amount of reconstruction desired is tunable. The effectiveness of the algorithm has been verified in both simulations and experiments using data from a real 3D scanner.

REFERENCES

[1] K. Hartmann, R. Krishnan, R. Merz, G. Neplotnik, F. B. Prinz, L. Schultz, M. Terk, and L. E. Weiss, "Robot-assisted shape deposition manufacturing," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on.* IEEE, 1994, pp. 2890–2895.

[2] J. Wang, H. Zhang, and T. Fuhlbrigge, "Improving machining accuracy with robot deformation compensation," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on.* IEEE, 2009, pp. 3826–3831.

[3] D. J. Todd, *Fundamentals of robot technology.* Halsted Press, 1986.

[4] D. W. Eggert, A. W. Fitzgibbon, and R. B. Fisher, "Simultaneous registration of multiple range views for use in reverse engineering of cad models," *Computer Vision and Image Understanding*, vol. 69, no. 3, pp. 253–272, 1998.

[5] R. Fabbri and B. Kimia, "3d curve sketch: Flexible curve-based stereo reconstruction and calibration," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 1538–1545.

[6] F. Han and S.-C. Zhu, "Bayesian reconstruction of 3d shapes and scenes from a single image," in *Higher-Level Knowledge in 3D Modeling and Motion Analysis, 2003. HLK 2003. First IEEE International Workshop on.* IEEE, 2003, pp. 12–20.

[7] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques.* ACM, 1994, pp. 295–302.

[8] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Schematic surface reconstruction," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 1498–1505.

[9] I.-K. Lee, "Curve reconstruction from unorganized points," *Computer aided geometric design*, vol. 17, no. 2, pp. 161–177, 2000.

[10] I.-K. Lee, J. Wallner, and H. Pottmann, "Scattered data approximation with kinematic surfaces," *SAMPTA*, vol. 99, pp. 72–77, 1999.

[11] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[12] O. Ruiz, C. Vanegas, and C. Cadavid, "Principal component and voronoi skeleton alternatives for curve reconstruction from noisy point sets," *Journal of Engineering Design*, vol. 18, no. 5, pp. 437–457, 2007.

[13] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[14] L. E. Kavraki, 2013. [Online]. Available: http://cnx.org

[15] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1–4.

[16] K. Khoshelham, "Accuracy analysis of kinect depth data," in *ISPRS workshop laser scanning*, vol. 38, 2011, p. 1.

[17] C. W. Lim and P. Y. Tao, "Enhancing robotic applications in the industry through force control," Singapore Institute of Manufacturing Technology, Tech. Rep., 2010.

Bibliography

Besl, P. J. and N. D. McKay (1992). "A method for registration of 3-D shapes". *IEEE Transactions on pattern analysis and machine intelligence* **14**:2, pp. 239–256.

Birch, T. S. M. (Nov. 2012). *High Accuracy Capture for Small Volume*. NaturalPoint. URL: `http://forum.naturalpoint.com/forum/ubbthreads.php?ubb=showflat&Number=32117`.

Feldmar, J., N. Ayache, and F. Betting (1995). "3D-2D projective registration of free-form curves and surfaces". In: *Computer Vision, 1995. Proceedings., Fifth International Conference on*. IEEE, pp. 549–556.

Horn, B. K. (1987). "Closed-form solution of absolute orientation using unit quaternions". *JOSA A* **4**:4, pp. 629–642.

Johnson, A. E. and M. Hebert (1997). "Surface registration by matching oriented points". In: *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*. IEEE, pp. 121–128.

Kavraki, L. E. (2013). URL: `http://cnx.org`.

Khoshelham, K (2011). "Accuracy analysis of kinect depth data". In: *ISPRS workshop laser scanning*. Vol. 38, p. 1.

Kuijt-Evers, L., L Groenesteijn, M. De Looze, and P Vink (2004). "Identifying factors of comfort in using hand tools". *Applied Ergonomics* **35**:5, pp. 453–458.

Lim, C. W. and P. Y. Tao (2010). *Enhancing Robotic Applications in the Industry Through Force Control*. Tech. rep. Singapore Institute of Manufacturing Technology.

Lin, F.-T. (2004). "Optimal Handle angle of the fencing foil for improved performance 1, 2". *Perceptual and motor skills* **98**:3, pp. 920–926.

Pauly, M., M. Gross, and L. P. Kobbelt (2002). "Efficient simplification of point-sampled surfaces". In: *Visualization, 2002. VIS 2002. IEEE*. IEEE, pp. 163–170.

Pearson, K. (1901). “LIII. On lines and planes of closest fit to systems of points in space”. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**:11, pp. 559–572.

Rauch, H. E., C. Striebel, and F Tung (1965). “Maximum likelihood estimates of linear dynamic systems”. *AIAA journal* **3**:8, pp. 1445–1450.

Rusu, R. B. (2010). “Semantic 3d object maps for everyday manipulation in human living environments”. *KI-Künstliche Intelligenz* **24**:4, pp. 345–348.

Schoenmarklin, R. W. and W. S. Marras (1989). “Effects of handle angle and work orientation on hammering: I. Wrist motion and hammering performance”. *Human Factors: The Journal of the Human Factors and Ergonomics Society* **31**:4, pp. 397–411.

Seth, T. S. M. (Nov. 2012). *Location of non spherical marker*. NaturalPoint. URL: http://forum.naturalpoint.com/forum/ubbthreads.php?ubb=showflat&Number=53442&#Post53442.

Supercapacitor (Oct. 2012). Battery university. URL: http://batteryuniversity.com/learn/article/whats_the_role_of_the_supercapacitor.

Taubin, G. (1991). “Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**:11, pp. 1115–1138.

Thornton, C. L. (1976). *Triangular covariance factorizations for Kalman filtering*. PhD thesis NASA Technical Memorandum 33-798. University of California, Los Angeles–Engineering.

Todd, D. (1986). *Fundamentals of robot technology: an introduction to industrial robots, teleoperators and robot vehicles*. Kogan Page. ISBN: 9781850910695. URL: http://books.google.se/books?id=uLFZAAAAYAAJ.

Yang, G., I.-M. Chen, S. H. Yeo, W. K. Lim, *et al.* (2002). “Simultaneous base and tool calibration for self-calibrated parallel robots”. *Robotica* **20**:4, pp. 367–374.