Statistical Learning for Efficient Industrial Robot Control

by

Xiaowen Yu

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering-Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Summer 2018

Statistical Learning for Efficient Industrial Robot Control

Abstract

Statistical Learning for Efficient Industrial Robot Control

by

Xiaowen Yu

Doctor of Philosophy in Engineering-Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

This dissertation presents a series of statistical learning methods to improve the overall performance for industrial robots, which are always required to perform accurate and fast motions in sensing limited scenarios, and/or changing environments. Statistical methods emphasize on system uncertainties and adaptation to the environment by learning from the experience. Four aspects have been studied in detail including robust visual servo, optimal trajectory planning, intelligent system identification for feedforward compensation, and optimal gain tuning.

For visual servo requiring high precision and high speed but suffering from limited sensing capability and system uncertainties, a statistical learning approach based on sensor fusion and robust vision is adopted for sensor data filtering and on-line parameter adaptation. This method can compensate for the large sensing latency and estimate system parameters simultaneously. The algorithm is tested on an 6-axis industrial robot performing precision glass handling task but with only limited quality vision sensors. Trajectory planning is another fundamental problem for industrial robot, especially for robots operating in extremely limited workspace. In order to generate obstacle-free trajectories with shortest cycle time, a statistical learning method, Probabilistic Roadmap (PRM), is first used for optimal trajectory planning. An optimization problem is then formulated based on spline parameterization. The planning algorithm is tested on a 3-axis industrial robot, and the result shows the efficiency of the planning method. Statistical learning approaches are also studied for the precision motion control and vibration suppression for industrial robots. Model-based controller for parallel robots is always hard to design due to the complexity of the robots. A model based feedforward controller is designed with dynamic identification with a regression performed using data generated by experiments with designed trajectory. Also, an experiment based guided gain search is proposed with direct end-effector sensing for vibration suppression. An optimization scheme is utilized to guide the searching direction with experiments and data learning. It is proved by experiments that the tracking error and vibration are greatly reduced on an 8-link wafer handling robot.

To my little princess, Natalie.

Contents

List of Figures

List of Tables

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor and committee chair Professor Masayoshi Tomizuka for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I also want to express my special gratitude to him and his wife Miwako Sensei for being supportive for my newborn baby. Without his guidance and persistent help this dissertation would not have been possible. I could not have imagined having a better advisor and mentor for my Ph.D study.

I would like to thank my committee members, Professor Kameshwar Poolla and Professor Ruzena Bajcsy for their insightful comments and encouragement, and for their extensive personal and professional guidance. I would like to thank Professor Karl Hedrick, who served the committee chair of my qualifying exam. His class on nonlinear control heavily influenced my view on the subject of robotic control systems. I would also like to thank Professor Oliver O'Reilly and Professor Pieter Abbeel for sitting on my qualifying committee. Discussing with them were always inspiring and enjoyable, and their hard questions incented me to widen my research from various perspectives.

Special acknowledgments to my sponsors, Applied Materials, Inc. and Lens Technology for their financial support. It is them who made my research meaningful by providing interesting and valuable industrial projects. My sincere thanks also goes to Dr. Sanggyum Kim, Dr. Junwu Zhao, Mr. Alex Minkovich, and Mr. Nir Merry, who provided me opportunities to join their team as intern, and who gave me access to the laboratory and research facilities. Without their precious support it would not be possible to conduct this research. In particular, I am very grateful to Dr. Cong Wang for his countless help and mentoring on my research projects.

I would like to thank my friends both in US and in China, thank you for listening, offering me advice, and supporting me through this entire process. Special thanks to my intern, Thomas Baker. It is really admirable that I see the project from conception to completion with his hard work and brilliant ideas.

I want to thank all the MSC members for the stimulating discussions, and for all the fun we have had in the past several years. Special thanks to Wenjie Chen, Xu Chen, Michael Chan, Kan Kanjanapas, Chi-Shen Tsai, Pedro Reynoso, Wenlong Zhang, Yizhou Wang, Raechel Tan, Minghui Zheng, Junkai Lu, Chung-Yen Lin, Chen-Yu Chan, Changliu Liu, Yaoqiong Du, Kevin Haninger, Shiying Zhou, Dennis Wai, Shuyang Li, Te Tang, Hsien-Chung Lin, Yu Zhao, Yongxiang Fan, Wei Zhan, Cheng Peng, Daisuke Kaneishi, Zining Wang, Kiwoo Shin, Liting Sun, Yu-Chu Huang, and Zhuo Xu.

Last but not the least, I would like to thank my family. I would like to thank my parents and my grandparents, whose love and guidance are with me in whatever I pursue. I always knew that you believed in me and wanted the best for me. They are the ultimate role models. Most importantly, I wish to thank my loving and supportive husband, Yu Zhao. Words cannot express how lucky I am to have him as both my husband and best friend.

Chapter 1

Introduction

1.1 Motivation and Contribution

Nowadays, most industrial robots have fairly primitive sensing. Scenarios where a favorable sensing system is not available are very common. In regard to the limited sensing capability, robots must contend with environment uncertainty and adapt to changes in the environment. On the other hand, in today's globally competitive business environment, the needs of an ever-changing market require manufacturers to continually enhance the production line effectiveness. This requires industrial robots to perform at faster speeds, and with shorter cycle time and idling time without losing performance. Statistical methods [1] emphasize on uncertainties and adaptation to the environment by learning from the experience. This dissertation presents a series of statistical learning approaches to improve the overall performance for industrial robots with limited sensing capability. Four contribution of this dissertation research are: 1) statistical learning and sensor fusion for robust visual servo, 2) optimal trajectory planning, 3) intelligent system identification for feedforward compensation, and 4) optimal gain tuning by learning from the experience.

Statistical Learning and Sensor Fusion for Robust Visual Servo

For visual servo requiring high precision and high speed but suffering from limited sensing capability and system parameter uncertainties, Chapter 2 presents a statistical learning approach based on sensor fusion and robust vision for sensor data filtering and on-line parameter adaptation. For visual servo for manufacturing applications, the requirements are becoming more and more stringent. It is desired that the vision system is low cost and easy to implement yet performs with high precision and high speed. However, with limited sensing capability as well as system uncertainties and disturbances, accurate and fast visual servo is difficult to achieve. A robust vision based on image learning is first proposed for object detection and disturbance rejection. A non-linear regression is performed to single out those objects with highest probability to be the objects of interest. Then a Bayesian inference based filtering algorithm, Unscented Kalman Filter (UKF), is implemented to compensate for the sensing limitations, such as low-sampling rate and large delay. For parameter adaptation, a major limitation in adaptive control is that the model must be linear with respect to the unknown parameters, which is not satisfied in most cases. Thus a dual estimation scheme along with the UKF is applied for on-line parameter estimation. This method dose not require any linearity with respect to the model parameters , and thus provides efficient parameter learning. A closed-loop controller based on sensor fusion is then developed for a fast and accurate performance. This approach is validated by experiment of a glass piece "pick and place" task using a Fanuc industrial robot manipulator [2, 3]. The result is given in Chapter 2.

Optimal Trajectory Planning

Trajectory planning is a fundamental problem for industrial robot. It is particularly challenging for robots operating in extremely limited workspace without workspace sensing. In order to generate a obstacle-free trajectory with shortest cycle time and sufficient smoothness for industrial robots in such environment, Chapter 3 presents a statistical learning method, Probabilistic Roadmap (PRM), for optimal trajectory planning. Automation of robot trajectory generation is important for robots performing multiple tasks under several working constraints yet with limited sensing capability. For example, the robot should not collide with any obstacles while maintaining a smooth motion. Also, the velocity and acceleration should be constrained in workspace to ensure the payload safety. Actuators' limits also need to be satisfied. For manufacturing productivity, the trajectory needs to be further optimized for shorter cycle time. In most current industrial applications, however, only part of the constraints are considered due to the complexity of the problem. In Chapter 3, a PRM method is first used to generate a road map of the workspace so that multiple trajectory planning can be performed. Then an optimization problem is formulated based on spline parameterization with all the working constraints. For fast trajectory planning, it is favorable that the optimization problem starts with a reasonable initial point. A method for finding such an initial point is given in details. The planning algorithm is tested on a 3-Degree of Freedom (DOF) industrial robot, and the result shows the efficiency of the planning method [4].

Intelligent System Identification for Feedforward Compensation

For some industrial robotic machining and precision delivery tasks, robots are required to perform accurate tracking. With motor-side encoders, the motor-side tracking can be achieved by precision motion control. However, precision tracking control requires accurate force/torque compensation. Challenges come from the complex nonlinear coupled multi-body dynamics that hard to model precisely. In order to obtain high-fidelity models, Chapter 4 discusses in detail the learning method for dynamic modeling and parameter identification. Although most of the system identification follow a standard procedure, it is worth noting that for a complex robot multi-body dynamics, problems with model decoupling and regression emerge. A data-driven approach is thus proposed to solve the problem efficiently without analytical derivation which requires heavy computational load. The modeling technique is applied to a 8-link parallel robot, whose dynamics is even more complicated than a serial robot due to the closed-loop chain structure. The robot dynamics is first modeled using Lagrangian method, and then transformed to a decoupled form for identification. Data-driven method is used to optimize the regression matrix for identification [5]. The identified model is used in a feedforward controller and tested by multiple experiments.

Optimal Gain Tuning

For the motion control of industrial robots, the end-effector performance is of the ultimate interest. However, industrial robots are generally only equipped with motor-side encoders. For precision motion control, controller gains, such as PID gains, are usually tuned based on motor-side performance. With a non-optimal PID controller, the robot may exhibit a considerable vibration and large tracking error on the end-effector leading to a long cycle time. Chapter 5 presents a statistical learning algorithm for gain tuning by directly measuring the performance of the end-effector. Traditional gain tuning methodologies include empirical methods such as Ziegler-Nichols [6] and analytical methods such as shaping of frequency response [7]. However, their effectiveness for nonlinear systems such as robots and for reduction of vibration is not conclusive. Thus an experiment guided gain search method based on optimization is proposed by direct end-effector sensing, which can provide fast gain tuning and in the mean time guarantee optimality. A batch of empirical data are first collected from a cost-effective accelerometer on the end-effector, then a statistical learning algorithm is performed to search for an optimal gain and provide the guidance for data generation [8]. The gain tuning method is also tested on the 8-link parallel robot in Chapter 4, and the results show that the optimal gain can be obtained within two iterations.

1.2 Dissertation Outline

The rest of this dissertation is organized as follows. Chapter 2 will present a filtering and parameter learning algorithm for robot robust visual servo with limited sensing capabilities. A closed-loop controller based on robust vision and sensor fusion is also developed. Chapter 3 will introduce a statistical learning method, Probabilistic Roadmap (PRM) for path generation. A constrained optimization problem is also formulated for optimal trajectory planning. Chapter 4 will discuss the intelligent learning method for dynamic modeling and parameter identification. A data-driven approach for complex multi-body dynamics is also presented for efficient model learning. Chapter 5 will present a statistical learning algorithm for robot controller gain tuning with direct end-effector measurements. Data are collected by experiments followed by a learning step to provide the guidance for data generation and also the optimal gain. Finally, Chapter 6 will summarize the contributions of this dissertation and give possible future work.

Chapter 2

Statistical Learning and Sensor Fusion for Robust Visual Servo

2.1 Introduction

Robotic automation is normally applied to repetitive and highly-structured tasks, which correspond to about 45% of industry jobs [9]. Higher value-added elements and complex tasks in the manufacturing process, such as 3D electro-mechanical-optical parts assembling, are hard to automate and still require human workers. This trend is accelerating and is creating a demand for automated systems that can feed and assemble complex 3D parts. A key element in these systems is vision guidance, using machine vision to locate and identify parts. However, traditional vision guidance with high precision assembly operations is limited by the accuracy of the robot it controls. To achieve high accuracy, such as one or two microns, an extremely expensive, high accuracy robot must be used. While many robots have high-resolution encoders, their ability to move to an absolute position commanded by a traditional vision system can be quite limited due to manufacturing variations, thermal expansion and mechanical effects. Also, traditional vision guidance always relies on the quality of the end-effector. The accuracy cannot be achieved if the robot uses low-cost end-effectors, such as vacuum grippers whose gripping suffers a large uncertainty [10].

On the other hand, visual servo closes the control loop visually and can achieve accuracy based on the robot encoder resolution rather than its absolute accuracy. It can also adapt to inaccurate grippers since both the part and target are viewed and only their relative positions are important. This flexibility can also be used to automate processes where parts and targets change in real time, such as the insertion of sutures into the hole of suture needles. As the part or target change, the vision system can detect the variations and adjust accordingly. In most current industrial applications, however, visual servo is controlled by a look-then-move method, which cannot satisfy the needs for real-time vision guidance. For industrial vision guidance systems, the visual servo need to be fast and robust while maintaining high precision.

Challenges come from various aspects. First, with low-cost cameras, object detection become intractable due to the limited quality of the camera. One of the biggest challenges sensors face is detecting transparent clear object, such as glass pieces. The detection of the glass piece is challenging because the glass is transparent and reflective, leading to unpredictable contrasts between the glass and the background. Reflections impose additional problem of adding contours of the surrounding environment. McHenry et al. [11] use hierarchical classifiers and distortions occurring at the boundaries of transparent objects as visual cues. This method, however, only works for detecting curved glass. Other methods [12] involve using the structured light camera which are too costly and slow to implement. In Section 2.3, small, flat, transparent glass pieces are robustly detected with image data learning based on a Levenberg-Marquardt nonlinear least-squares [13].

Second, robot Tool Center Point (TCP), which usually refers to the center point of the robot end-effector, is hard to calibrate when special gripper is used for accurate object handling. For example, the gripper is non-fixed to the robot, which moves along with the object it is handling. Standard TCP calibration [14] procedure which jogs the robot's TCP to a single point with four approaches only works for robots equipped with tip devices. Another way is to use external camera by calibrating the camera's position as well as the transformation between camera and the TCP [15]. Such method only works for robots with fixed transformation to the gripper. In order to do calibration in such limited condition, a Maximum A Postiori (MAP) method [16] and a projective transformation method are proposed in Section 2.3.

Third, the speed of visual servo is limited by the low-cost industrial robot vision system which is subject to considerable latency and limited sampling rate. In fact, due to the image processing is usually time-consuming depending on the complexity of the feature extraction, the delay from visual feedback will be further extended. To compensate for such issues, in Section 2.4, a nonlinear (due to the nonlinearity of the robot manipulator system) state filter is desirable. Wang et al. [17] use Extended Kalman Filter (EKF) to compensate for the low-sampling rate and latency from a position sensitive detector (PSD) camera, which, in fact, is not a real visual servo case. Besides, EKF [18] suffers from several flaws [19], and it may result in estimation divergence. Thus an improvement, Unscented Kalman Filter (UKF) [20], is proposed in Section 2.4 for on-line state estimation.

Last but not least, visual servo suffers from stability and robustness issues in the presence of system parameter uncertainties, leading to slow and diverging performance. Thus many researchers have developed adaptive controllers to compensate for the parameter uncertainty. Papanikolopoulos et al. [21] developed an online estimation method for the depth of the feature point. Wang et al. [22] developed an adaptive controller for unknown camera parameters based on a dynamic visual servo. However, most of the adaptive controllers are based on the camera model linear parameterization, which always involves heavy computational load on model derivation. In Section 2.5, a dual-estimation method [19] with UKF is proposed for on-line parameter identification, which does not require any linear parameterization of the model.

The robust visual servo is tested on an industrial application, which is the precise and fast glass piece placing for cell phones' front glass. The desired placing accuracy is sub-millimeter. The detection accuracy with only an Eye-to-Hand camera mounted by the robot is about 3mm, with which, the placing accuracy cannot be achieved. The robot can only put the glass piece close to the target position. Thus a two step approach is introduced for this application: the robot first put the glass piece close to the target position with the Eye-to-Hand camera, and then guide the glass piece to the target position with Eye-in-Hand cameras using visual servo. Two Eye-in-Hand cameras are thus mounted on the robot end-effector in a front-back manner to detect the entire piece of the glass (Fig.2.1). All cameras are chosen as low cost equipment in order to make the system cost not excessive. The 3mm accuracy of the first step is necessary so that the visual servo will work since it is a local method, and large initial error will result in divergence.

A closed loop controller based on visual servo is used to guide the glass piece to the target position. Most of the visual servo techniques [23], [24],and [25] use points as image features since they are easy for controller design [25]. However, in the glass placing task, not enough point features are available (explained in Sec.2.4). Line features, on the other hand, can provide full observation of the scene and richer geometric information for a more accurate placing performance. However, few literature reports the use of line features. Andreff et al. [26] use 3D line features based on binormalized Plücker coordinates, which is not for projected lines. Espiau et al. [27] mention the use of line features yet the derivation is problematic and no experimental result is given. In this Chapter, a controller of visual servo using line features is properly designed, and experimental result is provided.

In the rest of this Chapter, Section 2.2 introduces the system setup of the task. Section 2.3 presents statistical learning algorithm for transparent object detection as well as the calibration for vision system. Section 2.4 presents the state estimation and parameter adaptation using dual rate UKF with dual estimation. Section 2.5 gives the experimental results of the proposed algorithms. Section 2.6 concludes the chapter.

2.2 System Setup and Workflow

The background of the robot glass handling task is from the cell phones' protective glass manufacturing line. Initially, raw glass pieces need to be ground to a desired thickness with grinders, such as one in Fig.2.2. The desired thickness is achieved by the use of a mould called toothboard, which is a thin plastic plate. In Fig.2.2, toothboards (see Fig.2.1 for more details) with the desired thickness (around 1 mm) are on the surface of the lower grinder. After placing glass pieces into the slots of the toothboard, the upper grinder moves downward and engages with the lower grinder. By conducting rotational motions of both upper and lower grinders, the glass pieces are ground on both sides simultaneously. The major issue in this process is that the placing task is manual. Also, since the tolerance of placing glass pieces into the slot is very tight (sub-millimeter), the human workers need to place them individually.

One grinder can only process approximately 20 pieces in a grinding cycle. Thus, hundreds

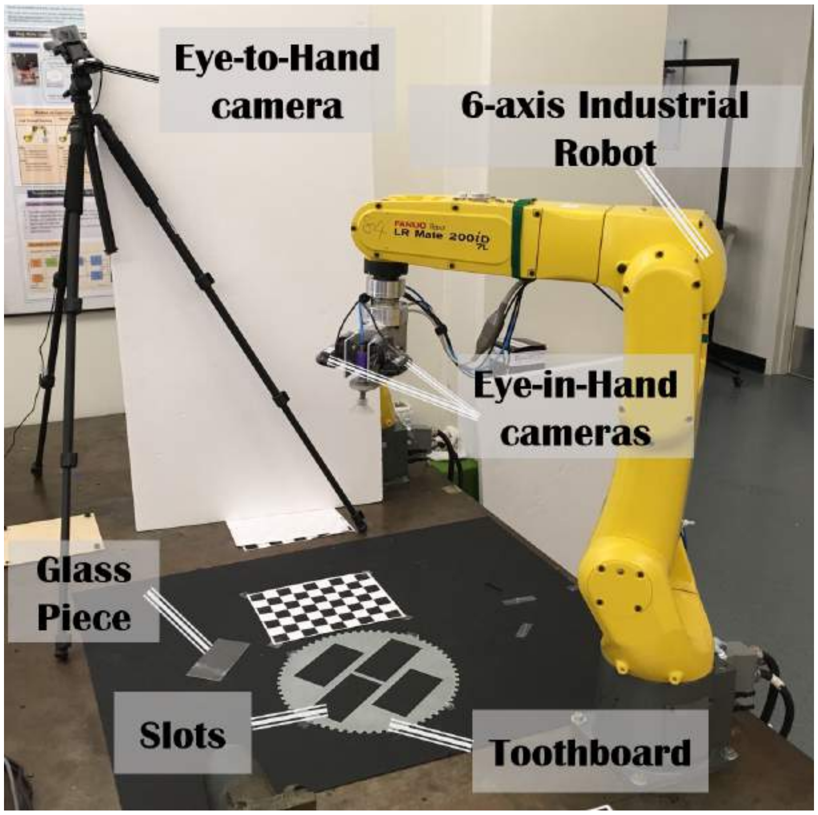


Figure 2.1: System overview

of grinders are in demand. Also, due to the complexity of the glass handling, one worker can only operate one grinder at a time leading to a huge demand on human resources. To study possible automation of the process, we use a six-axis robot manipulator equipped with vision sensors as shown in Fig.2.1. An Eye-to-Hand camera is installed to detect glass pieces and slots, which are randomly placed in the workspace (Fig.2.1). Two Eye-in-Hand cameras are mounted on the robot end-effector in a front-back manner to detect the entire piece of the glass (Fig.2.1). All cameras are chosen as low cost equipment in order to keep the system cost not excessive.

Apart from the cameras, the end-effector is also equipped with a suction cup and a venturi pump. This enables the robot to securely pick up the glass, manipulate it, and place it back on the workspace's surface. The suction cup's vacuum is generated with the venturi pump, located directly above the cup. Airflow and pressure required for the Venturi effect to take place within the pump is supplied by a compressor and a small lightweight tube. A schematic drawing is shown in Fig. 2.3

Additionally, a paper checkerboard with known sizes is placed into the workspace and serves as the world frame. The glass piece and the toothboard are initially positioned arbi-

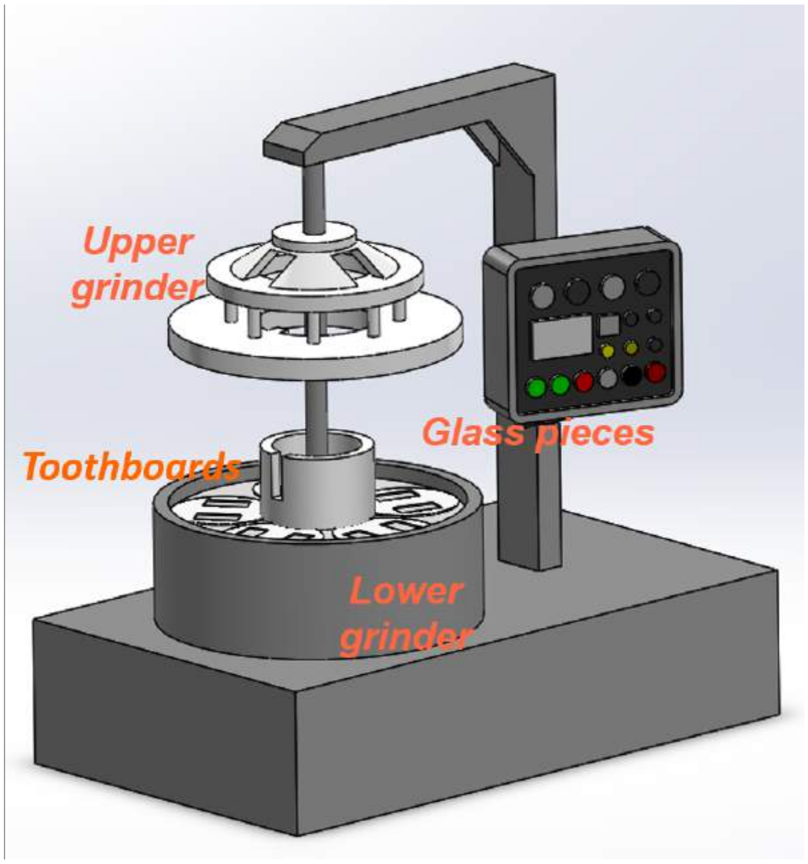


Figure 2.2: The Grinder

trarily in the workspace. The Eye-to-Hand camera is able to oversee the whole workspace and the robot arm is placed so that it may reach the complete workspace. As previously described, the aim is to place one glass piece into one of the four slots of the toothboard.

The work flow is designed to be a two step approach. First an open loop robot control approach ensures the glass piece is picked up by the manipulator and positioned above a slot. For this, the Eye-to-Hand camera detects the position and orientation of the glass and the slot relative to the checkerboard (world frame). The transformation from the world frame to the robot's base frame along with its forward kinematics are known. The transformations from the manipulator's flange to the Eye-in-Hand camera (camera frame) and suction cup (tool frame) are calibrated and measured respectively. With this knowledge, a trajectory can be generated, enabling the robotic arm to move to the glass' location, pick it up and position it above the desired slot.

In a second step, a closed loop robot control approach is chosen to correct the first step's inaccuracies and finally place the glass into the slot. This is achieved by detecting the glass and the slot with the Eye-in-Hand cameras, the control law's sensory input, and by

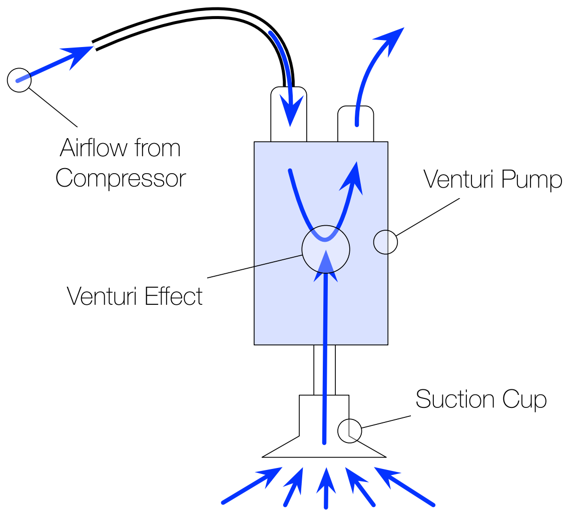


Figure 2.3: Generating suction with a venturi pump

formulating the control error and the control law, an imaged based visual servo approach. An overview of the main steps of the work flow is given in Fig. 2.4.

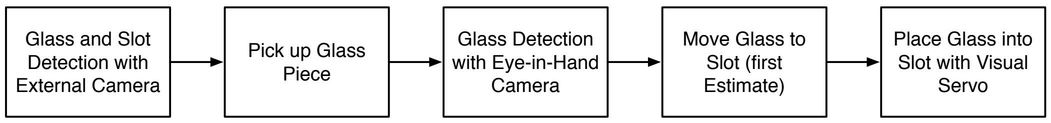

Glass and Slot Detection with External Camera
Pick up Glass Piece
Glass Detection with Eye-in-Hand Camera
Move Glass to Slot (first Estimate)
Place Glass into Slot with Visual Servo

Figure 2.4: Main steps of work flow

2.3 Object Detection and Tool Center Point Calibration

Detection with Eye-to-Hand Camera

An image acquired from the Eye-to-Hand camera is given in Fig.2.5 (excluding the checkerboard). By performing edge detection and clustering, the acquired image is separated into several clusters with each cluster containing a set of image points. To identify the clusters of slots and glass piece, the known geometry of the glass is fitted into each obtained cluster with Levenberg-Marquardt nonlinear least-squares regression.

Clustering

The setup of the Eye-to-Hand camera is in Fig.2.6. The checkerboard is used as a reference frame that relates the camera and the robot, and is used only once for calibration, then it is removed from the workspace. The acquired image is first rectified with the calibration and converted to a binary image by performing Canny edge detection. The entire image, after excluding points outside the workspace, is then separated into individual clusters by clustering with an agglomerative hierarchical cluster analysis. At the beginning of clustering, each point of the image is a single cluster. Then each pair of closest clusters are merged iteratively until a pre-set maximum number of clusters is reached. Noise and small clutter can be further reduced by removing clusters with a small number of points. The result is shown in Fig.2.7, which contains clusters of glass and slots, and remaining unwanted clusters need to be further removed.

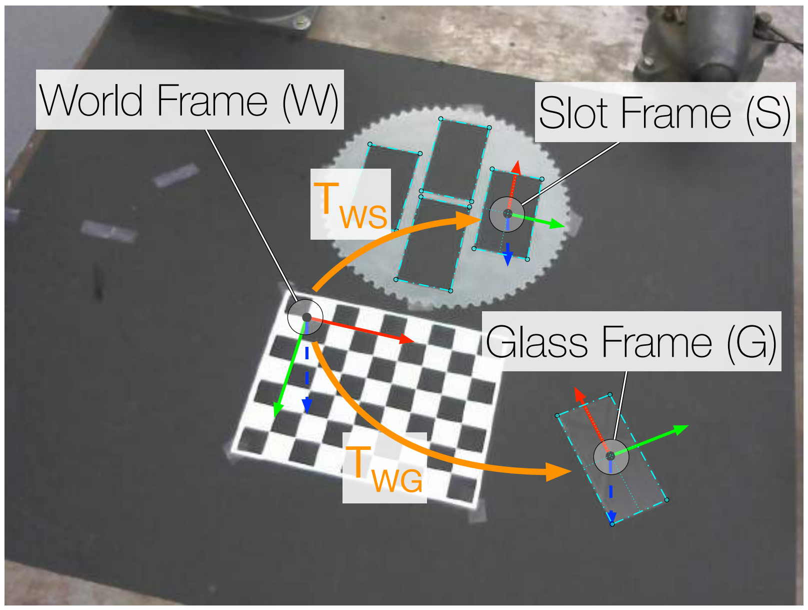


Figure 2.5: Image acquired from camera (without checkerboard)

Best Fitting by Non-linear Regression

The unwanted clusters can be removed by fitting each cluster with the known rectangular shape of the glass, which is formulated as an optimization problem:

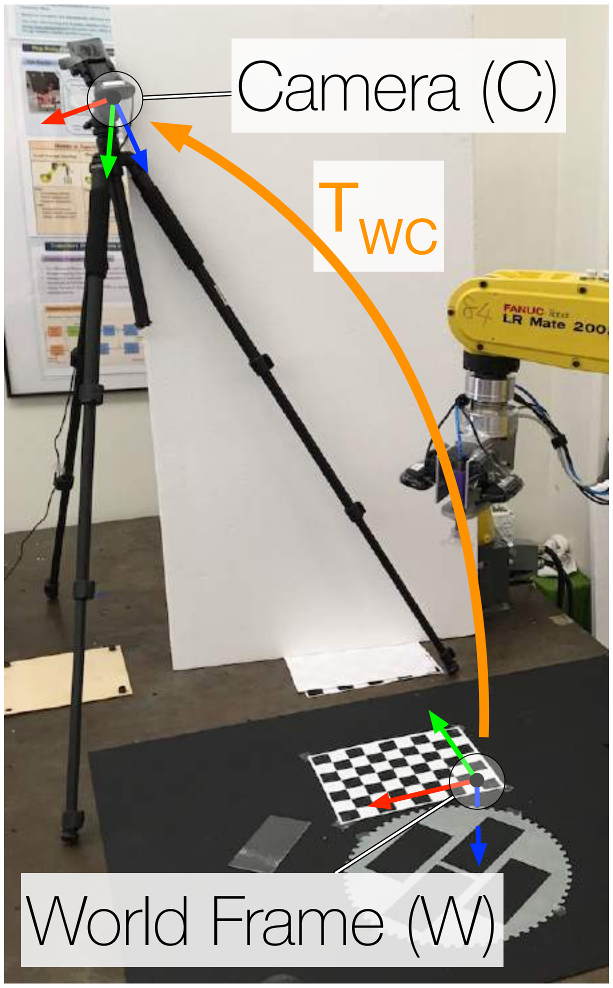


Figure 2.6: Eye-to-Hand camera setup

$$\boldsymbol{D} = \min_{\boldsymbol{\beta}} ||\boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{Q}(\boldsymbol{\beta})\right)||^2 \tag{2.1}$$

where $\boldsymbol{\beta}$ is the variable for the position and orientation of the known size rectangle:

$$\boldsymbol{\beta} = [\boldsymbol{X}, \boldsymbol{U}]^T \tag{2.2}$$

where $\boldsymbol{X} = [x, y]^T$ is the position vector of the rectangle's center, and $\boldsymbol{U} = [u, v]^T$ is the orientation vector of the rectangle's principle direction. The rectangle is discretized to a set of points $\boldsymbol{Q}(\boldsymbol{\beta})$ (Fig.2.8) at a density of 1mm. In Eq.(2.1) $\boldsymbol{x}$ is the set of points of the cluster, and $\boldsymbol{f}$ represents the distances from the rectangle ($\boldsymbol{Q}(\boldsymbol{\beta})$) to the cluster ($\boldsymbol{x}$). A k-nearest neighbor algorithm [28] is used to calculate the distance between each point

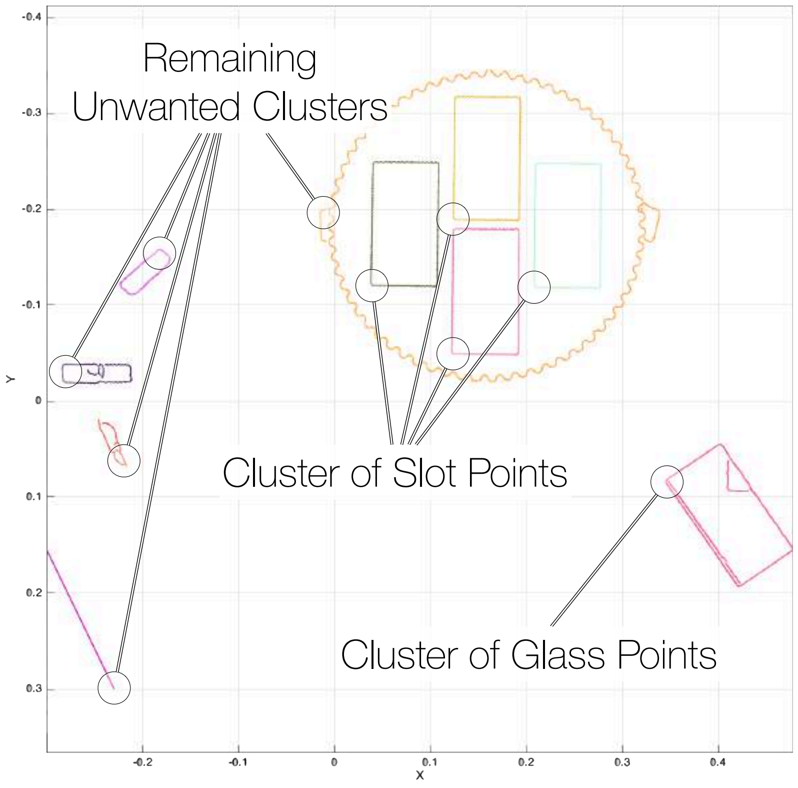


Figure 2.7: Clustering of the rectified Eye-to-Hand image

in $\boldsymbol{Q}(\boldsymbol{\beta})$ and its closest point in the cluster $\boldsymbol{x}$. For good solution and fast convergence of the optimization problem, the initial variable $\boldsymbol{\beta}_0$ should be carefully selected. The cluster's centroid and principle direction are chosen as $\boldsymbol{X}_0$ and $\boldsymbol{U}_0$ respectively. The optimization problem is solved with a non-linear least-squares Levenberg-Marquardt algorithm, and the fitting result is shown in Fig.2.9 where red lines are image clusters and blue rectangles are fitting result. Cluster with large fitting distance($\boldsymbol{D}$) exceeding a thresholds is not a glass piece or slot and is removed from the result.

Detection with Two Eye-in-Hand Cameras

When the first step is finished, the glass piece is placed close to the slot, and the second step, closed loop visual servo is activated. In the closed loop visual servo, each Eye-in-Hand camera sees a half of the glass piece as well as the slot (Fig.2.10 right). The detected edges of the glass piece and the slot are extracted as line features as feedback information. The glass piece stays stationary in the image while the slot moves along with the robot during placing. The goal is to align the slot with the glass piece in both images of Eye-in-Hand cameras with the motion of the robot.

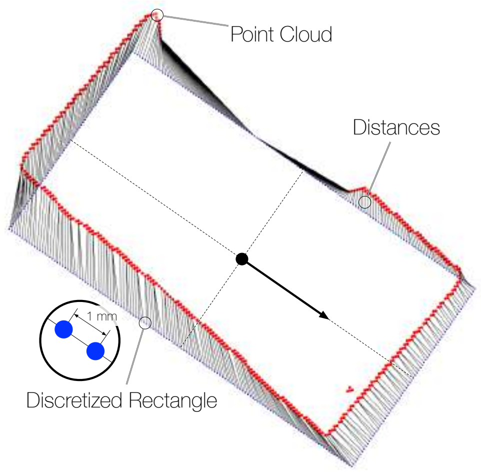


Figure 2.8: Fitting with discretized rectangle

The edge detection faces the difficulty of the glass being transparent and reflective. It is extremely hard to extract glass edges when glass and slot are in the same image due to the almost invisible edges of the glass (Fig.2.10 right). Thus the detection is separate for the glass and slot. Glass edges are first detected in a plain background, and slot edges are detected in the presence of the glass (Fig.2.10).

When detecting the glass piece in a plain background, the glass should be as little transparent and reflective as possible, so the contrast between the glass and the background is maximized. However, when detecting slot edges with the glass piece in presence, it is desirable for the glass to have the opposite properties to reduce the occlusion to the slot. Much research [11], [12] and effort has gone into methods of separating reflections from the background in images, and while the results are remarkable, they all have a high processing time in common. Instead, in this Section, a much simpler and faster approach is chosen. A green piece of fabric, as shown in Fig.2.11, is installed above the workspace to be advantageous in both cases.

Detection of The Glass

The detection procedure is as follows: first the green channel of the RGB image is extracted and a Sobel edge detection is performed to obtain a binary image. Then points that make up a convex hull of the binary image are singled out. By performing a Hough transform [29] on the image, three lines with the strongest Hough value representing the glass' three edges are detected, as shown in Fig.2.12.

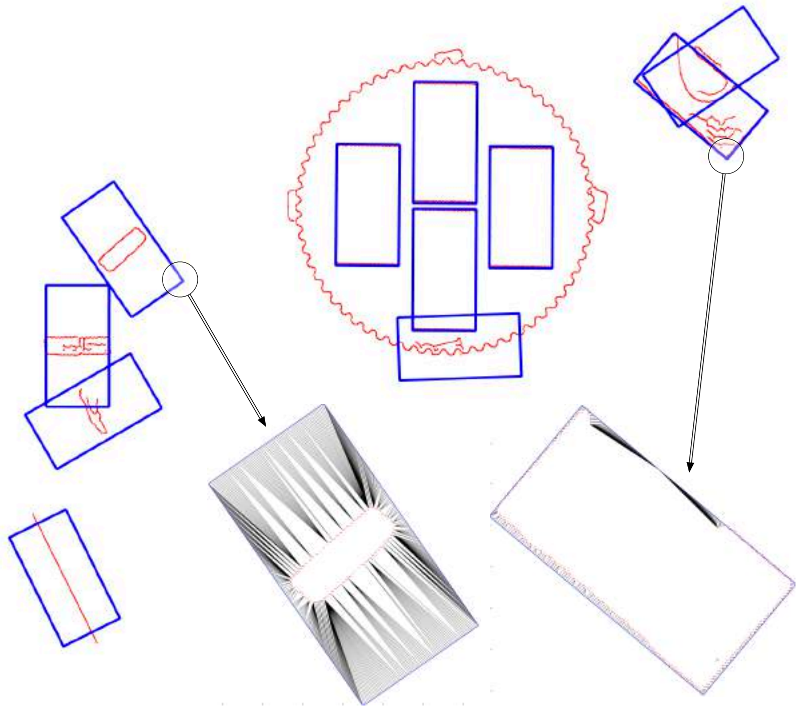


Figure 2.9: Identify glass and slots from regression results (Bad fitting: middle left; Good fitting: middle right; Black lines are distance)

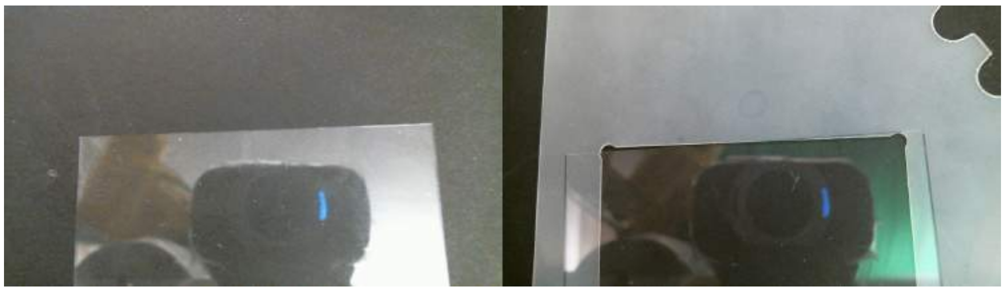


Figure 2.10: Detection of the glass and slot separately

Detection of the Slot

The precedure of detecting slot is similar to the detection of the glass piece except that instead of green channel, the red channel of the RGB image is extracted, shown in Fig.2.13 left. In the red channel, the contrast between the slot and the background is substantially enhanced and the results of detection are shown in Fig.2.13 right.

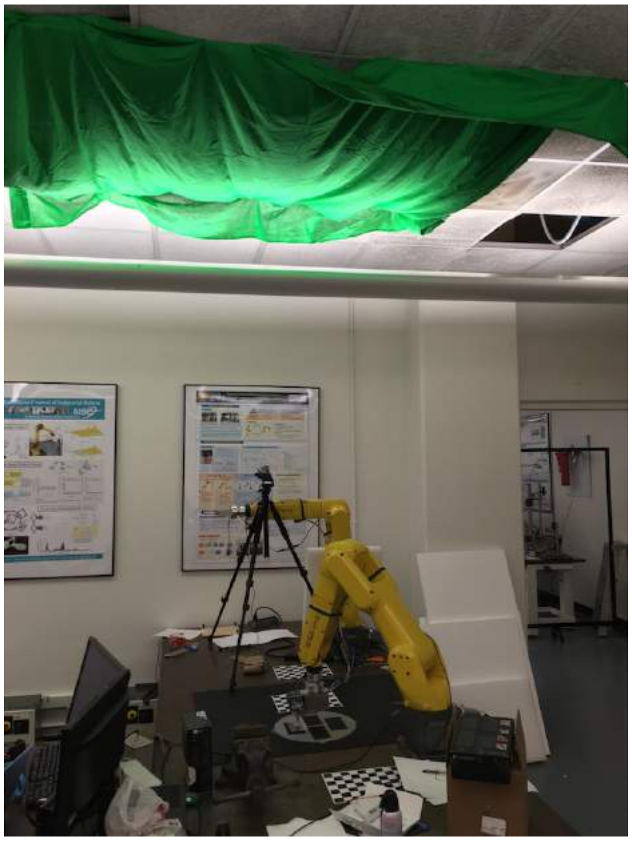


Figure 2.11: Change lighting conditions of workspace

Tool Center Point Calibration

After the robot picks up a glass piece, the loading part can be formulated as a placing task, in which the robot places the glass piece into a slot. However, system uncertainties associated with this task reduce the placing accuracy. First of all, the pick up point (which is placed at the center of the suction cup) relative to the robot's tool flange frame (the robot's 6th joint frame) cannot be calibrated very accurately. Even if it were precise, due to the limitations of the camera's resolution as well as the disturbance at the pick up step, it cannot be guaranteed that the glass is picked up directly at its center (Fig. 2.14). The resulting placing error will be further enlarged when the slot location is not detected accurately. To address such issue, instead of calibrating from the center of the suction cup to the robot's 6th joint frame, the glass frame (after picked up) relative to the robot's 6th joint frame should be calibrated which can be catagorized into a Tool Center Point (TCP) calibration problem with TCP being the glass frame. The standard TCP calibration procedure can be found in [14]. It jogs the robot's TCP to a single point at four approach points. A regression is performed to calibrate the TCP. Another way is to use external camera [15]. By calibrating

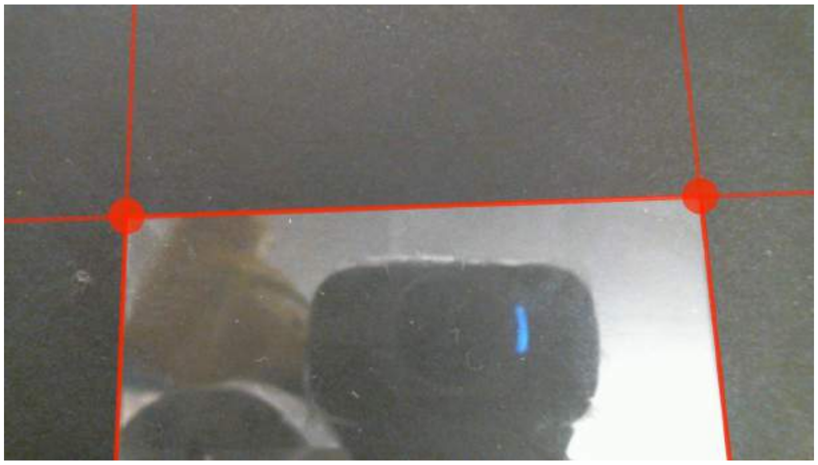


Figure 2.12: Detection of the glass edges

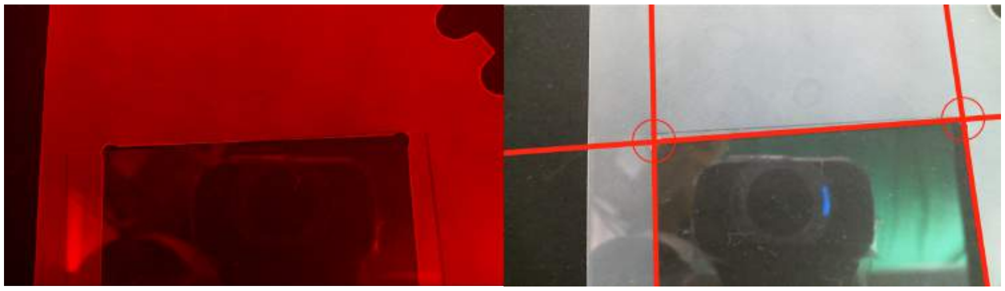


Figure 2.13: Detection of the slot edges

the camera's position as well as the transformation between camera and the TCP (by feature point extraction), the TCP is calibrated. However, none of the existing methods for TCP calibration can be directly utilized in this placing task since the TCP is not fixed relative to the robot's tool flange frame due to the uncertainty of the pick-up. In this Section, two TCP calibration methods are presented with one using Maximum A Posteriori (MAP) method and one using projective transformation.

TCP Calibration by MAP Method

To calibrate the glass frame relative to the robot's tool flange frame, two frame transformations need to be introduced as shown in Fig.2.15. One is the transformation T_{6c} from the robot's tool flange frame to the camera frame, and the other one is the transformation T_{cg} from the camera frame to a virtual glass plane frame. The origin of the glass plane is determined by a checkerboard put on the virtual plane (a metal plate). With the definition

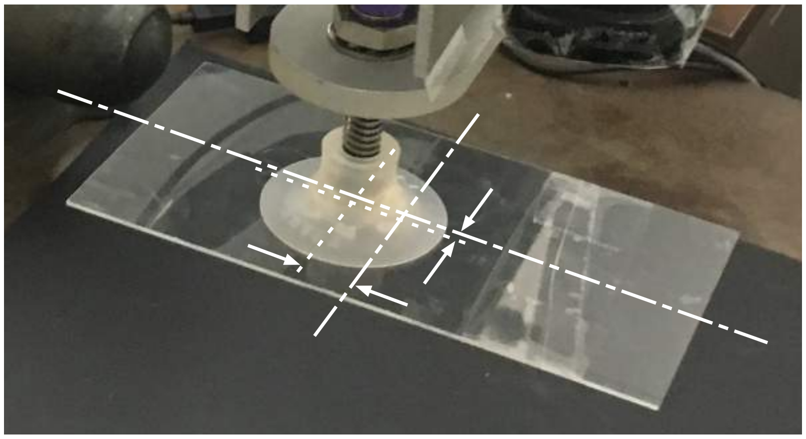


Figure 2.14: Glass piece picked up off-centre

of the two transformations, the first step of the glass placing task can be decomposed into four steps, as shown in Fig. 2.16.

First the glass piece is detected in the image by the Eye-in-Hand camera. With T_{cg}, the glass position in the glass plane frame can be determined. Then the glass frame relative to the robot's 6th joint frame is obtained with T_{cg} and T_{6c}. Finally with the robot kinematics, the glass piece can be placed to the slot. From the placing workflow, the key issue is the pre-calibration of T_{6c} and T_{cg}. In this Section, a prior guess of T_{6c} and T_{cg} is initialized by a standard checkerboard detection method. Then the glass piece which is attached to the robot end-effector, is put at several known positions relative to the robot's base frame as observations. A Maximum A Posteriori (MAP) method which maximizes the posterior probability, given the prior guess of T_{6c} and T_{cg} as well as observations, is performed to update T_{6c} and T_{cg}.

To obtain the prior guess of T_{6c} and T_{cg}, a checkerboard detection method is adopted due to its simplicity. As shown in Fig.2.17, two checkerboards are used to calibrate T_{6c} and T_{cg} respectively.

Checkerboard 2 in Fig.2.17 is at a known position relative to the robot's base frame, which is used for the calibration of T_{6c}. The equation for this calibration is:

$$T_{0b} = T_{06}T_{6c}T_{cb} \quad \rightarrow \quad T_{6c} = T_{06}^{-1}T_{0b}T_{cb}^{-1} \tag{2.3}$$

where T_{0b}, T_{cb}, T_{06} are the transformations from the robot's base frame (0) to the checkerboard frame (b), from the camera frame (c) to the checkerboard frame, and from the robot's base frame to the robot's tool flange frame (6) respectively. T_{0b} is known, T_{cb} is calibrated with the Matlab Computer Vision System Toolbox, and T_{06} is obtained from the robot's forward kinematics. By taking images of this checkerboard at different positions with the robot, a Least-Square is performed to get the best estimation of T_{6c}.

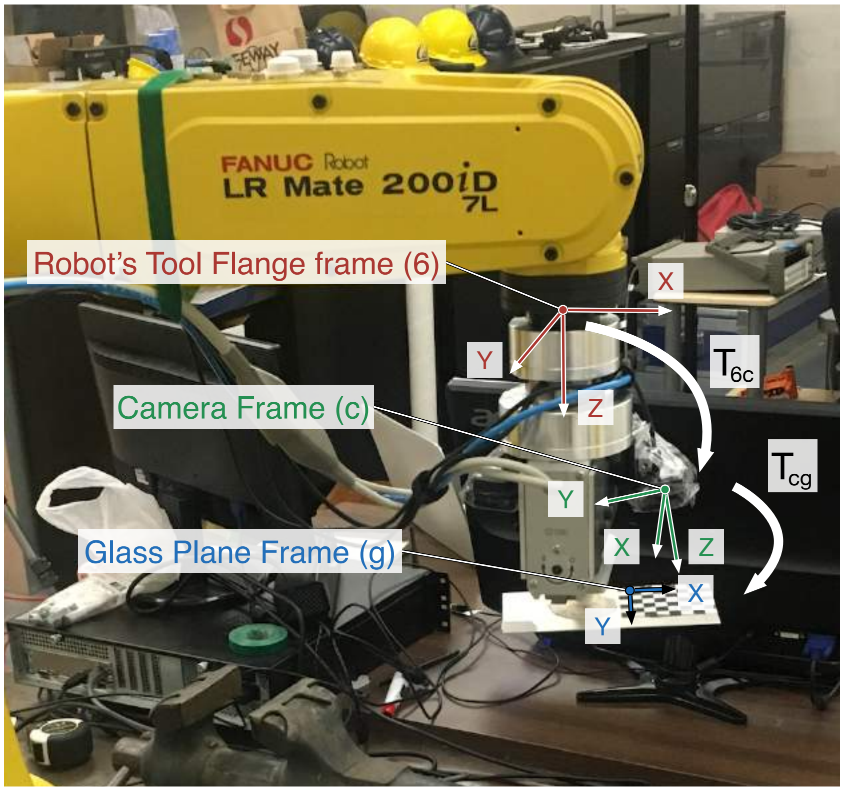


Figure 2.15: Transformation of T_{6c} and T_{cg} as well as glass plane

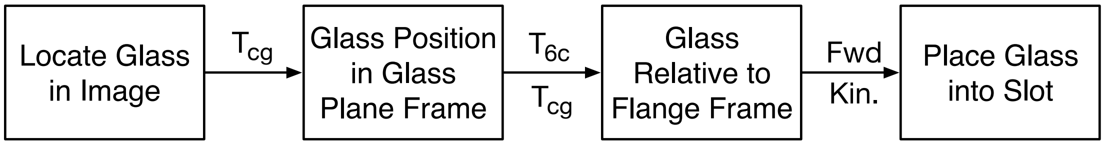

Locate Glass in Image
Tcg
Glass Position in Glass Plane Frame
T6c
Tcg
Glass Relative to Flange Frame
Fwd Kin.
Place Glass into Slot

Figure 2.16: Work flow of placing glass piece in slot

Checkerboard 1 in Fig.2.17 is located on the glass plane. By taking images of this checkerboard, T_{cg} is calibrated with the Matlab Computer Vision System Toolbox.

By putting the glass piece (attached to the robot end-effector) to five known positions on the table, five observations are obtained which are the known transformations from the checkerboard to the glass plane $T_{bg_i}, i = 1, \dots, 5$ (also to the robot's base frame), and $T_{bg} = T_{b0}T_{06}T_{6c}T_{cg}$. Assume that T_{b0} and T_{06} are known (forward kinematics), the observation is converted to $T_{6g} = T_{06}^{-1}T_{b0}^{-1}T_{bg}$. With the observation T_{6g}, the observation model is:

$$T_{6g} = T_{6c}T_{cg} + noise, \tag{2.4}$$

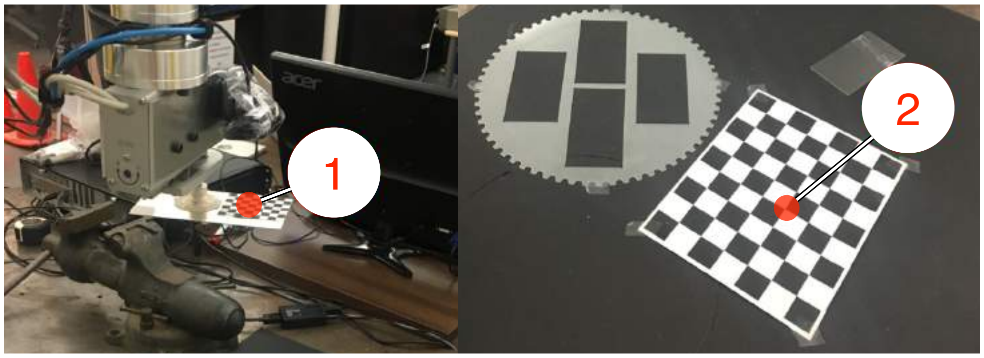


Figure 2.17: Two checkerboards for calibration

For each transformation T,

$$\begin{aligned} T &= \begin{bmatrix} R & t \\ \mathbf{0} & 1 \end{bmatrix} \\ R \in SO(3) &:= \{R | R^T R = RR^T = I, det(R) = 1\} \end{aligned} \tag{2.5}$$

in which R is the rotational part and t is the translational part. Thus rotation and translation can be estimated separately.

$$\begin{aligned} R_{6g} &= R_{6c} R_{cg} + noise, \\ t_{6g} &= R_{6c} t_{cg} + t_{6c} + noise \end{aligned} \tag{2.6}$$

For the estimation of rotation, the rotation matrix is first vectorized to X_R:

$$R = \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix}, X_R = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix} \tag{2.7}$$

The posterior probability is:

$$\begin{aligned} P(X_{R_{6c}}, X_{R_{cg}} | X_{R_{6g}}) &\propto P(X_{R_{6c}}, X_{R_{cg}}, X_{R_{6g}}) \\ &= P(X_{R_{6c}}) P(X_{R_{cg}}) P(X_{R_{6g}} | X_{R_{6c}}, X_{R_{cg}}) \end{aligned} \tag{2.8}$$

in which $P(X_{R_{6c}})$ and $P(X_{R_{cg}})$ are prior probabilities, $P(X_{R_{6g}} | X_{R_{6c}}, X_{R_{cg}})$ is likelihood. Assuming Gaussian noise, $P(X_{R_{6c}}) \sim \mathcal{N}(\mu_{R_{6c}}, \Sigma_{R_{6c}})$, $P(X_{R_{cg}}) \sim \mathcal{N}(\mu_{R_{cg}}, \Sigma_{R_{cg}})$ with $\mu_{R_{6c}}$ and $\mu_{R_{cg}}$ obtained from the previous section, $\Sigma_{R_{6c}}$and $\Sigma_{R_{cg}}$ obtained from the covariance of the data in the previous Section. $P(X_{R_{6g}} | X_{R_{6c}}, X_{R_{cg}}) \sim \mathcal{N}(\mu_{R_{6g}}, \Sigma_{R_{6g}})$ with $\mu_{R_{6g}} = X_{R_{6c}} X_{R_{cg}}$, and $\Sigma_{R_{6g}}$ tuned with cross validation.

For each observation, after taking the logarithm of the posterior probability, the log probability becomes:

$$\begin{aligned} l :&= logP(X_{R_{6c}}, X_{R_{cg}} | X_{R_{6g}}) \\ &= logP(X_{R_{6c}}) + logP(X_{R_{cg}}) + logP(X_{R_{6g}} | X_{R_{6c}}, X_{R_{cg}}) \end{aligned} \tag{2.9}$$

The MAP estimation of $X_{R_{6c}}$ and $X_{R_{cg}}$ are obtained by setting the partial derivatives to zero.

$$\frac{\partial l}{\partial X_{R_{6c}}} = \mathbf{0}, \quad \frac{\partial l}{\partial X_{R_{cg}}} = \mathbf{0} \tag{2.10}$$

where

$$\begin{aligned} \frac{\partial l}{\partial X_{R_{6c}}} =&(\mu_{R_{6c}} - X_{R_{6c}})^T \Sigma_{R_{6c}}^{-1} \\ &+(X_{R_{6g}} - \mu_{R_{6g}})^T \Sigma_{R_{6g}}^{-1} (I_3 \otimes R_{cg}^T), \\ \frac{\partial l}{\partial X_{R_{cg}}} =&(\mu_{R_{cg}} - X_{R_{cg}})^T \Sigma_{R_{cg}}^{-1} \\ &+(X_{R_{6g}} - \mu_{R_{6g}})^T \Sigma_{R_{6g}}^{-1} (R_{6c} \otimes I_3) \end{aligned} \tag{2.11}$$

Similarly, for the translation estimation, the observation model (Eq. 2.6) is transformed to:

$$t_{6g} = \underbrace{\begin{bmatrix} R_{6c} & I_3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} t_{cg} \\ t_{6c} \end{bmatrix}}_{X_t} + noise. \tag{2.12}$$

Still assuming Gaussian noise, $P(X_t) \sim \mathcal{N}(\mu_t, \Sigma_t)$, and $P(t_{cg}|X_t) \sim \mathcal{N}(AX_t, \Sigma_{t_{cg}})$. The translation estimation can be obtained by setting the partial derivative with respect to the log posterior probability to zero.

$$\begin{aligned} l :&= logP(X_t | t_{cg}) \\ &= logP(X_t) + logP(t_{cg} | X_t), \\ \frac{\partial l}{\partial X_t} &= \mathbf{0} \end{aligned} \tag{2.13}$$

For multiple observations, the log posterior probability becomes the sum of all data points.

$$l = \sum_i logP^i(T_{6c}, T_{cg} | T_{6g}^i) \tag{2.14}$$

Five observations are made (the glass plane is positioned at five known and distinct locations relative to the checkerboard) to update the prior probability. For cross validation, four data points are selected as the training set and the other one data point is the test set. $\Sigma_{R_{cg}}$ and $\Sigma_{t_{cg}}$ are tuned with cross validation. An optimal estimation is the one that minimizes the error of T_{bg} on all five test sets. The comparison of the prior estimation and

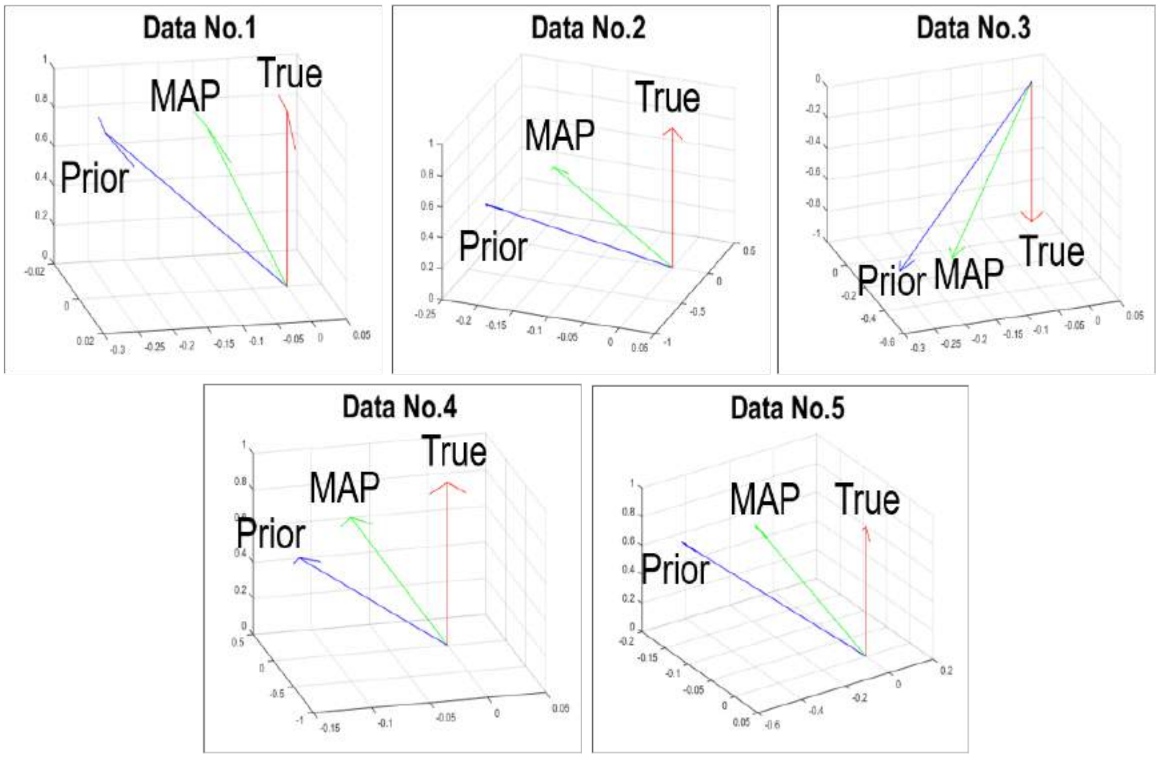


Figure 2.18: Rotation axis (R_{bg}) comparison of MAP and prior guess

the corrected estimation by MAP for T_{bg} is given in Fig. 2.18 and Fig. 2.19 for rotation and translation estimation respectively.

The accuracy of the calibration as well as the detection results are tested on the robot which is shown in Fig.2.20. Given the slot location from the Eye-to-Hand camera, the robot first calibrates the glass piece with respect to the robot's tool flange frame with the Eye-in-Hand camera, and then calculates the desired joint configuration with inverse kinematics. Fig. 2.20 shows the experiment used to test the result, and the desired glass position is at 5cm height from the slot. Fig. 2.21 shows the placing comparison between using prior calibration and calibration corrected by MAP. The resulting placing error of the prior calibration is about 10mm while for the calibration corrected with MAP is reduced to around 2 mm (based on 10 experiments). The error is further corrected by a visual servo technique using a dual Eye-in-Hand camera setting which finally guides the glass into the slot. The result will be illustrate in the following Sections.

Real-time TCP Calibration with Dual Eye-in-Hand Camera

For real-time TCP calibration with dual Eye-in-Hand camera, which is a different method from the MAP method, the transformation to be calibrated is the one from the camera frame

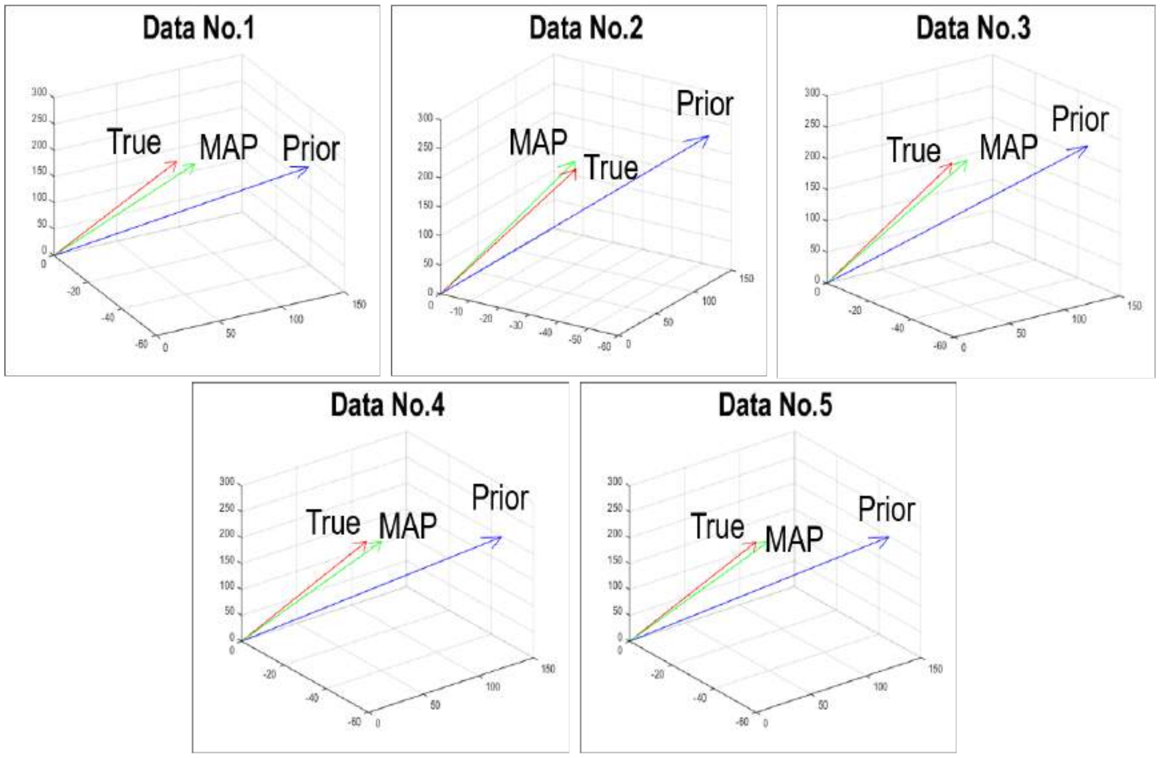


Figure 2.19: Transition (t_{bg}) comparison of MAP and prior guess

to the glass frame (Fig.2.22), a new T_{cg}. With pre-calibrated T_{6c} and T_{cg}, the placing work flow is the same as shown in Fig. 2.16.

After the robot picks up the glass piece, two Eye-in-Hand cameras detect four corners (each detects two) of the glass piece and the coordinates of the four corners in the image are obtained. Also, with the known size of the glass piece, four corners' coordinates in the glass frame are obtained. The mapping of a point from image frame to glass frame for each camera is determined by a projective transformation H_j ([30]):

$$\begin{aligned} H_j &= \begin{bmatrix} h_{j_{11}} & h_{j_{12}} & h_{j_{13}} \\ h_{j_{21}} & h_{j_{22}} & h_{j_{23}} \\ h_{j_{31}} & h_{j_{32}} & 1 \end{bmatrix} = \begin{bmatrix} h_{j_1} \\ h_{j_2} \\ h_{j_3} \end{bmatrix} \\ p_{j_i}^T &= p_{j_g}^T H_j, j = 1, 2 \end{aligned} \tag{2.15}$$

where p_{j_i} and p_{j_g} are points in the image frame and glass frame respectively, and each H_j ($j = 1, 2$) has eight parameters with j being the camera index. The goal is to solve for H_1 and thus the transformation from camera frame to glass frame T_1 can be obtained with

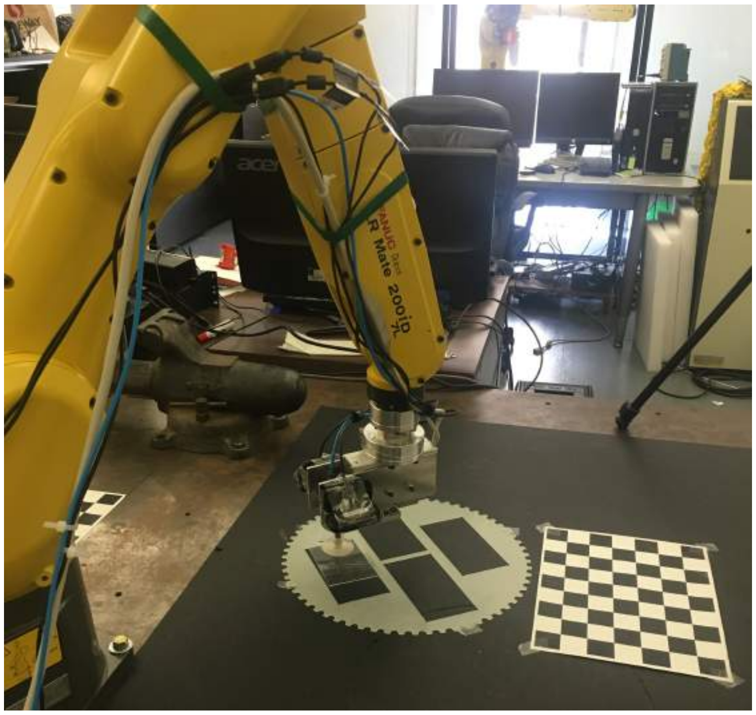


Figure 2.20: Experimental test

camera intrinsic matrix (K). Conversion from H_1 to T_1 is given in Eq.(2.16).

$$
\begin{aligned}
&K = \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix}, \ \lambda = \frac{1}{\|h_1 K^{-1}\|} \\
&r_1 = \lambda h_1 K^{-1}, r_2 = \lambda h_2 K^{-1}, r_3 = r_1 \times r_2 \\
&R = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}, t = \lambda h_3 K^{-1}, \ T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}
\end{aligned}
\tag{2.16}
$$

With coordinates of four pairs of points, eight mapping equations (each point has two coordinates) can be constructed for uniquely solving for H_1. H_2 is expressed by H_1 assuming the transformation from back camera frame to front camera frame is calibrated in order to maintain only eight parameters. However, with the relationship between H_2 and H_1 being non-linear, the equations are no longer linear and thus very hard to solve. In order to solve the equations in real-time, a good start point T_{1_0} is chosen to enable fast convergence. Because the glass will be always located at a roughly fixed position with respect to the Eye-in-Hand camera, a good start point can be determined easily with the size of the glass. The

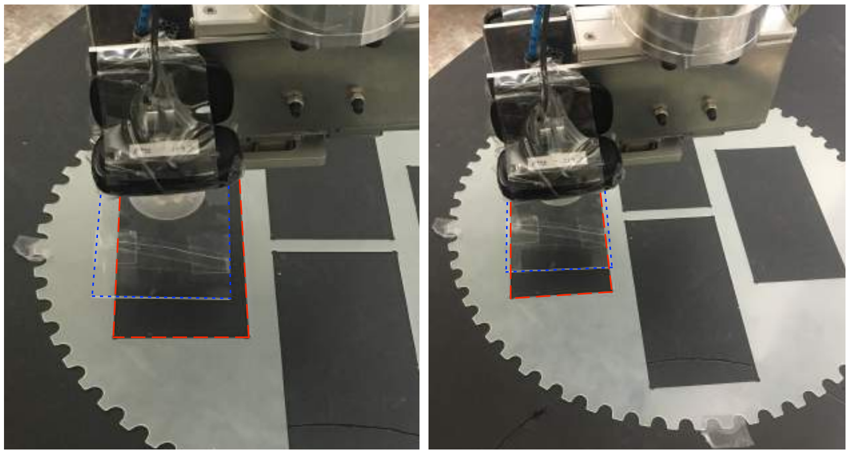


Figure 2.21: Experimental result

transformation T_1 is initialized as shown below:

$$T_{1_0} = \begin{bmatrix} 0 & 1 & 0 & -30(mm) \\ -1 & 0 & 0 & 100(mm) \\ 0 & 0 & 1 & 50(mm) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

A solution of T_1 can be obtained in less than a second, and one solving example is (depending on the points detected):

$$T_1 = \begin{bmatrix} -0.049 & 0.996 & -0.076 & -37.520(mm) \\ -0.983 & -0.035 & 0.182 & 131.214(mm) \\ 0.179 & 0.084 & 0.980 & 81.135(mm) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

Experimental Result of the Detection Accuracy

To verify the accuracy of the vision detection with Eye-to-Hand camera as well as the glass frame calibration, 10 experiments are performed. Each time a glass piece and the toothboard are placed randomly on the table and robot picks up the glass to place it into one slot. The glass piece and all slots were detected successfully for all 10 attempts. The pick up accuracy as well as the placing accuracy are given in Fig.2.23 and Fig.2.24. The high accuracy of the first step ensures the convergence of the second step, visual servo, since visual servo is a local method and it may diverge when the initial error is large.

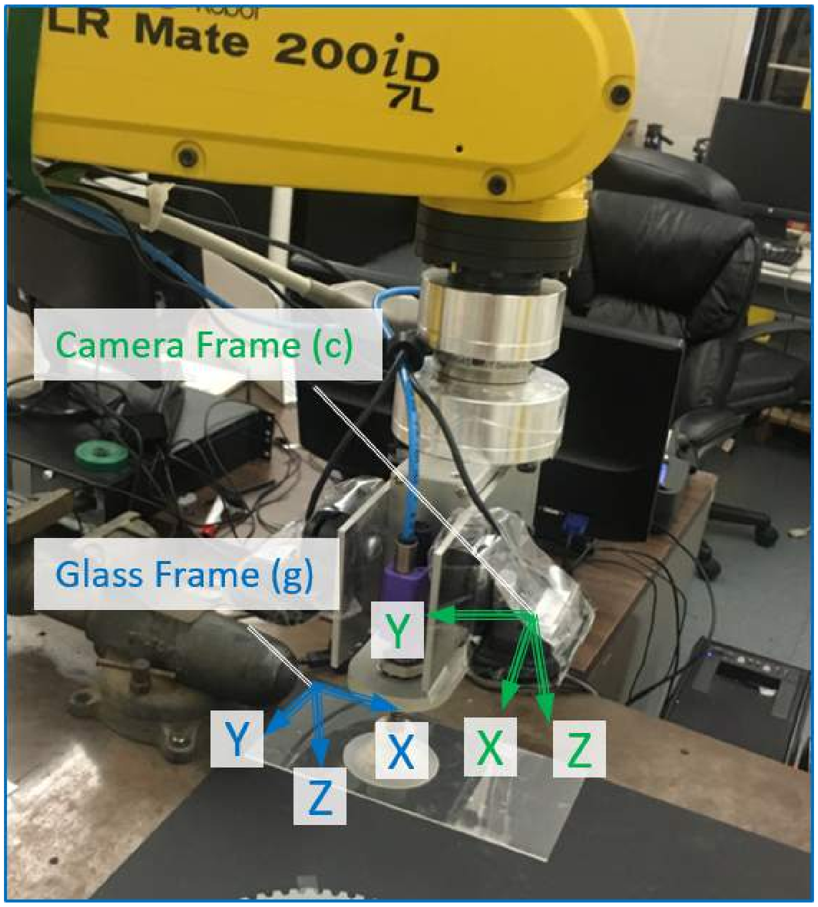


Figure 2.22: Glass frame and camera frame

2.4 State Estimation For Image Based Visual Servo

Task Description

After the robot moves to the target destination (slot position), glass will be above the slot at a distance. Then the placing step, which is the alignment of glass and slot, is performed with visual servo. In the placing task, the Eye-in-Hand camera sees both the slot and the glass. The goal of visual servo is to align the edges of the slot to the edges of the glass with the motion of the robot. In the ideal case, the alignment of three edges in both Eye-in-Hand cameras indicates the successful placing.

There are two basic visual servo types: position based and image based (Fig. 2.25 and Fig.2.26), where position based visual servo controls robot with the target position relative to the camera. Image based visual servo, on the other hand, takes the relative position implicitly from the image features. Position based visual servo considers the robot motion

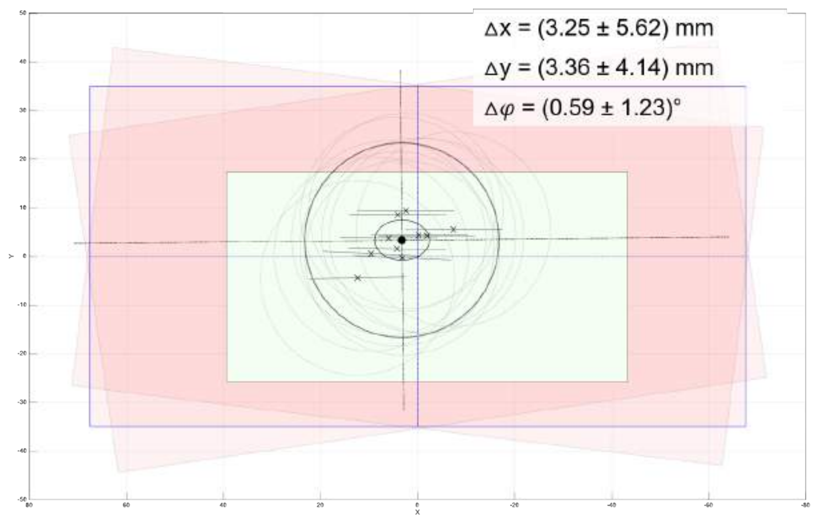


Figure 2.23: Glass pick up accuracy of the first step

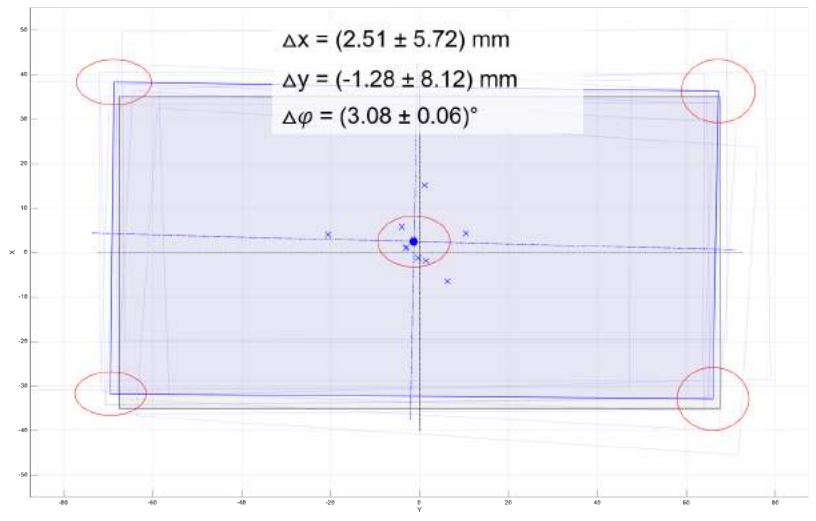


Figure 2.24: Glass placing accuracy of the first step

directly which avoid the motion of robot exceeding reach, yet feature points may leave the image. Image based visual servo controls the motion of robot directly in the image feature space. Although 3-D info of feature points are required, it is robust to the depth uncertainty.

During the placing task, because the glass is very close to the slot at the beginning, it can be assumed that the robot motion is small during the whole process. Also, it is crucial that feature points are kept in the image at all times for correct control input generation. Thus it is desirable to use image based visual servo, whose control structure is shown in Fig.2.26.

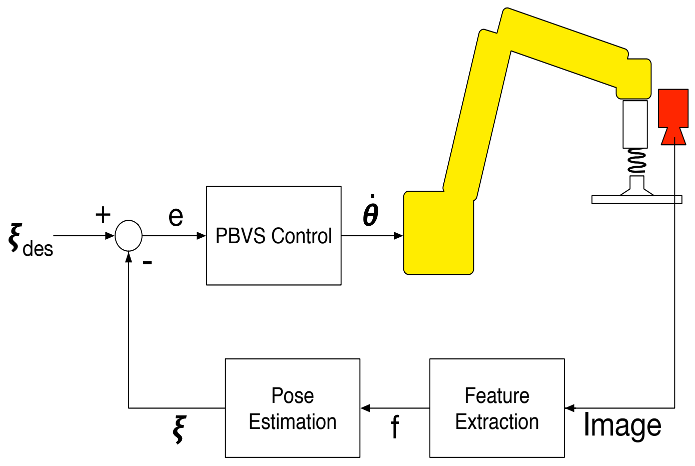


Figure 2.25: Position based visual servo frame

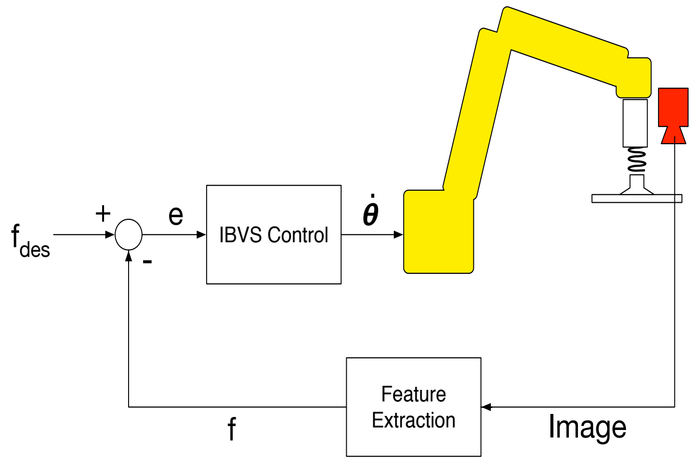


Figure 2.26: Image based visual servo frame

Several difficulties for the image based visual servo in this application need to be considered carefully. First, the image feature points feedback is very slow due to the hardware limitation (low cost webcam) and image processing. The sampling rate is about $2Hz$, which is very slow compared to the controller sampling frequency ($125Hz$) and robot motor sampling frequency ($1000Hz$). Not only the sampling rate is low, the delay is also huge. Because a high resolution image is required for both feature points detection as well as accurate placing, it takes a whole period for processing which leads to a one period delay. The delay is visible by human naked eyes, which results in robot slow motion or even diverging problem (feature points leave the image). In addition, although vision detection is robust, it may fail at some sample steps due to the sensitiveness to the environment change. Although vision can detect slot points at most of the time, a slight condition change or noise can result in detection failure. It is desirable that the visual servo is capable of false data rejection.

Another issue is the image based visual servo may suffer from low convergence rate when parameter uncertainty presents. Although it is robust to the depth estimation error, the performance will be sacrificed which leads to longer time convergence and lost feature points during the placing process. In this task, however, fast convergence is required since the table serves a hard constraints which forces the visual servo converges before hitting the table.

Image Jacobian

The result of edge detection with an Eye-in-Hand camera is given in Fig.2.27. Three edges of both the glass piece and the slot are detected as line features (e.g. two red lines indicate left edges of glass piece and slot respectively). Each line need to be transferred to a point first such that the controller can be designed using image Jacobian (image Jacobian is only for point features). The point is determined by the perpendicular foot from the image principle point (which could be intuitively viewed as the center point of the image) to the line. One example is illustrated in Fig.2.27. This approximation introduces a slight error to the controller which is negligible.

An image Jacobian based approach is adopted for controller design. The imgae Jacobian describes the relationship between camera motion and image feature points motion. Suppose a world point $\mathbf{P}$ with coordinates relative to camera is $\mathbf{P} = (X, Y, Z)$, camera velocity is $\mathbf{v} = (\boldsymbol{v}, \boldsymbol{\omega})$, and the normalized coordinates of $\mathbf{P}$ is $x = X/Z, y = Y/Z$. The motion of $\mathbf{P}$ relative to the camera is derived as:

$$\dot{\mathbf{P}} = -\boldsymbol{\omega} \times \mathbf{P} - \boldsymbol{v}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.19}$$

Considering the camera intrinsic parameters (intrinsic matrix), the image points (u, v) is related to the normalized coordinates as:

$$\begin{aligned} x &= \frac{u - u_0}{f_x} = \frac{\bar{u}}{f_x} \\ y &= \frac{v - v_0}{f_y} = \frac{\bar{v}}{f_y} \end{aligned} \tag{2.20}$$

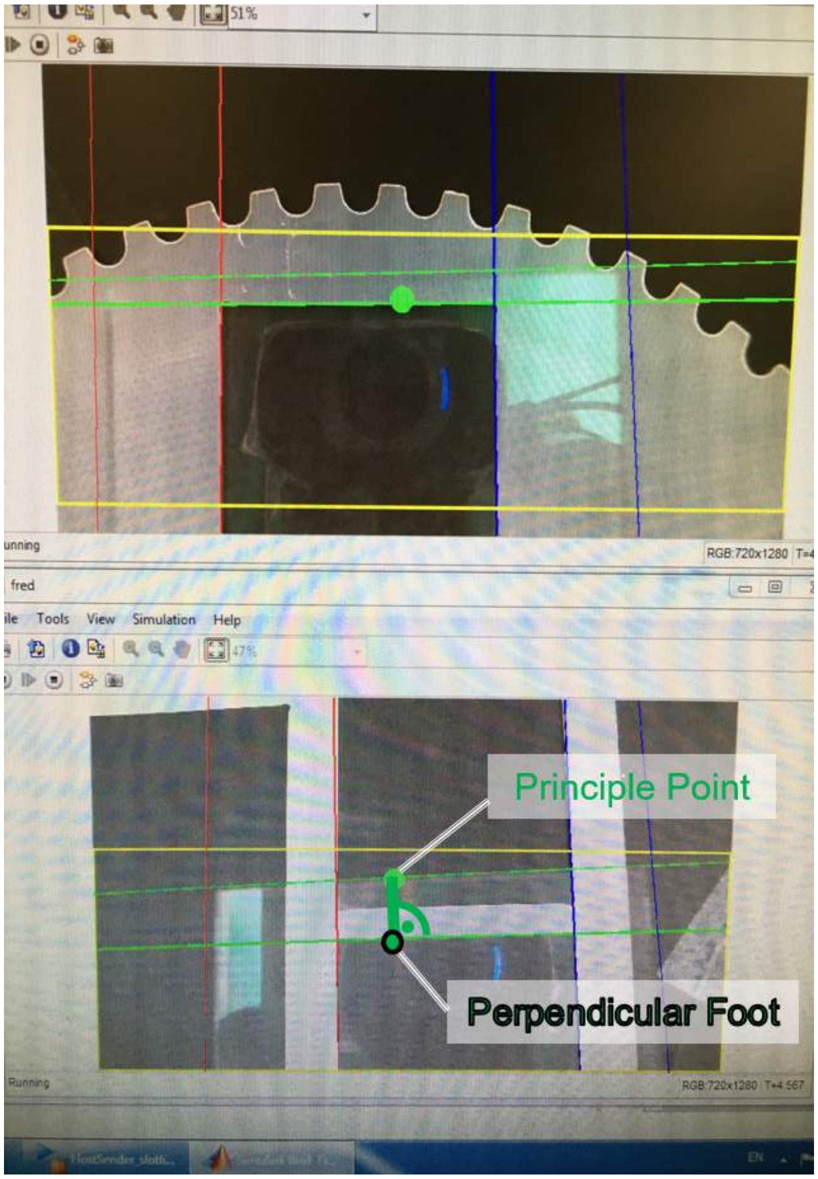


Figure 2.27: Line detection (back camera) and point approximation (slot's top edge)

where f_x and f_y are focal length for x and y directions. Substituting Eq. 2.20 into Eq. 2.19, the image Jacobian is obtained as:

$$\begin{bmatrix}\dot{\bar{u}}\\ \dot{\bar{v}}\end{bmatrix} = \underbrace{\begin{bmatrix}-\frac{f_x}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{f_y} & -\frac{f_x^2+\bar{u}^2}{f_x} & \frac{\bar{v}}{f_y}f_x\\ 0 & -\frac{f_y}{Z} & \frac{\bar{v}}{Z} & \frac{f_y^2+\bar{v}^2}{f_y} & -\frac{\bar{u}\bar{v}}{f_x} & -\frac{\bar{u}}{f_x}f_y\end{bmatrix}}_{J_p}\begin{bmatrix}v_x\\ v_y\\ v_z\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \tag{2.21}$$

For two Eye-in-Hand cameras, the camera velocity in Eq.(2.21) refers to each camera respectively. For control purposes, the back camera's velocity needs to be converted to the front camera velocity with the adjoint operator because two cameras are on the same rigid body:

$$\begin{aligned}\mathbf{v^b} &= Ad_{g^{bf}}\mathbf{v^f}\\ Ad_{g^{bf}} &= \begin{bmatrix}R_{bf} & t_{bf}\times R_{bf}\\ \mathbf{0} & R_{bf}\end{bmatrix}\\ T_{bf} &= \begin{bmatrix}R_{bf} & t_{bf}\\ \mathbf{0} & 1\end{bmatrix}\end{aligned} \tag{2.22}$$

where $\mathbf{v^b}$ and $\mathbf{v^f}$ are the back camera velocity and front camera velocity respectively. $Ad_{g^{bf}}$ is the adjoint operator. T_{bf} is the transformation from back camera to front camera. The image Jacobian for the back camera is then:

$$J_{p_2} = J_p Ad_{g^{bf}} \tag{2.23}$$

Visual Servo and Sensor Fusion

Because the detected edges are all expressed by points, the controller can be designed using image Jacobian. The control goal is the alignment of all three point pairs for both cameras. Let the error of the alignment be:

$$\begin{aligned} error_i = \mathbf{P_i} - \mathbf{P_i^*} = \begin{bmatrix}\bar{u}_{i1}-\bar{u}_{i1}^*\\ \bar{v}_{i1}-\bar{v}_{i1}^*\\ \bar{u}_{i2}-\bar{u}_{i2}^*\\ \bar{v}_{i2}-\bar{v}_{i2}^*\\ \bar{u}_{i3}-\bar{u}_{i3}^*\\ \bar{v}_{i3}-\bar{v}_{i3}^*\end{bmatrix}\\ i = 1, 2\end{aligned} \tag{2.24}$$

where $\mathbf{P_i}$ denotes the three image points of the slot and $\mathbf{P_i^*}$ represents the three image points of the glass for the ith camera. A linear controller can be obtained easily if the error is designed to decay to zero exponentially

$$\begin{aligned}\dot{\mathbf{P}}_\mathbf{i} &= \lambda(\mathbf{P_i^*} - \mathbf{P_i})\\ i &= 1, 2\end{aligned} \tag{2.25}$$

With Eq.(2.21), the velocity of each cameras is computed with:

$$\mathbf{v_i} = \begin{bmatrix} Jp_{i1} \\ Jp_{i2} \\ Jp_{i3} \end{bmatrix}^{-1} \begin{bmatrix} \dot{u}_{i1} \\ \dot{v}_{i1} \\ \dot{u}_{i2} \\ \dot{v}_{i2} \\ \dot{u}_{i3} \\ \dot{v}_{i3} \end{bmatrix} = \lambda \begin{bmatrix} Jp_{i1} \\ Jp_{i2} \\ Jp_{i3} \end{bmatrix}^{-1} (\mathbf{P_i^*} - \mathbf{P_i}) \qquad i = 1, 2 \tag{2.26}$$

where Jp_{ij} $(i = 1, 2, j = 1, 2, 3)$ represents the image Jacobian for the ith camera and jth point. It should be noted that for a unique solution of the camera velocity, at least three image points are required such that the image Jacobian is of full rank. Thus only line features can provide sufficient information of the scene. Then the desired camera velocity is combined as a weighted summation of two camera velocities.

$$\mathbf{v} = \eta\mathbf{v_1} + (1 - \eta)\mathbf{v_2}, \tag{2.27}$$

where η is determined by the normalized error of two cameras:

$$\eta = \frac{err_1}{err_1 + err_2} \tag{2.28}$$

where err_1 and err_1 are normalized error with the initial error:

$$\begin{aligned} err_1 &= error_1/error_1(0) \\ err_2 &= error_2/error_2(0) \end{aligned} \tag{2.29}$$

The camera frame is chosen as the robot end-effector, such that the camera velocity is directly related to the robot joint velocity by the robot Jacobian J_r, and the control input $\dot{\mathbf{q}}$ is subsequently calculated:

$$\begin{aligned} \mathbf{v} &= J_r(\mathbf{q})\dot{\mathbf{q}} \\ \dot{\mathbf{q}} &= J_r(\mathbf{q})^{-1}\mathbf{v} \end{aligned} \tag{2.30}$$

Dual-rate UKF State Estimation

The naive visual servo framework can only be executed at a very low speed due to the low frame rate and large latency of the camera. To compensate for such issue, an UKF is proposed to estimate the system states, which are the coordinates of the image points of the slot, in real-time. Due to the frequency mismatch between the sensor and the controller, the UKF is designed at dual-rate, i.e., a high sampling rate of the controller $(125Hz)$ and a low sampling rate of the sensor (around $2Hz$). The low rate sampling time is set to be a constant multiplication of the high rate sampling time $(\delta T_l = L\delta T_h, L = 66)$, and the latency is set as a whole period (latency $= T_l$).

Figure 2.28 illustrates the dual-rate filtering. When data is available at a low sampling rate (every L step), a correction action is performed to correct the past states estimation followed by a L-step prediction action to estimate the current states. Then, single step predictions are performed at high sampling rate during the gap, before the next sensing data is available. The low sampling rate model is:

$$\begin{aligned} x_{k+1} &= x_k + \delta T_l J_{p_i} J_r \dot{\mathbf{q}}_k + w_k = f(x_k) + w_k, i = 1, 2 \\ y_k &= x_k + v_k = h(x_k) + v_k \end{aligned} \tag{2.31}$$

where x represents the coordinates of the image points of the slot, J_{p_i} is the image Jacobian defined in Eq.(2.21), and J_r is the robot Jacobian given in Eq.(2.30). The robot joint angular velocity is $\dot{\mathbf{q}}_k$, which is the control input to this model. δT_l is the low rate sampling time, w_k is modeled as an additive process noise with covariance Q, y_k is the measurement from the vision detection, and v_k is modeled as an additive measurement noise with covariance R.

The high sampling rate model shares the similar structure of the low rate model:

$$\begin{aligned} x_{kL+m+1} &= x_{kL+m} + \delta T_h J_{p_i} J_r \dot{\mathbf{q}}_{kL+m} + w_{kL+m}, i = 1, 2 \\ m &= 0, 1, \ldots, L-1 \end{aligned} \tag{2.32}$$

It should be noted that k is the low rate index.

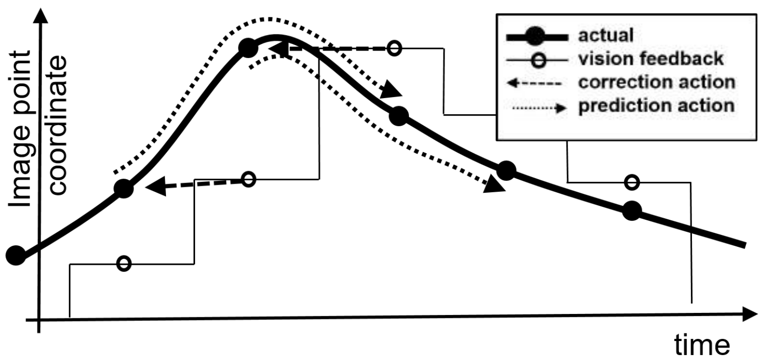


Figure 2.28: Dual-rate UKF for low sampling rate and large latency compensation

At every L steps, when sensing data is available, the states estimation $\hat{x}_{k|k-1}$ and covariance $P_{k|k-1}$ from the previous low sampling rate step are given, and $\hat{x}_{k|k}$ and $P_{k|k}$ are updated with the correction step. $\hat{x}_{k+1|k}$ and $P_{k+1|k}$ are updated with the prediction step.

During the correction step, states $\hat{x}_{k|k-1}$ and $P_{k|k-1}$ are first augmented with the measurement noise:

$$\begin{aligned} x^a_{k|k-1} &= \begin{bmatrix} \hat{x}^T_{k|k-1} & E[v^T_k] \end{bmatrix}^T \\ P^a_{k|k-1} &= \begin{bmatrix} P_{k|k-1} & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix} \end{aligned} \tag{2.33}$$

Then $2M+1$ sigma points and weights (for states and covariance) are generated where M is the dimension of the augmented states:

$$\begin{aligned}\chi^0_{k|k-1} &= x^a_{k|k-1}\\ \chi^i_{k|k-1} &= x^a_{k|k-1} + (\sqrt{(M+\lambda)P^a_{k|k-1}})_i, i = 1,\ldots,M\\ \chi^i_{k|k-1} &= x^a_{k|k-1} - (\sqrt{(M+\lambda)P^a_{k|k-1}})_{i-M}, i = M+1,\ldots,2M\\ \lambda &= \alpha^2(M+\kappa) - M\\ W^0_s &= \frac{\lambda}{M+\lambda}\\ W^0_c &= \frac{\lambda}{M+\lambda} + (1-\alpha^2+\beta)\\ W^i_s = W^i_c &= \frac{1}{2(M+\lambda)}, i = 1,\ldots,2M\end{aligned} \tag{2.34}$$

α, κ, β are parameters that can be tuned. The sigma points are projected through the observation function h given in Eq.(2.31):

$$\gamma^i_k = h(\chi^i_{k|k-1}), i = 0,\ldots,2M \tag{2.35}$$

The predicted measurement, predicted measurement covariance and state-measurement cross covariance are:

$$\begin{aligned}\hat{y}_k &= \sum_{i=0}^{2M} W^i_s \gamma^i_k\\ P_{y_k y_k} &= \sum_{i=0}^{2M} W^i_c[\gamma^i_k - \hat{y}_k][\gamma^i_k - \hat{y}_k]^T\\ P_{x_k y_k} &= \sum_{i=0}^{2M} W^i_c[\chi^i_{k|k-1} - \hat{x}_{k|k-1}][\gamma^i_k - \hat{y}_k]^T\end{aligned} \tag{2.36}$$

The UKF kalman gain is:

$$K_k = P_{x_k y_k} P^{-1}_{y_k y_k} \tag{2.37}$$

The corrected $\hat{x}_{k|k-1}$ and $P_{k|k}$ are updated as:

$$\begin{aligned}\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_k)\\ P_{k|k} &= P_{k|k-1} - K_k P_{y_k y_k} K^T_k\end{aligned} \tag{2.38}$$

During the prediction step, $2M+1$ sigma points are generated with $\hat{x}_{k|k}$ and $P_{k|k}$ with process noise covariance Q similar to the correction step. Then sigma points are propagated through the transition function f as given in Eq.(2.31).

$$\chi^i_{k+1|k} = f(\chi^i_{k|k}), i = 0, \ldots, 2M \tag{2.39}$$

As the weights are the same as in the correction step, the updated prediction $\hat{x}_{k+1|k}$ and $P_{k+1|k}$ are:

$$\begin{aligned}\hat{x}_{k+1|k} &= \sum_{i=0}^{2M} W^i_s \chi^i_{k+1|k} \\ P_{k+1|k} &= \sum_{i=0}^{2M} W^i_c [\chi^i_{k+1|k} - \hat{x}_{k+1|k}][\chi^i_{k+1|k} - \hat{x}_{k+1|k}]^T\end{aligned} \tag{2.40}$$

When sensing data is not available, a high sampling rate single-step prediction is performed at every step with the high sampling rate model as shown in Eq.(2.32). Given $\hat{x}_{kL+m}$ and P_{kL+m} with $m = 0, 1, \ldots, L-1$, $\hat{x}_{kL+m+1}$ and P_{kL+m+1} are updated with the high-rate model given by Eq.(2.32). It should be noted that when $m = 0$, $\hat{x}_{kL}$ and P_{kL} should be $\hat{x}_{k|k-1}$ and $P_{k|k-1}$ updated from the low sampling rate UKF.

UKF with Dual-estimation

With UKF compensation, although much faster performance is achieved shown in Section 2.5, it is still too slow to meet the requirement. This is due to the fact that the visual servo suffers from slow convergence under system parameter uncertainty. To further improve the placing performance, an UKF with dual-estimation is proposed to estimate states and unknown system parameters simultaneously. At every L steps, two UKFs are running in parallel on two models (system states and parameters) to update system parameters and the states simultaneously.

The system parameters are chosen as the displacement vector from the back Eye-in-Hand camera to the front Eye-in-Hand camera t_{bf} in Eq.(2.22). The model for state estimation is given in Eq.(2.31) with parameter t_{bf} included to the back camera:

$$\begin{aligned}x_{k+1} &= x_k + \delta T_l J_{p_2}(t_k) J_r \dot{\mathbf{q}}_k + w_k \\ y_k &= x_k + v_k\end{aligned} \tag{2.41}$$

where t represents t_{bf}, and a separate model for unknown parameter t is:

$$\begin{aligned}t_k &= t_{k-1} + u_{k-1} \\ y_k &= x_{k-1} + \delta T_l J_{p_2}(t_k) J_r \dot{\mathbf{q}}_{k-1} + v_k + n_k\end{aligned} \tag{2.42}$$

At each time step k (low sampling rate index), when measurement y_k is available, t_k is updated with y_k, $\hat{x}_{k-1|k-1}$ and t_{k-1}. After t_k is updated, $\hat{x}_{k|k}$ and $\hat{x}_{k|k+1}$ are updated with t_k. The parameter adaptation result is shown in Fig.2.29, in which parameter is initialized as a nominal value from a pre-calibration, shown in blue dotted line. Red lines show the adaptation of actual parameters. Fig.2.29 implies that visual servo is very sensitive to parameter uncertainty whose range is as small as 2mm.

The UKF filtering result is given in Fig.2.30 for each camera and each line. It should be noted that for each line, filtering is only good for one coordinate with larger range. This is due to the point approximation error explained in “Image Jacobian” in this Section. However, such error does not compromise the placing performance since the filtering error is very small that is negligible.

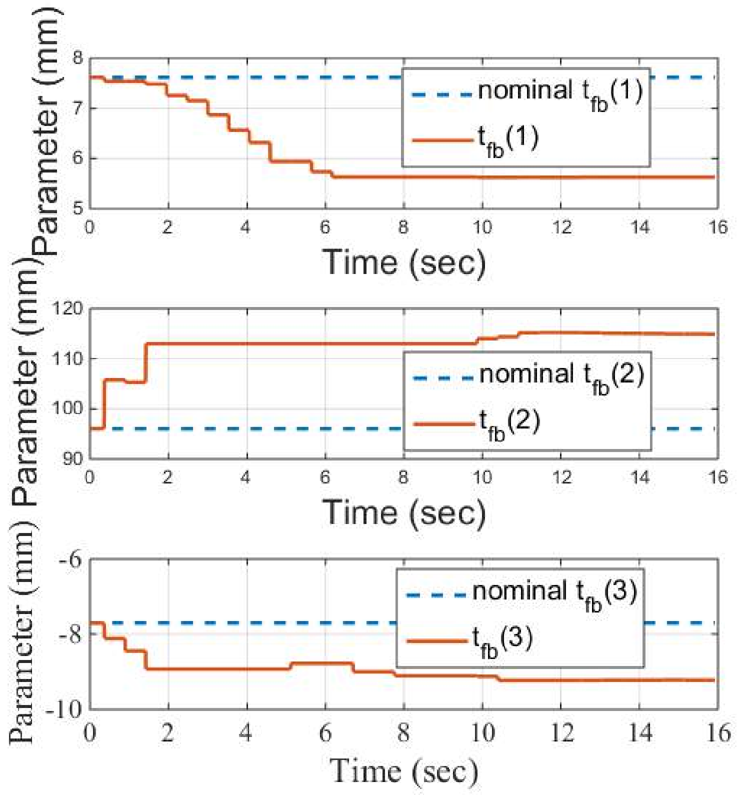


Figure 2.29: Parameter adaptation with UKF dual-est

2.5 Result

In the experiment, in fact, failure of detection occurs because the vision sensor is sensitive to the disturbance or the subtle change of the environment, such as lighting condition. Thus a filter is first adopted to reject any false detection, shown in Fig.2.31. Blue lines are the raw data from cameras, and there are multiple jumps (discontinuities implying the failures of detection) in the raw data. The data after failure rejection is shown by red lines, which

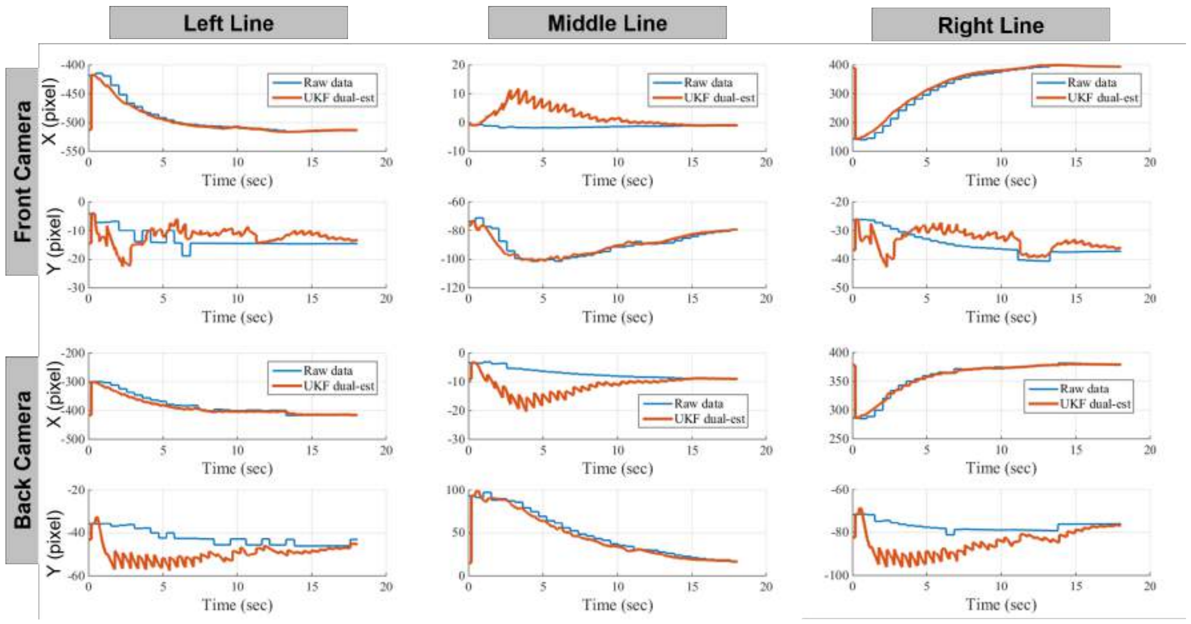


Figure 2.30: UKF with dual-estimation Filtering

removes jumps in the raw data and gives a more continuous data that can be used by controller.

The raw data (after failure rejection) and the data with UKF filtering are given in Fig.2.30. Two rows of plots show the results for front and back Eye-in-Hand cameras respectively. Three columns of plots are for the left, middle and right edges of the slot during placing respectively. Each plot shows the coordinates in x and y axes of the line (represented by a point) in the image. Blue lines are the raw data after failure rejection and red lines are the data after filtering. It should be noted that for each line, filtering is only good for one coordinate with larger range. This is due to the point approximation error explained in “Image Jacobian” in this Section. However, such error does not compromise the placing performance since the filtering error is very small that is negligible.

Figure 2.32 shows the error convergence for a successful placing when using raw data, UKF only and UKF with dual-estimation. Rows and columns are still for two cameras and three edges of the slot respectively. The error is defined in Sec.2.4, which is the difference between slot edges and glass edges. Blue solid lines are the errors of using only raw data, which takes over 100 seconds for convergence. Red dotted lines are the errors of using UKF only, and the convergence time is reduced to about 40 seconds. Yellow dashed lines are the errors of using UKF with dual estimatin, which only takes less than 20 seconds for convergence. The criteria for a successful placing is the coordinate error less than 20 pixels. Figure 2.32 shows that the UKF with dual-estimation has superior performance than the other two methods. Comparison of three methods are also given in Tab.2.1, which shows

Table 2.1: Visual servo comparison of using raw data, UKF and UKF w/ dual-estimation

	Raw data	UKF only	UKF w/ dual-est.
Success rate[1]	90%	90%	100%
Conv time[2]	102.7s	44.6s	14.6s

[1] Based on 10 experiments with randomly placed glass piece.
[2] Average convergence time

the success rate and average convergence time based on 10 successive experiments with glass piece randomly placed on the surface. A video is also available for the robot fast glass placing demonstration [31].

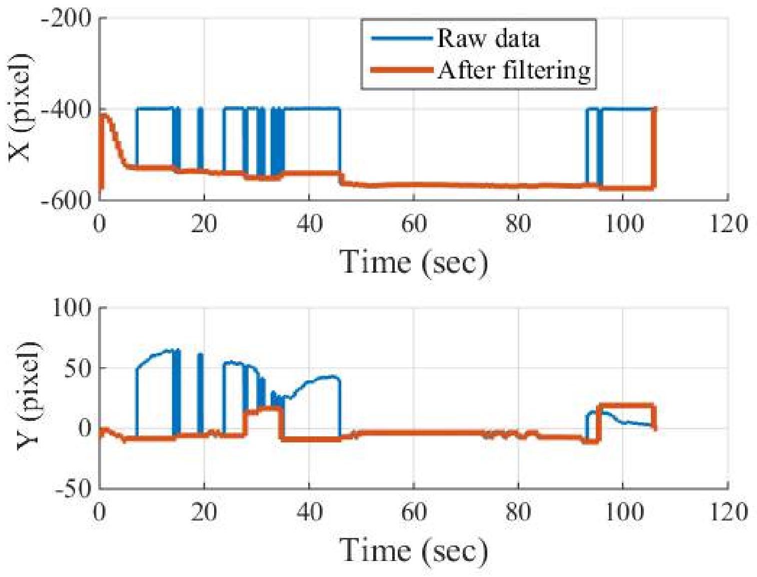


Figure 2.31: Failure of detection and after filtering

2.6 Chapter Summery

This chapter presented a series of statistical learning approaches to overcome the difficulties for fast, accurate and robust visual servo. The work is intended to help an industrial manipulator for glass piece handling using real-time visual feedback. Due to the hardware limitations, challenges come from several aspects, namely, object detection, TCP calibration,

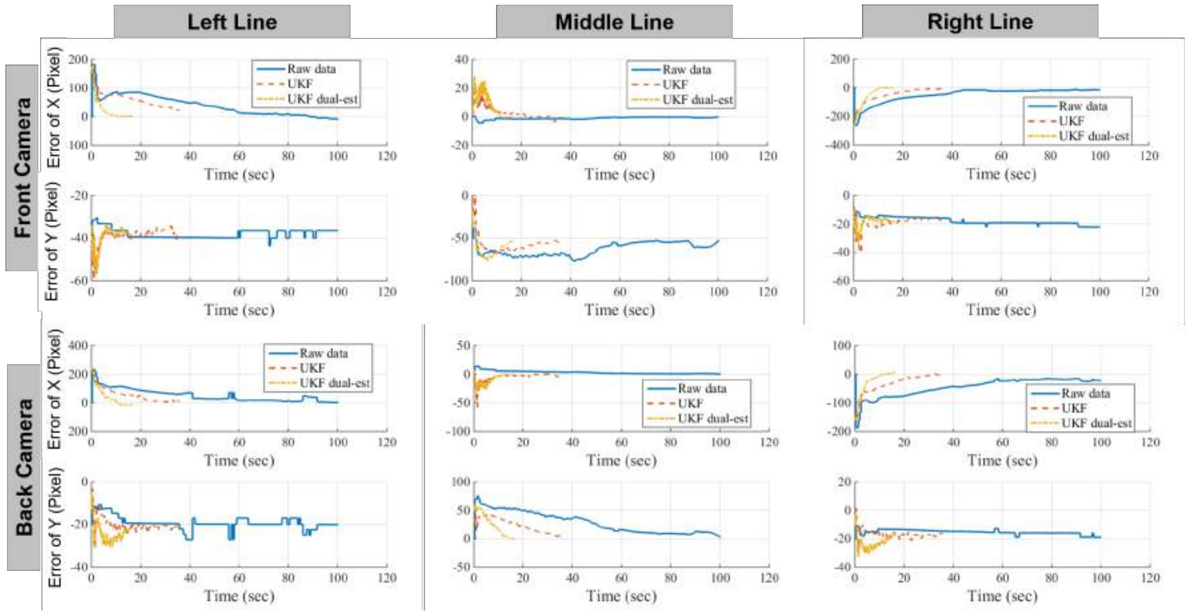


Figure 2.32: Error convergence comparison of using raw-data, UKF and UKF with dual-estimation

visual feedback with large latency and low sampling rate, and system uncertainties. To solve such issues, a two step approach is proposed to achieve high placing performance. The first step consists of picking up the glass piece with the robot and moving it above the slot. The second step focuses on guiding the glass into the slot with real-time visual feedback.

In the first step, glass and slots are both detected robustly with an Eye-to-Hand camera by a non-linear geometrical regression. Accurate detection ensures that the visual servo, which is the second step, initiated with a small error. A closed loop controller is implemented to guide the glass piece into the slot. Instead of using point features of the object, line features are adopted for higher placing precision. To compensate for the cameras' very low frame rate and parameter uncertainties, a dual-rate UKF with dual-estimation is implemented. Experimental results show the effectiveness of the placing performance.

In the experiment, a 6 DOF Fanuc robot arm is used, however, the method proposed in this Chapter can be widely applied to other low cost and less accurate robot manipulator for more cost-effective system. Since the visual servo based closed loop control can achieve a high precision on the robot encoder resolution rather than the robot's absolute accuracy.

Chapter 3

Optimal Trajectory Planning

3.1 Introduction

Automation of trajectory planning for industrial manipulators working in constrained environment is an important topic. Moving all the joints of a robot in a coordinate manner is particularly challenging since the motion have to respect the kinematic constraints of the manipulator, avoid collisions with objects in presence, and maintain time optimality. Due to the non-linearity and complexity of the robot kinetics as well as the constraints, it is impossible for manual planning. Currently, for industrial applications, a "teach point" method, which is to use teach pendant to manually record the via points along a path, is still in use. Such method is labor intensive and time consuming which has to be replaced by automatic motion planning.

Constrained trajectory planning for robotic manipulators [32, 33, 34, 35] has been studied in great detail over the last several decades. One approach is the sampling based planning. The idea of sampling based planning is to avoid explicit construction of robot configuration space ($\mathcal{C}$ space) and instead to search through the $\mathcal{C}$ space with a sampling scheme. In sampling-based planning, collision detection is viewed as a black box which seperates the searching from the explicit robot geometric models. For example, Expansive Space Trees and Probabilistic Roadmap (PRM) methods [36, 37, 38] are suited for high dimensional motion planning in robot $\mathcal{C}$ space, with kinodynamic constraints. In fact, PRM method works well in static environments and for multi-query problems. Rapidly-exploring Random Trees (RRT) [39] was originally developed for motion planning under differential constraints, and were shown to be successful for general high-dimensional planning. However, sampling-based methods suffer from the fact that it is hard to take into account the path smoothness and non-linear differential constraints, which will result in jerky and unnatural motion. Also, there is no guarantee on optimality or completeness.

Another approach is the optimization based motion planning, which tries to optimize over an objective function formulated as a combination of smoothness, obstacle collision or/and other performance criteria of the trajectory. In the potential filed method [40], potential energy is composed of two parts: attractive potential and repulsive potential. The agent gets less potential when near the goal or far away from the obstacle. In the elastic band method [41], a trajectory is assigned an internal energy related to its length or smoothness, along with an external energy generated by obstacles or taskbased potentials. Other gradient based approaches, such as CHOMP [42] and STOMP [43], share much in common with elastic bands planning, except that the covariant gradient descent method is used, and obstacles are considered directly in the workspace of the robot, where the notions of distance and inner product are more natural. These methods, however, usually have heavy computational loads or work only for a predetermined duration.

One particularly difficult example for constrained trajectory planning of robotic manipulator is for a wafer handling robot. A silicon wafer handling robot works inside the equipment front end module (EFEM) of a semiconductor manufacturing machine (Fig.3.1). In Fig.3.1, an EFEM is the interface of a semiconductor manufacturing machine to the factory. One side of the EFEM connects to the factory through loadports (the portal to the outside environment of the semiconductor manufacturing machine, i.e, the atmospheric environment). The other side of the EFEM connects to the wafer processing chambers through loadlocks (the portal to the vacuum environment of the semiconductor manufacturing machine). Wafers are picked up by the wafer handling robot from loadports and then transferred to loadlocks. An vacuum robot picks up wafers from loadlocks and transfers them to vacuum chambers for wafer processing.

Usually, the EFEM has a near-rectangular internal space. The space inside an EFEM is extremely limited relative to the size of the atmospheric wafer transfer robot. In addition, there are usually multiple loadports and loadlocks. The goal is to automate the trajectory planning, which means that with given loadport and loadlock positions, the trajectory planner can generate an "optimal" smooth trajectory that the robot can follow without violating any working constraints. The working constraints are defined in robot work space as well as in joint space. The first constraint is the obstacle constraint which means that a robot should not collide with obstacles (which is EFEM in this case). The second constraint is the end-effector constraint which requires that the initial and final positions and orientations of the end-effector are favourable for wafer-delivery (pick up). In addition, in order to avoid the sliding of wafers, which may result in particle contamination and even wafer tip-over, the acceleration of the wafer should be limited. There are also limits on joint velocity and acceleration, which are determined by the capabilities of the motors. A trajectory that satisfies all constraints mentioned above should be further optimized for shorter cycle time, which is crucial for semiconductor manufacturing throughput.

The motion planning problem for the wafer handling robot in this Chapter can be categorized as a multi-query planning problem because of the multiple loadlock and loadport positions. In a single query planning problem, on the other hand, a robot manipulator only goes to a single desired goal position with a given initial position. It is thus desirable to extensively investigate the working environment prior to search for one particular path. PRM method is suitable in this case since it can rapidly construct a roadmap of the environment.

For trajectory smoothing, one commonly used smoothing method uses shortcutting heuris-

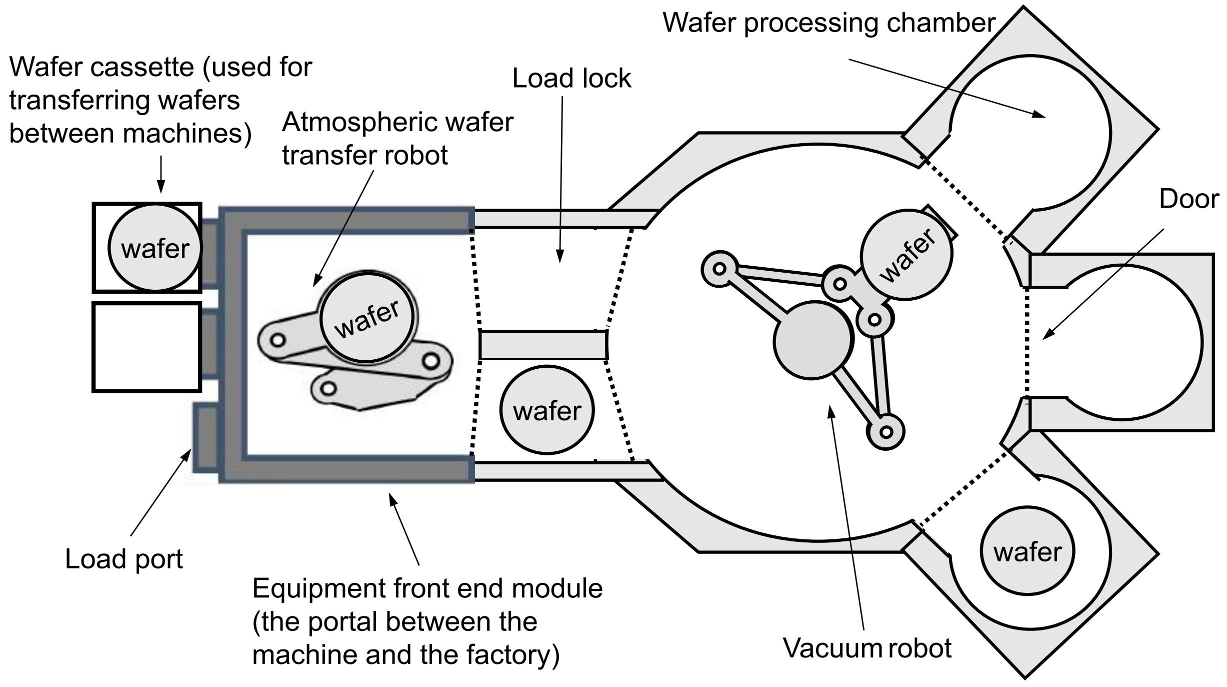

Wafer processing chamber
Wafer cassette (used for transferring wafers between machines)
Load lock
Atmospheric wafer transfer robot
Door
wafer
wafer
wafer
wafer
wafer
Load port
Equipment front end module (the portal between the machine and the factory)
Vacuum robot

Figure 3.1: A robot manipulator works inside an EFEM of a semiconductor manufacturing machine

tic, which is to replace a subpath by a shortcut segment when it is collision free [44]. This method, however, cannot generate trajectories with higher order smoothness and cannot guarantee any optimality. Spline curve [45, 46] seems to be a suitable choice since they are able to restrict the trajectory in a safe region and in the mean time parameterize the trajectory with temporal information. For example, Bezier curve can generate curves bounded by control points. However, the order of the curve grows when the number of the control points increases . In our case, for example, there are 12 control points, and the order of the curve will be 11, which is not realistic. Moreover, bezier curve only works for a single piece of curve. On the other hand, B-spline function [47, 48] is ideal for generating a curve which is composed of several segments since continuity condition is automatically satisfied. Also, it is able to restrict the curve in a safe region and in the mean time keeps the order of the curve constant.

However, direct B-spline fitting always violates the original path and lead to collision. Gasparetto and Zanotto [49] use B-spline function to generate and optimize a trajectory over a set of via-points. Although all the kinematics constraints are satisfied, it cannot guarantee a safe interpolation when obstacles are present.

In this Chapter, a cubic B-spline function is used to fit the path. By carefully locating the control points, it is guaranteed that the fitting is collision free. A constrained optimization problem is formulated with the objective function as the total execution time and constraints as kinematic constraints. The B-spline curve parametrization makes the optimization problem extremely simplified and a good initial point can be determined by a standard routine. On-line implementation is possible with this approach which is desirable in industrial applications.

The rest of the Chapter is organized as follows: in Section 3.2, the kinematic model of the robot and a model for the obstacle are described. The path finding problem is defined. In Section 3.3, an adaptive PRM planner with three types of samplers is introduced to generate the roadmap with the given obstacles. With a query as an input to the PRM planner, it returns a piecewise linear safe path which is in fact a shortest path in the roadmap. Section 3.4 gives the whole algorithm of transferring the piecewise linear path into a smooth trajectory that the robot can follow while satisfying all the constraints. Section 3.5 shows by simulation the effectiveness of the algorithm given in Section 3.4.

3.2 Configuration of The Atmospheric Wafer Transfer Robot and The Obstacle

The atmospheric wafer transfer robot is a 3-axis robot with three revolute joints. The robot is a planner robot whose motion is restricted in the x-y plane, as shown in Fig.3.2

The robot works in an EFEM, which gives an extremely limited work space. Because the robot is a planer one, the EFEM is modeled as a closed boundary around the robot. The top view of the boundary, loadports and loadlocks, as well as the robot are shown in Fig.3.3. There are multiple positions for loadports and loadlocks. The robot needs to pick up wafers from loadports and unload wafers to loadlocks. In this Chapter, however, only the motion inside the EFEM is considered. The beginning of the motion is right after the robot picks up the wafer from the loadport and retracts back to EFEM. The end of the motion is before the robot extends to unload wafer to the loadlock, as shown in Fig.3.4. The goal of trajectory planning is that given the positions of a loadport and a loadlock (and immediately the corresponding configurations inside the EFEM), generate a safe trajectory which satisfies all constraints.

The constraints include:

1. the trajectory should be defined in robot joint space.
2. robot should not collide with obstacles (which is the boundary).
3. constraints on the maximum velocity and acceleration of the joints should be satisfied.
4. the wafer-center (end-effector) acceleration amplitude should be limited within a threshold to prevent wafer sliding.

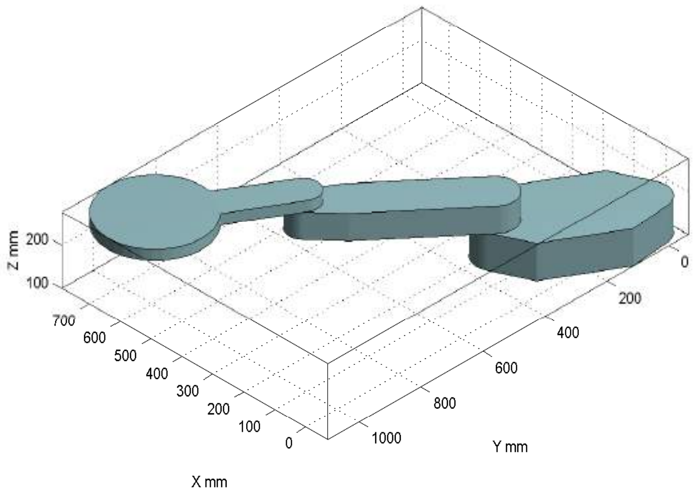


Figure 3.2: Model of the 3-axis manipulator works in an EFEM

In addition to the constraints, the cycle time should be as short as possible because productivity is a very sensitive issue in industrial applications. The total time duration of the trajectory should be minimized without violating any constraints.

3.3 PRM Planning

A probabilistic roadmap (PRM) planner is formulated for the multi-query problem. “The basic idea behind PRM is to take random samples from the $\mathcal{C}$ space of the robot, testing them for whether they are in the free space, and use a local planner to attempt to connect these configurations to other nearby configurations. The starting and goal configurations are added in, and a graph search algorithm is applied to the resulting graph to determine a path between the starting and goal configurations” (*Wikipedia*). In this Section, PRM tries to construct a roadmap that covers the obstacle-free space as much as possible. Because of the unknown property of the obstacle-free space, an adaptive sampling method is adopted to adjust samplers. The detailed formulation of this method can be found in [36].

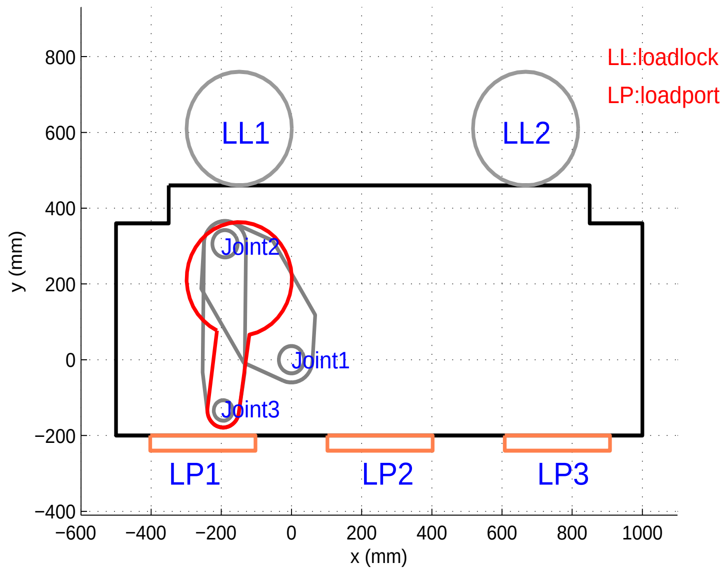
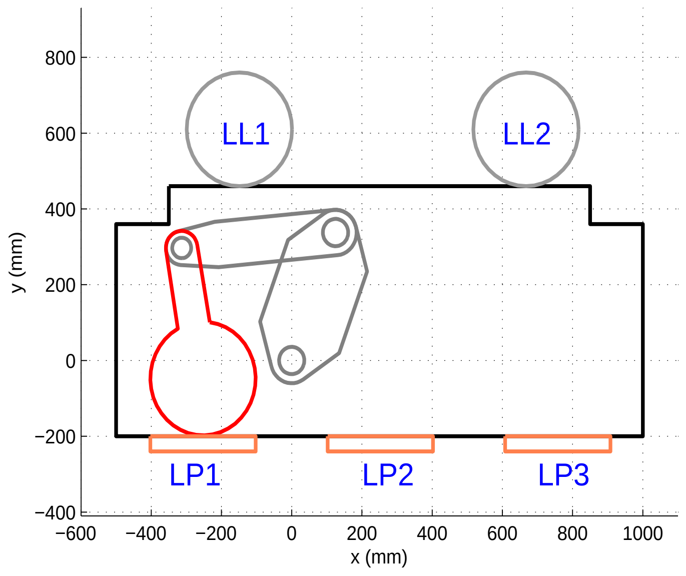


Figure 3.3: The Equipment Front End Module (EFEM) is modeled as a closed boundary obstacle

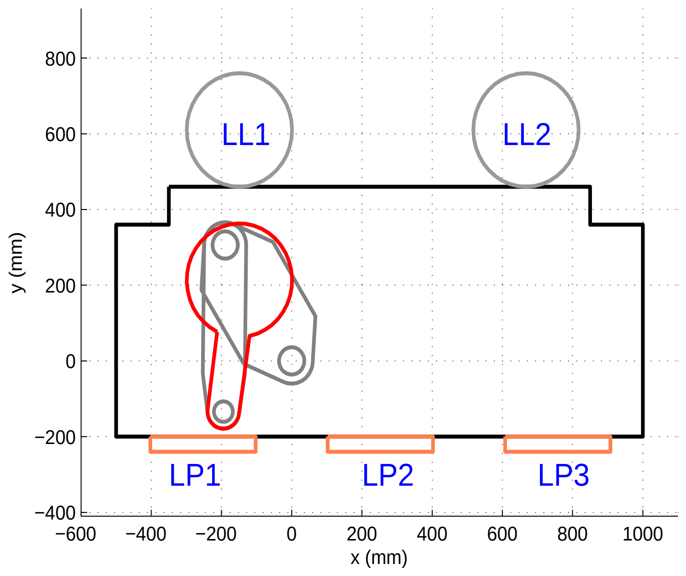
There are three samplers in the sampler pool each with a distinctive strength: Uniform sampler, Gaussian sampler and Bridge Test sampler. The uniform sampler provides good coverage of $\mathcal{C}$ space while the Gaussian and Bridge Test samplers deal with the narrow passages if there is any in the $\mathcal{C}$ space. The sampling strategy is to give reward to the sampler that generates useful point and punish sampler that generates useless point. Another important part in generating roadmap is the collision checking for points in robot $\mathcal{C}$ space. The collision checking for the PRM planner is based on geometric collision model which considers all the collision scenarios that can happen between the robot and the obstacle. This approach is complicated and hard to implement. However, when the CAD models of the robot as well as the EFEM are available, it is possible to utilize these CAD models for fast collision checking. For example, some bounding volume hierarchy based softwares [50] are available and can be implemented easily.

The result of the roadmap constructed is given in Fig.3.5 and Fig.3.6. The red region is the obstacle-free space, all the blue dots are sampled points (robot configurations) generated by samplers (Fig.3.5 (b)), and all the black lines show the connectivity between two points in the obstacle-free space (Fig.3.6). The roadmap shows that the proposed method can provide a good coverage of the whole obstacle-free space.

Once a roadmap is constructed, the PRM planner enters another phase: the query phase. With a given query, which is a pair of initial and goal configurations (q_I and q_G), an A-star search method is adopted to search through the roadmap for a shortest path, as shown in



a) An initial robot configuration inside the EFEM



b) A Goal robot configuration inside the EFEM

Figure 3.4: Motion is considered inside the EFEM

Fig.3.6 (red line).

3.4 Trajectory Generation Using Cubic B-spline

With the path searched from the roadmap, it has to be smoothed out and parametrized by time "t" to become a trajectory. The trajectory must be smooth and safe (no collision happens). Besides, the execution time should be as short as possible. In this Section, the trajectory is parametrized by time intervals based on cubic B-spline functions, then it is optimized with respect to the total execution time while not violating any operational constraints.

Blending within A Safe Region

The basic idea of trajectory smoothing is to blend the connecting points of the piecewise linear path searched from the roadmap (Fig.3.6) by spline curves. Because only the piecewise linear path is feasible, each connecting point of the path is first bounded by a box, which represents a safe region that the trajectory is safe as long as it remains inside the box, as shown in Fig.3.7. Once the boxes are generated, safe blending can be performed to generate smooth trajectory within the box.

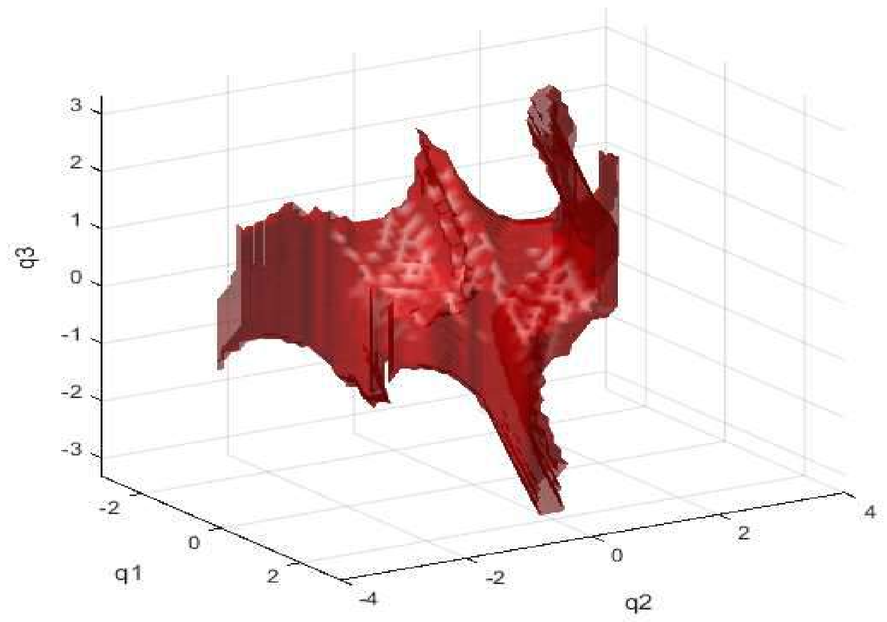
The bounding box is generated based on the shortest distance from the robot current configuration (defined by the connecting point) to the obstacle. The goal is to find a box for a given connecting configuration q, i.e, to find a Δq ($\Delta q = [\Delta q_1, \Delta q_2, \Delta q_3]^T$, and $\Delta q_i > 0$,$i = 1, 2, 3$ for 3 joints) such that a new configuration q' is a safe one as long as $|q'_i - q_i| < \Delta q_i, i = 1, 2, 3$. With the given shortest distance d from robot to obstacle at the given configuration q, Δq can be determined if all points on the robot links travel less than distance d when the robot moves from configuration q to q'.

This problem can be further decomposed for three joints respectively. First, for each joint (joint i), a Lipschitz condition is derived as:

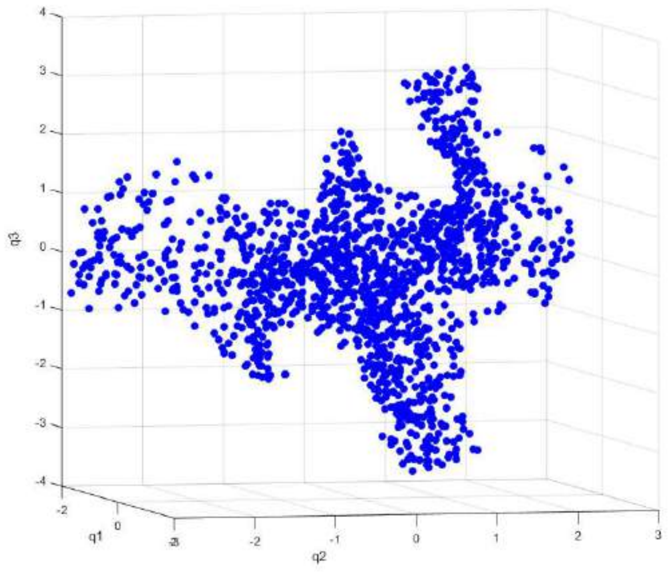
$$\begin{aligned} &\|a(q_1, \ldots, q_{i-1}, q_i, q_{i+1}, \ldots, q_n) - \\ &a(q_1, \ldots, q_{i-1}, q'_i, q_{i+1}, \ldots, q_n)\| \\ &< c_i|q_i - q'_i| \end{aligned} \tag{3.1}$$

in which a is any point on the robot, c_i is a Lipschitz constant, n is the number of joints (in this case, $n = 3$), and the Lipschitz condition is written in a general form. The goal is to make the Lipschitz constant c_i as small as possible to get a bigger variation in q_i. One example of determining c_i is in Fig.3.8. In this example, the robot has only one link, and c_i is simply the length of the robot r. The bound of the robot displacement is:

$$\|a(q) - a(q')\| < \sum_{i=1}^{n} c_i|q_i - q'_i| \tag{3.2}$$



a) Obstacle-free space



b) Samples generated by PRM (1500 nodes and 39900 edges, edges are not displayed)

Figure 3.5: Obstacle-free space and samples generated by PRM

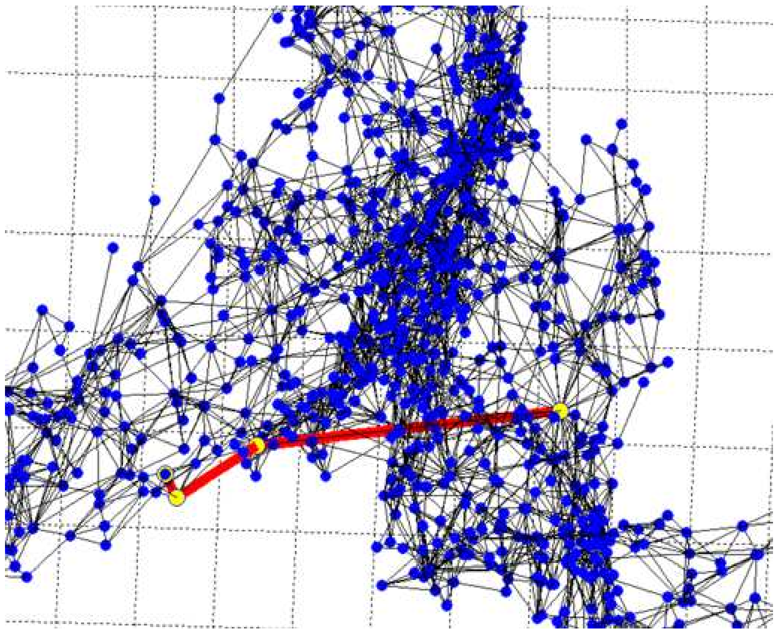


Figure 3.6: The shortest path connects q_I and q_G searched in roadmap by A^*

Let $\sum_{i=1}^{n} c_i|q_i - q_i'| < d$, Δq is derived as

$$\Delta q_i = \frac{d}{nc_i}, i = 1, 2, 3 \tag{3.3}$$

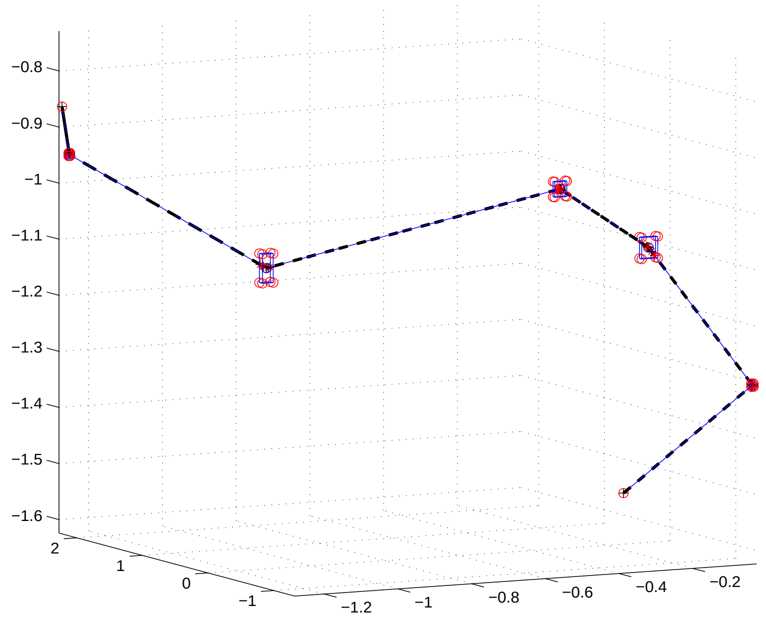
The bounding box for configuration q is determined by Δq, and the intersection of straight line segments and bounding boxes are also determined (Fig.3.7(b)).

Parametrizing Trajectory by Cubic B-splines

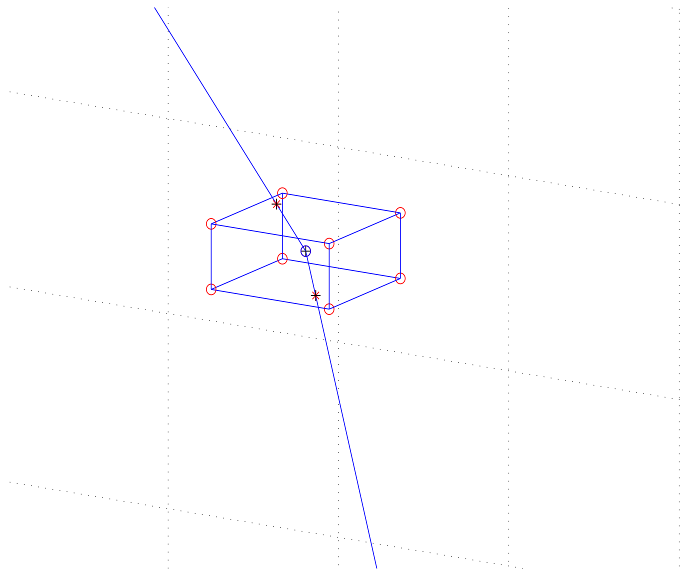
First, a B-spline curve is defined by its degree $k-1$, order k, control points, and nodes. A B-spline curve $p(t)$ is expressed as a linear combination of base functions weighted by control points.

$$p(t) = \sum_{i=0}^{n} CP_i N_{i,k}(t), t_{k-1} \leq t \leq t_{n+1} \tag{3.4}$$

where $CP_0, \ldots, CP_n$ are $n+1$ control points, $t_{k-1}, \ldots, t_{n+1}$ are nodes (which is a time sequence, and it is clamped. i.e, the values have multiplicity k at extremeties so that $p(t_{k-1})$ and $p(t_{n+1})$ coincide with the first and last control points respectively), and $N_{i,k}(t)$'s are base



a) Bounding boxes for all connecting points of a path



b) Bounding box for a single connecting point

Figure 3.7: Bounded region of a given piece wise linear path

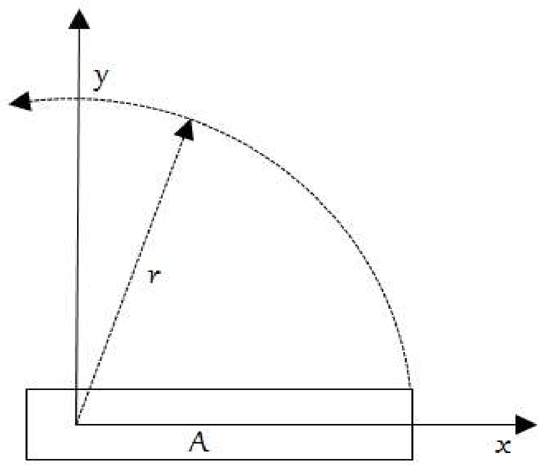


Figure 3.8: One link robot example for determining c_i

functions recursively determined by de Boor-Cox formula:

$$
\begin{aligned}
&N_{i,0}(t) = \begin{cases} 1, & t_i \le t \le t_{i+1} \\ 0, & else \end{cases} \\
&N_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} N_{i,k-1}(t) \\
&+ \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(t), \\
&and\,let\, \tfrac{0}{0} = 0.
\end{aligned}
\tag{3.5}
$$

Symbols used when defining a B-spline curve are given in Table 3.1. Because cubic spline curves are at least twice differentiable, which is a favourable property for trajectory smoothness, the cubic B-spline is selected for path fitting.

The trajectory should be restricted within the path and bounded boxes obtained in the previous sections, i.e., the blending curves connecting adjacent straight lines must be inside bounded boxes. To meet these requirements, with the strong convex hull property of the B-spline (which is given below), control points are distributed as follows: triple coincident control points are put at the start and end of the path, which ensures that the start and the end of the trajectory are complete stops (zero speed and acceleration); Single control points are put at all connecting points as well as at the midway of intersection points and connecting points. An example of the distribution of the control points is in Fig. 3.9. With this distribution, the B-spline curve is connected by several segments, and all of them are safe.

Strong Convex Hull property: A B-spline curve is contained in the convex hull of its control poly line. To be specific, if t is in knot span $[t_i, t_{i+1})$,$p(t)$ is in the convex hull of

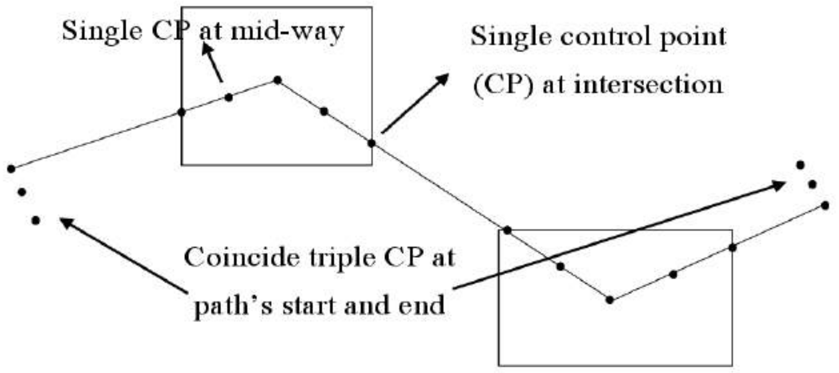

Single CP at mid-way
Single control point (CP) at intersection
Coincide triple CP at path's start and end

Figure 3.9: Control points distribution example

Table 3.1: Symbols used when define the B-spline

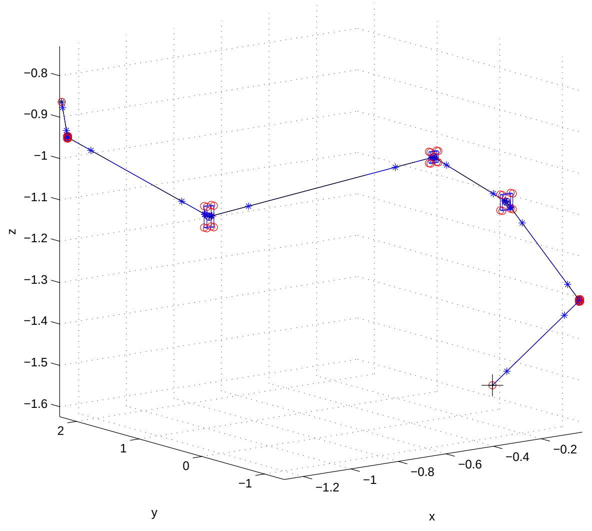
Symbol	Definition
$k-1$	Degree of the B-spline
k	Order of the B-spline
$p(t)$	B-spline curve
$N_{i,k-1}(t)$	Base function of degree k-1
CP_i	Control points of the B-spline
$n+1$	Number of control points
t_i	Nodes of the B-spline(time sequence)
$k+n+1$	Number of nodes

control points $CP_{i-k+1}, CP_{i-k+2}, \dots, CP_i$. This is because $N_{i,k-1}(t) \geq 0, \sum N_{i,k-1}(t) = 1$.

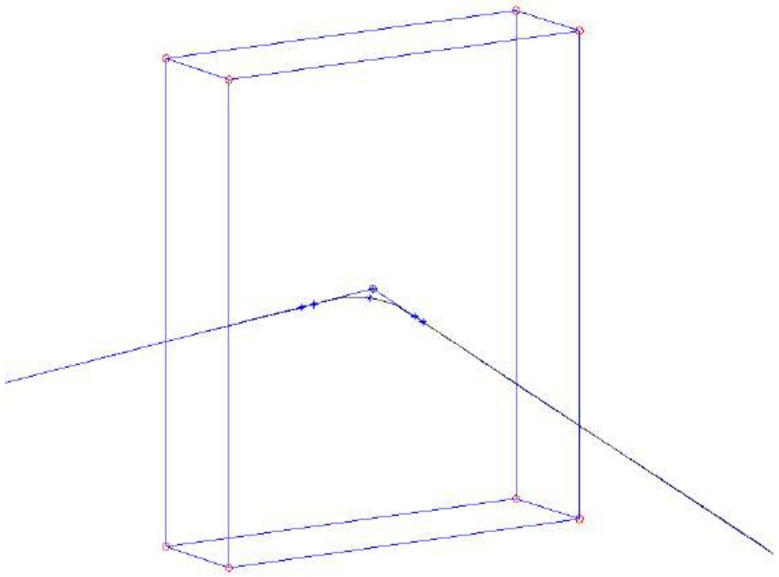
With this property, for a B-spline curve of degree $k-1=3$, supposing that control points are $CP_0, \dots, CP_n$, $k+n+1$ nodes are needed for a complete B-spline trajectory. Let nodes be $t_0, \dots, t_{k+n}$, which is in fact a time sequence. The trajectory $p(t)$ is parametrized by time sequence while under the safe operation condition. One trajectory with a time sequence substituted is given in Fig.3.10, and the trajectory is a safe one.

The Optimization Problem with Kinematic Constrains

The trajectory parametrized by a time sequence $t_0, \dots, t_{n-k+2}$ features sufficient smoothness properties, i.e, it is guaranteed that the trajectory, velocity and acceleration are continuous



a) Trajectory with time sequence substituted



b) Zoom in of a segment of the trajectory

Figure 3.10: Trajectory generated by B-spline curve

functions. In industrial applications, however, the execution time is crucial for manufacturing throughput. Thus the trajectory should be further optimized while taking into account the kinematic constraints.

Kinematic constraints are usually boundaries of joints' velocity and acceleration. However, in this application, the kinematic constraints are not only defined in robot joint space, but also in robot work space. The acceleration of the wafer center in the work space must be limited because the wafer is in contact with the blade only by means of friction. It has been proved empirically that wafer sliding is avoided when wafer center acceleration is bounded.

To explicitly express the constraints in the optimization problem, some remarks about the derivatives of the B-spline trajectory are necessary. The derivative of a B-spline curve is

$$\begin{aligned} v(t) &= p'(t) = (\textstyle\sum_{i=0}^{n} CP_i N_{i,k}(t))' = \sum_{i=0}^{n} CP_i N'_{i,k}(t) \\ &= (k-1)\textstyle\sum_{i=1}^{n}(\frac{CP_i - CP_{i-1}}{t_{i+k}-t_i})N_{i,k-1}(t) \\ &= \textstyle\sum_{i=1}^{n} CPV_i \cdot N_{i,k-1}(t), \\ CPV_i &= (k-1)(\frac{CP_i - CP_{i-1}}{t_{i+k}-t_i})N_{i,k-1}(t) \end{aligned} \tag{3.6}$$

in which CPV_i are control points of velocity. Similarly, the acceleration is:

$$\begin{aligned} a(t) &= v'(t) = (\textstyle\sum_{i=1}^{n} CPV_i N_{i,k-1}(t))' \\ &= \textstyle\sum_{i=1}^{n-1} CPA_i \cdot N_{i,k-2}(t), \\ CPA_i &= \frac{k-2}{t_{i+k-1} - t_{i+1}}(CPV_i - CPV_{i-1}) \end{aligned} \tag{3.7}$$

in which CPA_i are control points of acceleration. Thus the trajectory, velocity and acceleration are all expressed in B-spline form. Recall the Convex Hull property of B-spline curve, which states that a B-spline curve is contained by all of its control points. Thus to bound a B-spline curve, it is sufficient (not necessary) to bound its control points instead. The kinematic constraints on velocity and acceleration in joint space are then written as:

$$\begin{aligned} |CPV_{j,i}| &\leq LV_j, i = 1, \dots, n, j = 1, 2, 3 \\ |CPA_{j,i}| &\leq LA_j, i = 1, \dots, n-1, j = 1, 2, 3 \end{aligned} \tag{3.8}$$

in which j is the joint number, and LV_j and LA_j are limits of velocity and acceleration respectively. It should be noted that all control points of velocity and acceleration are functions of time sequence.

The wafer center acceleration in work space is first calculated by forward kinematics and is expressed as a piece-wise function parametrized by time t and time sequence $t_0, \dots, t_{n-k+2}$. The acceleration constraints imposed on acceleration function should be held for any time instant, which is an infinite constraints problem that is hard to address. Moreover, because of the forward kinematics involves complex non-linearity of the robot kinematics, the acceleration constraints in the work space requires high computational load. A pseudo semi-infinite constraints approach is proposed to express the kinematic constrains in work space efficiently.

The acceleration function at all time sequence $t_0, \ldots, t_{n-k+2}$ are sampled and set within the constraints.

$$\begin{aligned} &|Acc\_ws_j(t = t_i; \{t_0, \ldots, t_{n-k+2}\})| \leq LA\_ws_j, \\ &i = 0, \ldots, n-k+2, j = x, y \end{aligned} \tag{3.9}$$

in which Acc_ws is the acceleration function, LA_ws is the acceleration limit in work space, and j stands for two axes x and y.

It should also be noted that there is constraint set on the time intervals, which is due to the fact that they are lower bounded because of the limits of the joint velocity. It is hard to get this lower bound because the joint positions are not fixed at the time spot in the time sequence $t_0, \ldots, t_{n-k+2}$. However, it is possible to get a rough lower bound as a substitute by considering going straight line from initial to goal point at maximum speed. Besides, t_0 should be zero.

$$\begin{aligned} &t_i - t_{i-1} \geq \min_j \frac{|q_G - q_I|}{LV}, \\ &i = 1, \ldots, n-k+2, j = 1, 2, 3, t_0 = 0 \end{aligned} \tag{3.10}$$

After all kinematic constraints are defined, the objective function of the optimization problem is defined as the total execution time.

$$\begin{aligned} &\underset{\{t_0, \ldots, t_{n-k+2}\}}{min} \; t_{n-k+2} \\ &subject\ to\ kinematic\ constraints (Eq.\ 3.8, 3.9, and\ 3.10) \end{aligned} \tag{3.11}$$

3.5 Simulation Results of The Optimized Trajectory

The optimization problem formulated in Eq.(3.11) is a constrained non-linear optimization problem which is usually solved by iteration. Thus the optimization problem is sensitive to the initial point which affects the running time or even the result.

First the kinematic limits are given in Table 3.2. For the robot moving from loadport1 to loadlock1, the initial and goal configurations corresponding to the given loadport and loadlock position are shown in Fig.3.4. The solution of the optimization problem is obtained by an "interior-point" method in Matlab. The "interior-point" method requires that the initial searching point is feasible. By iteratively increasing the gain of the barrier function, it is proved that the "interior-point" method converge to a solution [51].

The feasible initial point can be determined by the following steps. First a initial time sequence $\mathbf{h}$ is obtained by considering the lower bound of time intervals with Table 3.2, which is a uniformly distributed time sequence with the final time as $\min_j \frac{|q_G - q_I|}{LV}, j = 1, 2, 3$. Then by substituting $\mathbf{h}$ into Eq.(3.4), the B-spline curve $p(t)$ is obtained, and velocity control points (CPV_i), acceleration control points (CPA_i) as well as acceleration in the workspace (Acc_ws_j) are all obtained. A scaled time variable τ is defined as: $\tau = \lambda t$, and λ can be computed by the control points and workspace acceleration, as given in Eq.(3.12). Then the initial point is $\mathbf{h}^{(0)} = \lambda \mathbf{h}$.

Table 3.2: Values of kinematic limits

Joints	Kinematic limits in joint space	
	Velocity[rad/s]	Acceleration[rad/s^2]
1	4.2	8.4
2	8.4	16.8
3	6.3	12.6
Axis	Kinematic limits in work space	
	Acceleration[mg]	
x	100	
y	100	

$$\begin{aligned}
&\lambda_1 = max_{j,i}\frac{CPV_{j,i}}{LV_j}, i = 1,\ldots,n, j = 1,2,3\\
&\lambda_2 = max_{j,i}\frac{CPA_{j,i}}{LA_j}, i = 1,\ldots,n-1, j = 1,2,3\\
&\lambda_3 = max_{j,i}\frac{Acc\_ws_{j,i}}{LA\_ws_j}, i = 1,\ldots,n, j = x,y\\
&\lambda = max(1\ \lambda_1\ \sqrt{\lambda_2}\ \sqrt{\lambda_3})
\end{aligned} \tag{3.12}$$

Fig.3.11 gives the motion sequence of the generated trajectory. The simulation results in Fig.3.12-Fig.3.13 show that all constraints are satisfied. Figure 3.12 shows the positions and velocities of robot's three joints of the optimized trajectory. Figure 3.13 shows both the accelerations of robot's three joints and wafer center (in x and y directions in work space). Noting that the trajectory is far from the constraints in joint velocity and acceleration is mainly because the workspace acceleration constraint is more limited. The proposed method also works when the kinematic constraints are small. The change of the kinematic limits will directly change the initial point (always feasible), and the optimization process will optimize over the initial point and remain feasible.

The resulting optimized trajectory is not global optimal. It is a sub-optimal trajectory. First, the kinematic constraints imposed on the velocity and acceleration(Eq.(3.8)) are only sufficient but not necessary, which means that there might exist a better trajectory though violating the constraints. Also because the optimization is a non-convex one, it is hard to guarantee that the solution is global optimal. However, given the optimization with inequality constraints and feasible initial point, the "interior-point" method is guaranteed to output an improved feasible solution.

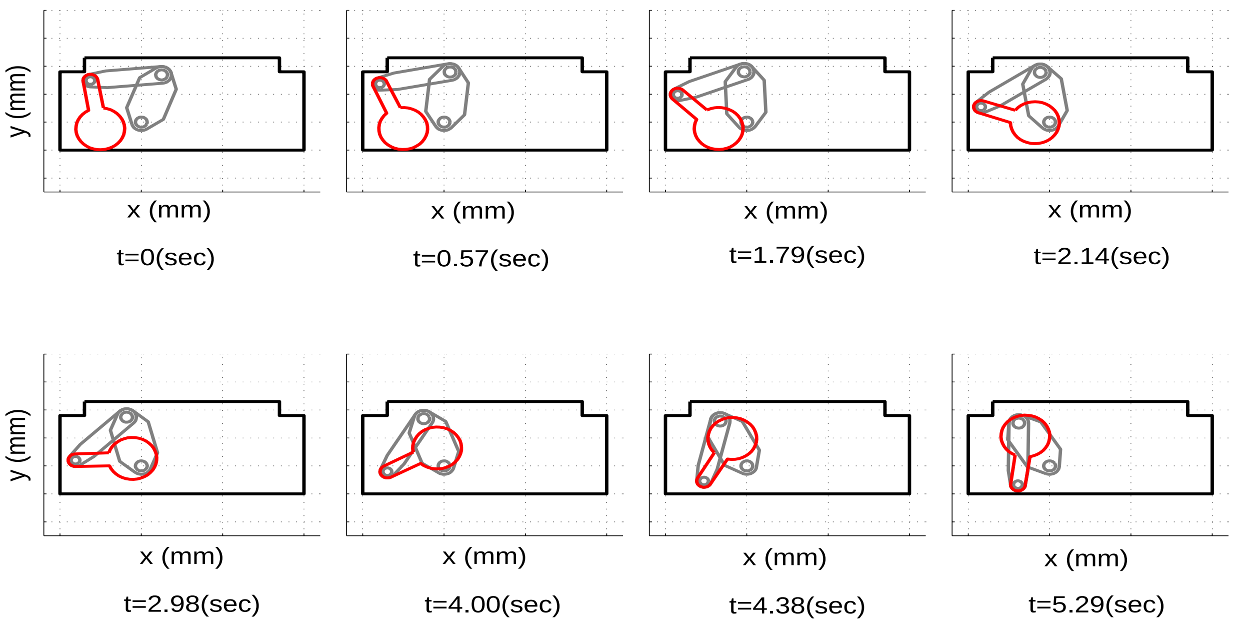


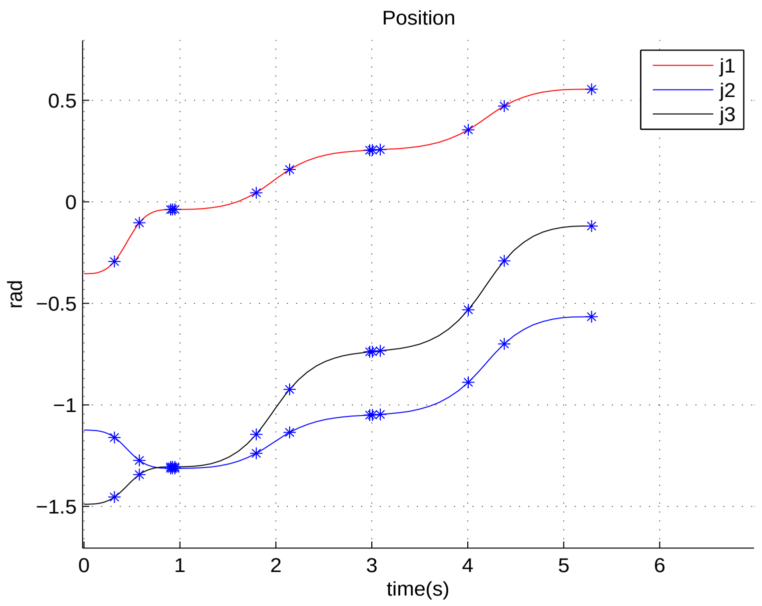
Figure 3.11: The motion sequence of the trajectory

3.6 Chapter Summery

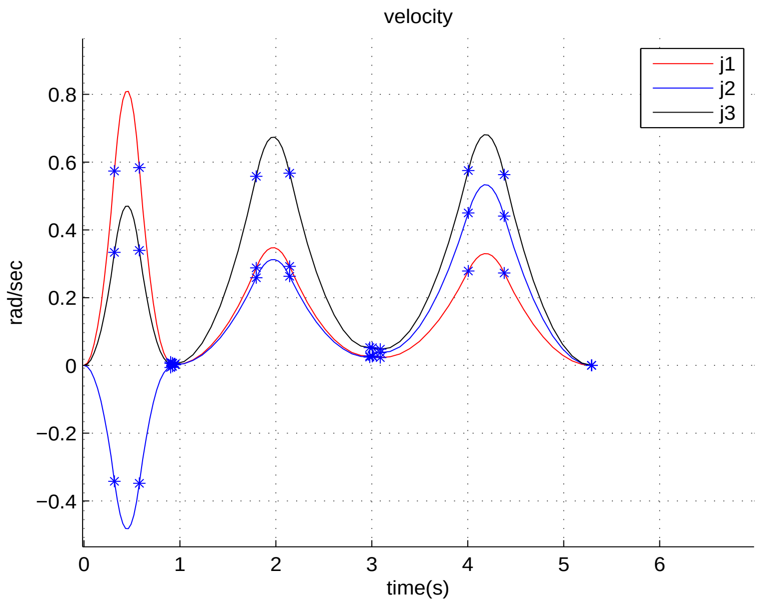
In this Chapter, automation of trajectory planning for industrial manipulators with operational constraints is attempted to solve with combined statistical learning method and optimization strategy. Sampling based method, which searches through the robot $\mathcal{C}$ space with a sampling scheme does not consider path smoothness and non-linear differential constraints. On the other hand, optimization based method requires heavy computational loads or only works for a predetermined duration. By combining the two approaches: sampling and optimization based method, the trajectory planning can be solved not only fast but also with certain optimality.

One example given in this Chapter is a wafer handling robot that works in the Equipment Front End Module (EFEM) of a semiconductor manufacturing machine. The generated trajectory is guaranteed to be a safe one without colliding with any obstacles and is optimized with respect to the total execution time for working efficiency. Unlike most existing methods for trajectory planning either partially solved the problem or with heavy computational load, the proposed trajectory planning method for this problem solves the problem with consideration of almost all aspects, which involves obstacle avoidance and cycle time optimization under kinematic constraints.

For the obstacle avoidance, first a roadmap describes the connectivity of the free space is

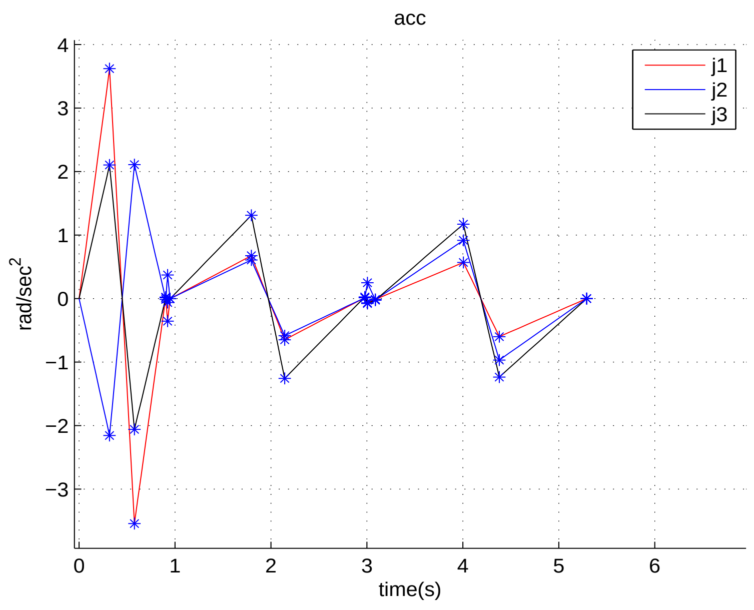


a) Position of the three joints for the optimized trajectory

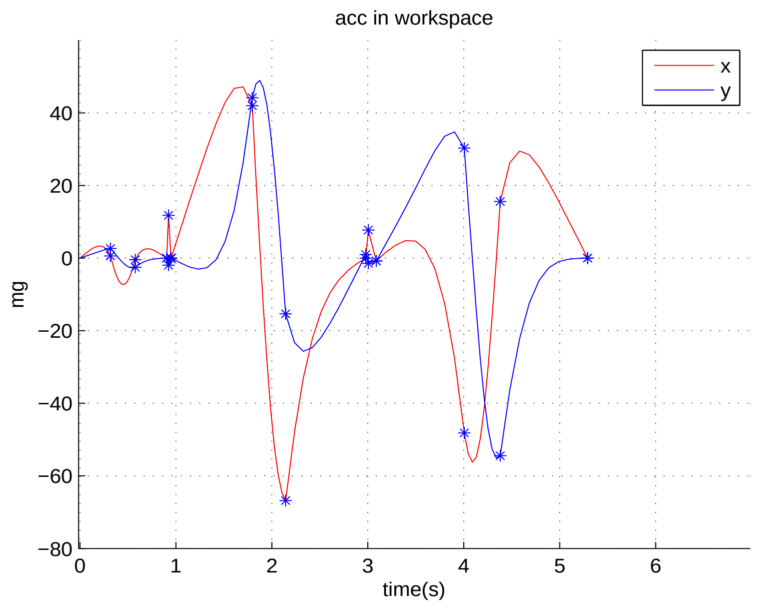


b) Velocity of the three joints for the optimized trajectory

Figure 3.12: Position and velocity of the three joints for the optimized trajectory (under constraints in Tab.3.2)



a) Acceleration of the three joints for the optimized trajectory



b) Acceleration in the workspace for the optimized trajectory

Figure 3.13: Acceleration for the optimized trajectory (under constraints in Tab.3.2)

constructed by the probabilistic roadmap (PRM) method with multiple samplers adaptively updated. Then a shortest path is searched for the given query, i.e, the initial and goal configuration of the robot. With the path searched from the roadmap, the trajectory is smoothed by means of Cubic B-spline parametrization, and all kinematic constraints are formulated in this manner.

The problem is formulated as an optimization problem and solved by an "interior point" method and the results show the effectiveness of the proposed method.

Chapter 4

Intelligent System Identification for Feedforward Compensation

4.1 Introduction

In industrial autonomous manufacturing, robotic manipulators currently play an important role, which are required to perform accurate tracking in many industrial applications. The use of parallel manipulators [52, 53], which always have a closed loop kinematic chain opposed to serial link robots, has been increased due to some advantages found for parallel manipulators. Parallel manipulator have higher stiffness, lighter and faster structures which leads to a better force distribution, easier inverse kinematics computing, and so on. This class of robots is ideal for a better manipulation of heavy loads which yield accurate positioning results. Some reported applications for this class of robots are haptic devices, lifting applications (even lifting of other manipulators), product transportation and classification, flight simulators, and so on [54].

Typically, parallel robots offer greater efficiency and faster acceleration at the end-effecter, and therefore are more suitable for fast assembly lines. However, because of the complexity of related kinematics and dynamics analyses, the derivation of dynamic equations of motion for parallel robots is more involved. Formulation of such equations that are suitable for controller design is still an open question. One popular method [55, 56] is to first separate the closed chain into several open chains, and then derive the equations of motion. This method will lead to a set of differential algebraic equations that are not easy to solve. Uicker [57] tried to formulate the equations of motion in terms of all dependent generalized coordinates, and then eliminated all holonomic constrains to end up with differential equations with independent generalized coordinates. Getting explicit motion equations are generally impossible because of the complexity of parallel robot kinematics. But once explicit dynamic equations of motion are obtained, it is possible to extend the exsisting robot control strategies to parallel robots [58].

One example is the silicon wafer handling robot. The processing of silicon wafers, which is a thin slice of silicon used in the fabrication of integrated circuits(IC), is complex and is involved with chemistry and physics. During this process, even a small particle may ruin the entire wafer, and result in hundreds or even thousands of dollars lost. Thus to successfully transform the silicon wafer into IC, this process must be finished under the absolute absence of contaminants. Normally, this process is operated in vacuum environment, with contaminants stringently controlled. Today, etch machines with multiple chambers are used for wafer fabrication. The wafer handling system is used to automatically transfer wafers among different chambers of a semiconductor etch machine. The wafer handling robot is a part of the handling system. To fulfill the handling task, 2 degrees of freedom is necessary. Due to the nature of high stiffness, kinematical closed loop robots or parallel robots are more suited for this task than the serial robots (e.g. SCARA). In order to improve the working efficiency, a parallel robot with dual blade, which adopts two identical closed loop mechanisms, is used for wafer handling, as shown in Fig.4.1.

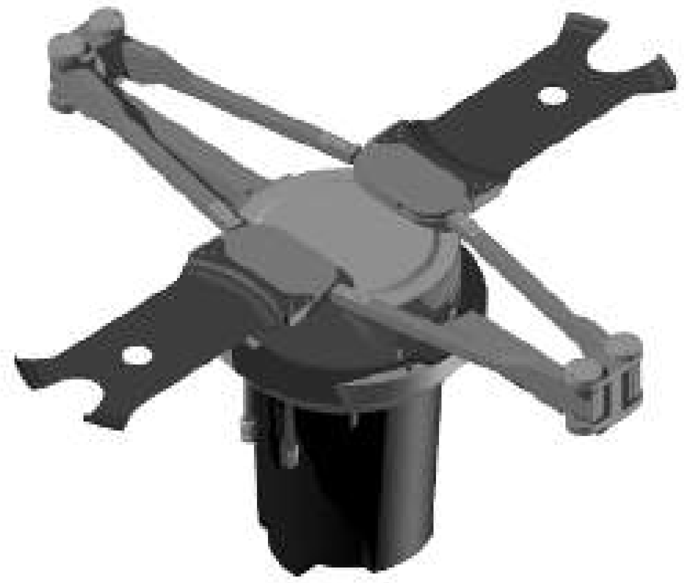


Figure 4.1: Dual blade wafer handling robot

Though closed loop mechanism is used to enhance the stiffness of the robot, due to the chamber entrance size, the link is quite thin and thus introduces some flexibility to the system. Moreover, the magnetic coupler used for isolation of the robot link and load on the vacuum side from the motor on the atmospheric side introduces not only compliance in torque transmission but also actuation-sensing mismatch problem. The result is that, with only a PID controller, the robot may exhibit a considerable vibration and produce non-negligible tracking error during the tracking of a desired wafer handling trajectory. Large tracking error reduces the quality of products, and vibration results in wafer contamination by production of particles.

It should be noted that the tuning of the fixed structure controllers such as PID controllers may not be sufficient for the wafer handling robot to meet the performance specifications. Robot dynamic control schemes such as the computed torque method may potentially achieve high trajectory tracking accuracy and tracking smoothness. These approaches always use robot dynamic model to calculate the desired drive torque. Although it is always hard to obtain the dynamic parameters of robots, the structure of the wafer handling robot enables the identification of all dynamic parameters.

In this Chapter, we exploit Lagrange's equations of motion to derive the dynamic model of a dual-blade wafer handling robot. An explicit form of dynamic model is obtained. With the assumption of accurate lengths of links, we transform the dynamic model into a decoupled form to enable dynamic identification by least square regression [59]. A data-driven approach is proposed to solve the problems of decoupling and regression. Both the Lagrange's equations of motion and identification procedure are verified through numerical simulation. Friction is also modeled to make the dynamic model more reliable. Identification experiment is performed on an actual dual-blade robot. The result has demonstrated the correctness of the dynamic model and the feasibility of the identification method. The identified model is finally used by a feedforward controller for improved performance.

The rest of the Chapter is organized as follows: in Section 4.2, the dynamic modeling of the wafer handling robot is given using Lagrangian method, including the kinematics of the dual blade robot. In Section 4.3, the robot dynamics is transformed to a decoupled form for identification, and the identification result is verified by numerical simulation. Section 4.4 shows how a feedforward controller is applied to the robot with the identified dynamic model. Section 3.5 concludes the Chapter.

4.2 Dynamics Modeling

Kinematics of Dual Blade Wafer Handling Robot

The dual blade wafer handling robot adopts two frog-legged five-bar mechanisms, as shown in Fig.4.1. From the perspective of kinematics, this robot is equivalent to the diagram shown in Fig.4.2. The five-bar mechanism adopted in the dual blade robot is driven by the two rotational joints 1 and 5. The motion of joints 3 and 4 are restricted to ensure the robot has the desired motion, i.e. the symmetric motion with respect to the extensional axis. This robot can only perform one rotational motion and one extensional motion, as shown in Fig.4.2.

Two types of motion of this robot are considered separately. First, the extensional motion is considered. Specific corotational coordinates are used to simplify the kinematics analysis. Corotational coordinates are α and ϵ as specified in Fig.4.3. Note α and ϵ are not the rotation

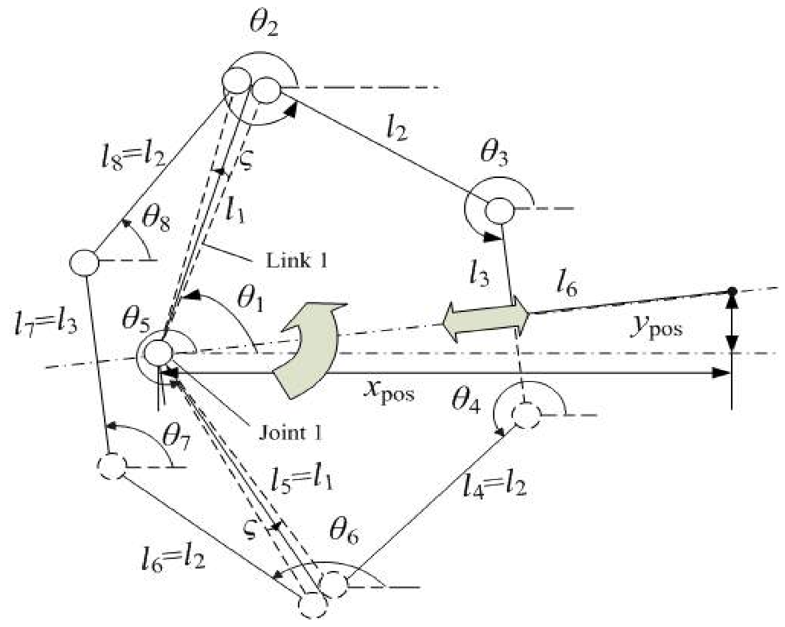


Figure 4.2: Kinematics of the dual blade wafer handling robot

angles of the motors. Their zero reference line is the perpendicular bisector of the wrist link (l_3). All the angles in Fig.4.3 span from $-\pi$ to π.

The relationships between the angles are:

$$\begin{cases} l_1 sin(\alpha) + l_2 sin(\beta) - \frac{l_3}{2} = 0 \\ \gamma + \frac{\pi}{2} = 0 \\ \delta + \beta + \pi = 0 \\ \epsilon + \alpha = 0 \\ l_2 sin(\mu) = l_1 sin(\alpha + \zeta) - \frac{l_3}{2} \\ \eta = \frac{\pi}{2} \\ \phi + \mu = \pi \end{cases} \tag{4.1}$$

Again, note α and ϵ are not the rotation angles of the motors. The solution of the above equation is:

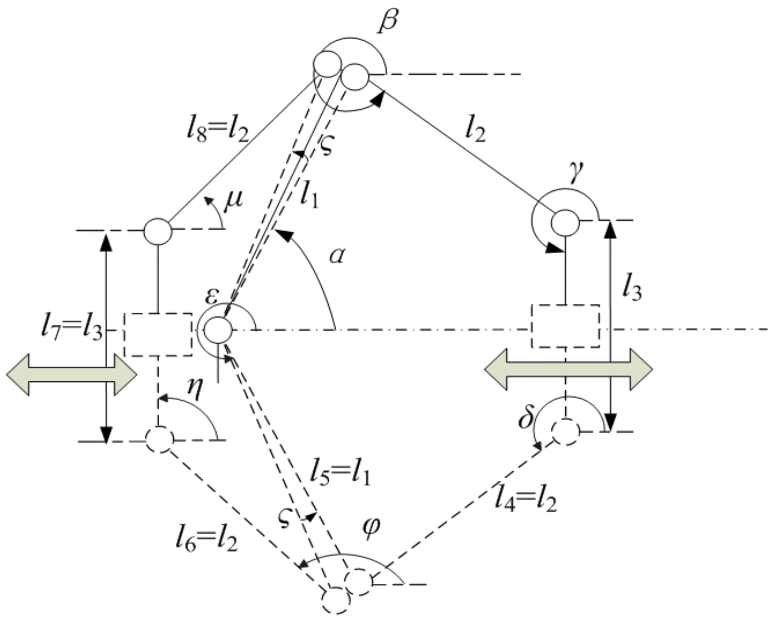


Figure 4.3: Corotational coordinates for extension motion

$$\begin{cases} \beta = arcsin(\frac{\frac{l_3}{2} - l_1 sin(\alpha)}{l_2}) \\ \gamma = -\frac{\pi}{2} \\ \delta = -\pi - arcsin(\frac{\frac{l_3}{2} - l_1 sin(\alpha)}{l_2}) \\ \epsilon = -\alpha \\ \mu = arcsin(\frac{l_1 sin(\alpha+\zeta) - \frac{l_3}{2}}{l_2}) \\ \eta = \frac{\pi}{2} \\ \phi = \pi - arcsin(\frac{l_1 sin(\alpha+\zeta) - \frac{l_3}{2}}{l_2}) \end{cases} \tag{4.2}$$

Now consider the rotation motion with the extension motion. Let the angle between link i and the $x-axis$ (the one fixed to the ground) be θ_i, as shown in Fig.4.2. Note θ_1 and θ_5 are the rotational angles of the motors. The angles have the following relationships:

$$\begin{cases} \theta_1 = \alpha + \frac{(\theta_1+\theta_5)}{2} \\ \theta_2 = \beta + \frac{(\theta_1+\theta_5)}{2} \\ \theta_3 = \gamma + \frac{(\theta_1+\theta_5)}{2} \\ \theta_4 = \delta + \frac{(\theta_1+\theta_5)}{2} \\ \theta_5 = \epsilon + \frac{(\theta_1+\theta_5)}{2} \\ \theta_6 = \phi + \frac{(\theta_1+\theta_5)}{2} \\ \theta_7 = \eta + \frac{(\theta_1+\theta_5)}{2} \\ \theta_8 = \mu + \frac{(\theta_1+\theta_5)}{2} \end{cases} \tag{4.3}$$

All the angles can be expressed as functions of the rotational angles of the motors:

$$\begin{cases} \theta_2 = \frac{\theta_1}{2} + \frac{\theta_5}{2} + arcsin(\frac{l_3 - 2l_1 sin(\frac{\theta_1}{2} - \frac{\theta_5}{2})}{2l_2}) \\ \theta_3 = \frac{\theta_1}{2} + \frac{\theta_5}{2} - \frac{\pi}{2} \\ \theta_4 = \frac{\theta_1}{2} + \frac{\theta_5}{2} - \pi - arcsin(\frac{\frac{l_3}{2} - l_1 sin(\frac{\theta_1}{2} - \frac{\theta_5}{2})}{l_2}) \\ \theta_6 = \pi - arcsin(\frac{l_1 sin((\frac{\theta_1}{2} - \frac{\theta_5}{2}) + \zeta) - \frac{l_3}{2}}{l_2}) + \frac{\theta_1+\theta_5}{2} \\ \theta_7 = \frac{\pi}{2} + \frac{\theta_1+\theta_5}{2} \\ \theta_8 = arcsin(\frac{l_1 sin((\frac{\theta_1}{2} - \frac{\theta_5}{2}) + \zeta) - \frac{l_3}{2}}{l_2}) + \frac{\theta_1+\theta_5}{2} \end{cases} \tag{4.4}$$

The relationships between all joint angles and the independent generalized coordinates are then given explicitly by Eq.4.4.

Dynamics

Dynamics model can be derived by Lagrange's equations of motion. Lagrangian of this robot only includes kinetic energy as this is a planar robot. The Lagrange's equations are:

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_1}\right) - \frac{\partial T}{\partial \theta_1} = \tau_1 \\ \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_5}\right) - \frac{\partial T}{\partial \theta_5} = \tau_2 \end{cases} \tag{4.5}$$

where τ_1 is the torque applied to link 1 by motor 1, and τ_2 is the torque applied to link 5 by motor 2. T is the kinetic energy, which is the sum of the kinetic energy of all links.

$$T = \sum_{i=1}^{8} T_i \tag{4.6}$$

where

$$\begin{cases} T_i = \frac{1}{2} J_i \dot{\theta}_i^2, & i = 1, 5 \\ T_i = \frac{1}{2} m_i \|\boldsymbol{v}\|_2^2 + \frac{1}{2} J_i \dot{\theta}_i^2, & i = 2, 3, 4, 6, 7, 8 \end{cases} \tag{4.7}$$

J_i's for link 1 and 5 are computed with respect to the origin. J_i's for other links are computed with respect to the center-of-mass. m_i is the mass of link i. $\boldsymbol{v}_i$ is the velocity of the center-of-mass of link i. The explicit kinematics solution enables the explicit relationship between link velocity and motor speed.

$$\boldsymbol{v}_i = \boldsymbol{J}_{vi} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_5 \end{pmatrix}, \quad \dot{\theta}_i = \boldsymbol{J}_{\omega i} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_5 \end{pmatrix} \tag{4.8}$$

Both $\boldsymbol{J}_{vi}$ and $\boldsymbol{J}_{\omega i}$ can be obtained through symbolic computation. The terms in Lagrange's equations can then be derived explicitly as:

$$\begin{aligned} \frac{d}{dt}\begin{pmatrix} \dfrac{\partial T_i}{\partial \dot{\theta}_1} \\ \dfrac{\partial T_i}{\partial \dot{\theta}_5} \end{pmatrix} &= \left(m_i \boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi} + J_i \boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i}\right) \ddot{\boldsymbol{\theta}} \\ &\quad + \left(m_i \frac{d}{dt}\left(\boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi}\right) + J_i \frac{d}{dt}\left(\boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i}\right)\right) \dot{\boldsymbol{\theta}} \end{aligned} \tag{4.9}$$

and

$$\begin{pmatrix} \dfrac{\partial T_i}{\partial \theta_1} \\ \dfrac{\partial T_i}{\partial \theta_5} \end{pmatrix} = \begin{pmatrix} \dfrac{1}{2} m_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi}\right)}{\partial \theta_1} \dot{\boldsymbol{\theta}} + \dfrac{1}{2} J_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i}\right)}{\partial \theta_1} \dot{\boldsymbol{\theta}} \\ \dfrac{1}{2} m_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi}\right)}{\partial \theta_5} \dot{\boldsymbol{\theta}} + \dfrac{1}{2} J_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i}\right)}{\partial \theta_5} \dot{\boldsymbol{\theta}} \end{pmatrix} \tag{4.10}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_5)^T$. The Lagrange's equations of motion can be easily rewritten into the form of

$$\boldsymbol{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \boldsymbol{\tau} \tag{4.11}$$

where $\boldsymbol{\tau} = (\tau_1, \tau_2)^T$. In order to reduce symbolic computation complexity, both $\boldsymbol{M}$ and $\boldsymbol{C}$ are decomposed into 8 parts with respect to 8 links as

$$\begin{cases} \boldsymbol{M}(\boldsymbol{\theta}) & = \sum_{i=1}^{8} \boldsymbol{M}_i(\boldsymbol{\theta}) \\ \boldsymbol{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) & = \sum_{i=1}^{8} \boldsymbol{C}_i(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \end{cases} \tag{4.12}$$

where

$$\begin{aligned} \boldsymbol{M}_i(\boldsymbol{\theta}) &= m_i \boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi} + J_i \boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i} \\ \boldsymbol{C}_i(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) &= m_i \frac{d}{dt}\left(\boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi}\right) + J_i \frac{d}{dt}\left(\boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i}\right) \\ &\quad + \begin{pmatrix} \dfrac{1}{2} m_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi}\right)}{\partial \theta_1} + \dfrac{1}{2} J_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i}\right)}{\partial \theta_1} \\ \dfrac{1}{2} m_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi}\right)}{\partial \theta_5} + \dfrac{1}{2} J_i \dot{\boldsymbol{\theta}}^T \dfrac{\partial\left(\boldsymbol{J}_{\omega i}^T \boldsymbol{J}_{\omega i}\right)}{\partial \theta_5} \end{pmatrix} \end{aligned} \tag{4.13}$$

4.3 Dynamics Model Decoupling

Model Decoupling

We try to identify the dynamic parameters with the most straightforward method, i.e. least-squares regression. The dynamic model should be transformed into the decoupled form $\boldsymbol{YZ} = \boldsymbol{\tau}$, where $\boldsymbol{Y}$ is a function of only motor rotation angles and $\boldsymbol{Z}$ is a function of only the mechanical parameters. This transformation is called YZ decoupling [60]. The resulting dynamics model treats the motor torque as a linear combination of functions of mechanical parameters. Assuming that there is no need to identify the lengths of links, this transformation can be derived.

Refer to section 4.2, $\boldsymbol{M}$ and $\boldsymbol{C}$ can be rewritten as sum of 8 parts with respect to 8 links as Equation (4.12). In this case, each link of this robot is treated as a 2-D rigid body instead of a beam. The center-of-mass can be arbitrarily set on a plane instead of along a line. This makes dynamics modeling more reliable and more complex. With simply kinematics analysis, it is not difficult to find for each link a linear relationship between the mechanical parameters and $\boldsymbol{M},\boldsymbol{C}$ in Eq.4.13. This equation can be further derived as

$$\begin{cases} \boldsymbol{M}_i(\boldsymbol{\theta}) & = \boldsymbol{K}_{mi}^1(\boldsymbol{\theta}) \left[J_i + m_i\left(r_i^2 + s_i^2\right)\right] \\ & \quad + \boldsymbol{K}_{mi}^2(\boldsymbol{\theta}) m_i r_i + \boldsymbol{K}_{mi}^3(\boldsymbol{\theta}) m_i s_i \\ & \quad + \boldsymbol{K}_{mi}^4(\boldsymbol{\theta}) m_i \\ \boldsymbol{C}_i(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) & = \boldsymbol{K}_{ci}^1(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) \left[J_i + m_i\left(r_i^2 + s_i^2\right)\right] \\ & \quad + \boldsymbol{K}_{ci}^2(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) m_i r_i + \boldsymbol{K}_{mi}^3(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) m_i s_i \\ & \quad + \boldsymbol{K}_{ci}^4(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) m_i \end{cases} \tag{4.14}$$

where $[r_i,\ s_i]^T$ is the position vector of the center-of-mass of link i in the link's coordinate system. $\boldsymbol{K}_{mi}^j$ and $\boldsymbol{K}_{ci}^j$ are functions of the motor rotation angles and velocities. For either link 1 or link 5, mechanical parameters include only J_i $(i = 1, 5)$. Therefore this robot has at most $4 \times 6 + 2 = 26$ independent mechanical parameters. This relationship enables construction of a decoupled dynamics model as

$$\begin{pmatrix}\boldsymbol{Y}_1 & \boldsymbol{Y}_2 & \cdots & \boldsymbol{Y}_{26}\end{pmatrix} \boldsymbol{Z} = \boldsymbol{\tau} \tag{4.15}$$

where $\boldsymbol{Z} = \left(J_1,\ J_2 + m_2\left(r_2^2 + s_2^2\right),\ m_2 r_2,\ \cdots,\ m_8,\ J_5\right)^T$ is a function of mechanical parameters. The aim of dynamic parameter identification is to obtain $\boldsymbol{Z}$ (given $\boldsymbol{Y}$). The least-square regression can be performed when sufficient experimental data has been acquired.

For the i^{th} experimental data,

$$\begin{pmatrix}\boldsymbol{Y}_1^i & \boldsymbol{Y}_2^i & \cdots & \boldsymbol{Y}_{26}^i\end{pmatrix} \boldsymbol{Z} = \boldsymbol{\tau}^i \tag{4.16}$$

All experimental data (totally k experiments) can be compiled as

$$\begin{pmatrix} \boldsymbol{Y}_1^1 & \boldsymbol{Y}_2^1 & \cdots & \boldsymbol{Y}_{26}^1 \\ \boldsymbol{Y}_1^2 & \boldsymbol{Y}_2^2 & \cdots & \boldsymbol{Y}_{26}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Y}_1^k & \boldsymbol{Y}_2^k & \cdots & \boldsymbol{Y}_{26}^k \end{pmatrix} \boldsymbol{Z} = \begin{pmatrix} \boldsymbol{\tau}^1 \\ \boldsymbol{\tau}^2 \\ \vdots \\ \boldsymbol{\tau}^k \end{pmatrix} \tag{4.17}$$

The large $\boldsymbol{Y}$ matrix above is called the regression matrix, denoted as $\boldsymbol{Y}_I$. For least-square regression purpose, $\boldsymbol{Y}_I$ should have a full column rank. For parallel robots, determining independent columns in an analytical way is difficult and inefficient. Here we use a data-driven method to find the largest set of linearly independent columns of $\boldsymbol{Y}_I$.

1) N groups (e.g. 100) of random positions, velocities and accelerations are chosen to construct the $\boldsymbol{Y}_I$ matrix.

2) A set $\boldsymbol{S}_n$ is constructed as the largest set of linearly independent columns for the first n columns of $\boldsymbol{Y}_I$. Let $\boldsymbol{Y}_n$ denotes the nth column of $\boldsymbol{Y}_I$. Initialize $\boldsymbol{S}_1 = [\boldsymbol{Y}_1]$.

3) From n=2 to 26, if $rank\,([\boldsymbol{S}_{n-1}, \boldsymbol{Y}_n]) > rank\,(\boldsymbol{S}_{n-1})$, $\boldsymbol{Y}_n$ must be linearly independent to columns in $\boldsymbol{S}_{n-1}$, thus $\boldsymbol{S}_n = [\boldsymbol{S}_n, \boldsymbol{Y}_n]$. Otherwise $\boldsymbol{S}_n = \boldsymbol{S}_{n-1}$.

The result of this algorithm is

$$\begin{aligned} \boldsymbol{S}_{26} = \ & [\boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{Y}_3, \boldsymbol{Y}_4, \boldsymbol{Y}_6, \boldsymbol{Y}_7, \boldsymbol{Y}_8, \boldsymbol{Y}_{10}, \boldsymbol{Y}_{11} \\ & \boldsymbol{Y}_{12}, \boldsymbol{Y}_{14}, \boldsymbol{Y}_{15}, \boldsymbol{Y}_{16}, \boldsymbol{Y}_{19}, \boldsymbol{Y}_{20}, \boldsymbol{Y}_{22}, \boldsymbol{Y}_{24}] \end{aligned} \tag{4.18}$$

Thus the dynamics model of VHP dual-blade robot can be rewritten to $\boldsymbol{Y}'\boldsymbol{Z}' = \boldsymbol{\tau}$, where $\boldsymbol{Y}'$ is a $(2 \times k) \times 17$ matrix (totally k experiments) with full column rank. For least-square regression purpose, $2 \times k$ should be greater than 17. In other word, the word "sufficient" means at least 8 groups of experiment data should be taken. More experiments can be taken to reduce the effect of noise.

Simulation Verification

To verify the dynamics model, we compare the solution of analytical closed-form model with a multi-body motion simulation. The analytical closed-form model is solved as differential equations in MATLAB, and the multi-body motion simulation is performed in commercial CAD software Pro/ENGINEER. In the simulated motion scenario, two constant torques with different magnitudes are applied to the two motors respectively. The resulting motion of the robot involves both rotation and extension. Due to the lack of design parameters (mass, inertia of moment, center of mass) of the actual dual-blade wafer handling robot, the mechanical parameters of the simulation are set manually.

The numerical solution obtained by MATLAB and the result of multi-body simulation are shown in Fig.4.5. The simulation has demonstrated the correctness of the dynamics model.

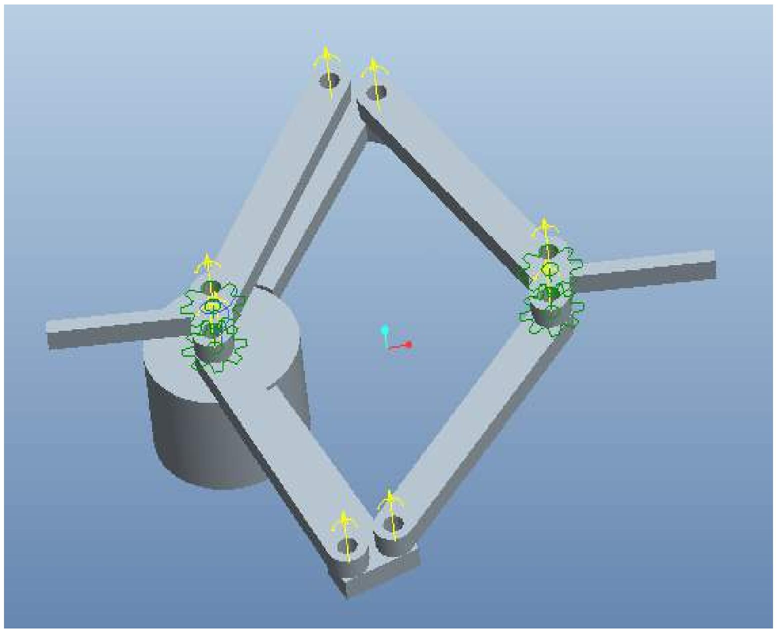


Figure 4.4: Multi-body simulation

Identification has also been simulated to verify the decoupled dynamic model. For this simulation, the mechanical parameters are set manually in the multi-body model. In the first simulation for the verification, time varying torques are applied to excite the robot in the multi-body simulation software, the data of motor rotation is collected, as shown in Fig.4.6.

Dynamic parameters identification is then performed in MATLAB with least-square regression. The identified mechanical parameters are used to generate a torque profile for a given trajectory. The given trajectory is a cubic function as in Eq.4.19, which has constant jerk, zero initial value for speed and acceleration.

$$\begin{cases} \theta_1 = 55 - 35t^3(deg) \\ \theta_5 = -55 + 35t^3(deg) \end{cases} \tag{4.19}$$

The torque profile for the given trajectory is then generated using identified dynamic model. This torque profile is applied to the robot in a new multi-body simulation. The resulting motion is compared with the given trajectory to verify the correctness of identification, as shown in Fig.4.7. Only motor angles are shown here. The angular speed and acceleration have the same consistency.

The simulation procedure has demonstrated the validity of decoupled dynamic model, as well as the feasibility of the dynamics identification.

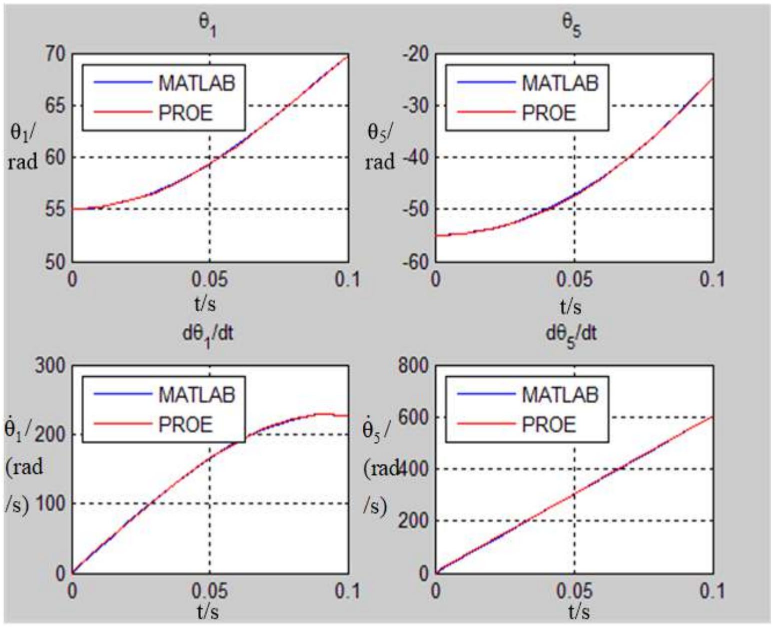


Figure 4.5: Verification of the dynamic model

Identification Experiment

Experiments on the actual dual-blade wafer handling robot indicate that friction has major influence on the dynamics. Figure 4.8 compares the experimental robot response and the simulated response based on the identified model with neglected friction. The red lines are torques computed using the identified model of a sinusoidal motion. The blue lines are the actual applied torque. The experimental and simulated responses exhibit significant difference due to the neglected friction. Other sources that may contributed to this includes joint dynamics and flexibility on the belt (located on joints 3, 4, 7 and 8).

Refering to [61], friction on all joints can be lumped on to the motor shafts, and sorted as decoupled and coupled friction. Coulomb and viscous friction are considered. We lump the distributed friction to joint 1 and 5. General viscous friction is in the form of:

$$\tau_{vf} = \begin{pmatrix} z_{18} & z_{19} \\ z_{20} & z_{21} \end{pmatrix} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_5 \end{pmatrix} \tag{4.20}$$

The Coulomb friction is the form

$$\tau_{vf} = \begin{pmatrix} z_{22} sign(\dot{\theta}_1) \\ z_{23} sign(\dot{\theta}_5) \end{pmatrix} + z_{24} sign(\dot{\theta}_1 - \dot{\theta}_5) \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{4.21}$$

where $z_{22}, z_{23}, z_{24} > 0$. The dynamics model is:

$$\boldsymbol{Y'Z'} + \boldsymbol{\tau}_{vf} + \boldsymbol{\tau}_{cf} = \boldsymbol{\tau} \tag{4.22}$$

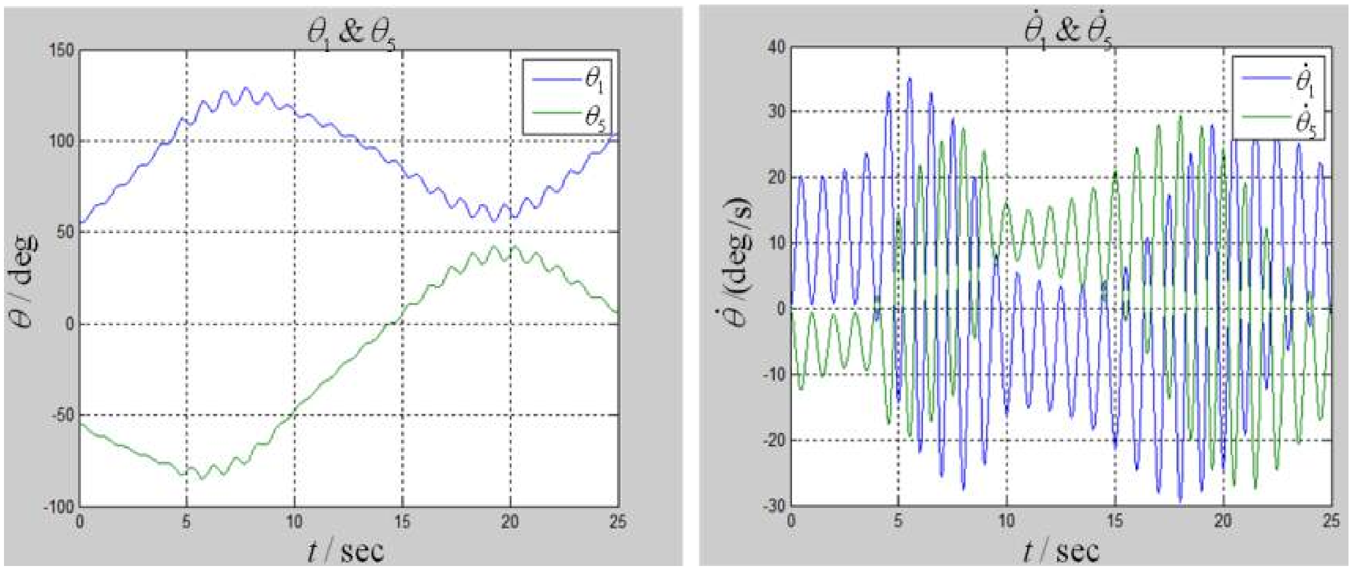


Figure 4.6: Motor rotation angle from simulation

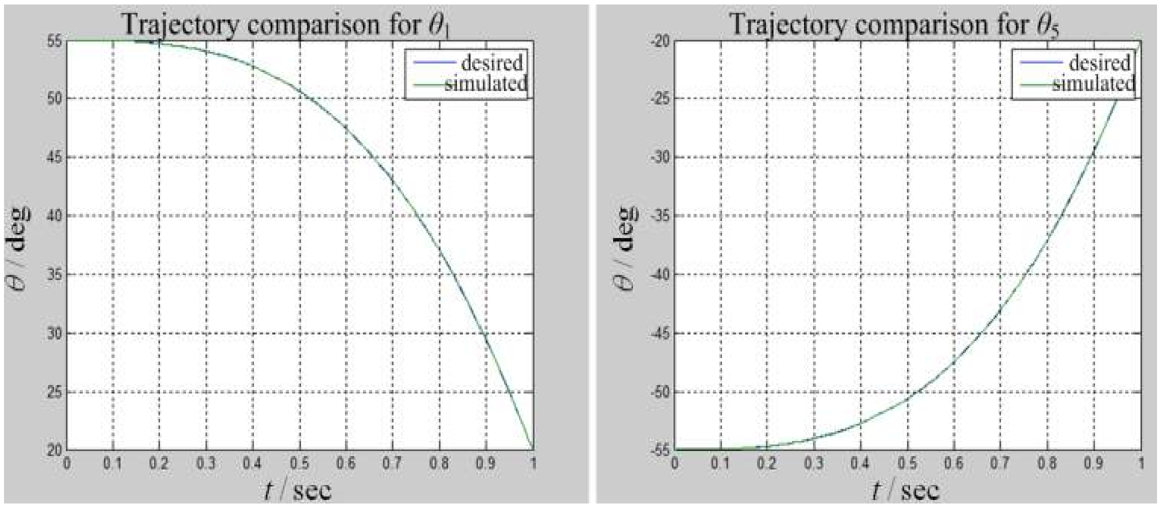


Figure 4.7: Trajectory comparison result

This model can be easily rewritten into a decoupled form $\boldsymbol{Y}''\boldsymbol{Z}'' = \boldsymbol{\tau}$, where

$$\boldsymbol{Y}'' = \begin{pmatrix}\boldsymbol{Y}' & \boldsymbol{Y}_{vf} & \boldsymbol{Y}_{cf}\end{pmatrix}; \tag{4.23}$$

$$\boldsymbol{Y}_{vf} = \begin{pmatrix}\dot{\theta}_1 & \dot{\theta}_5 & 0 & 0 \\ 0 & 0 & \dot{\theta}_1 & \dot{\theta}_5\end{pmatrix} \tag{4.24}$$

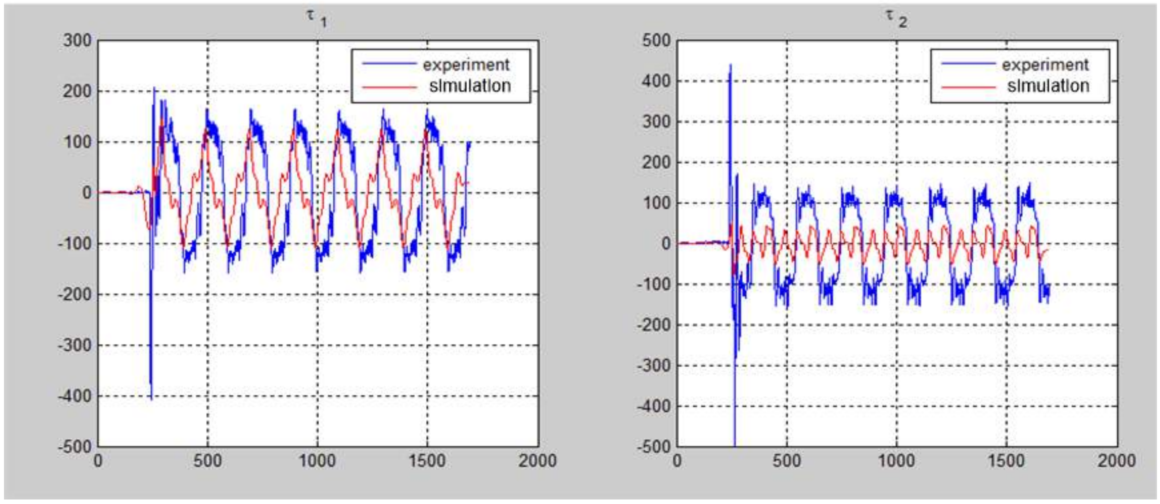


Figure 4.8: Identification without friction

and

$$\boldsymbol{Y}_{cf} = \begin{pmatrix} sign\dot{\theta}_1 & 0 & sign(\dot{\theta}_1 - \dot{\theta}_5) \\ 0 & sign\dot{\theta}_5 & sign(\dot{\theta}_5 - \dot{\theta}_1) \end{pmatrix} \tag{4.25}$$

With this friction model, the least-square regression gives a good result, as shown in Fig. 4.9 for the same validation condition as for Fig.4.8.

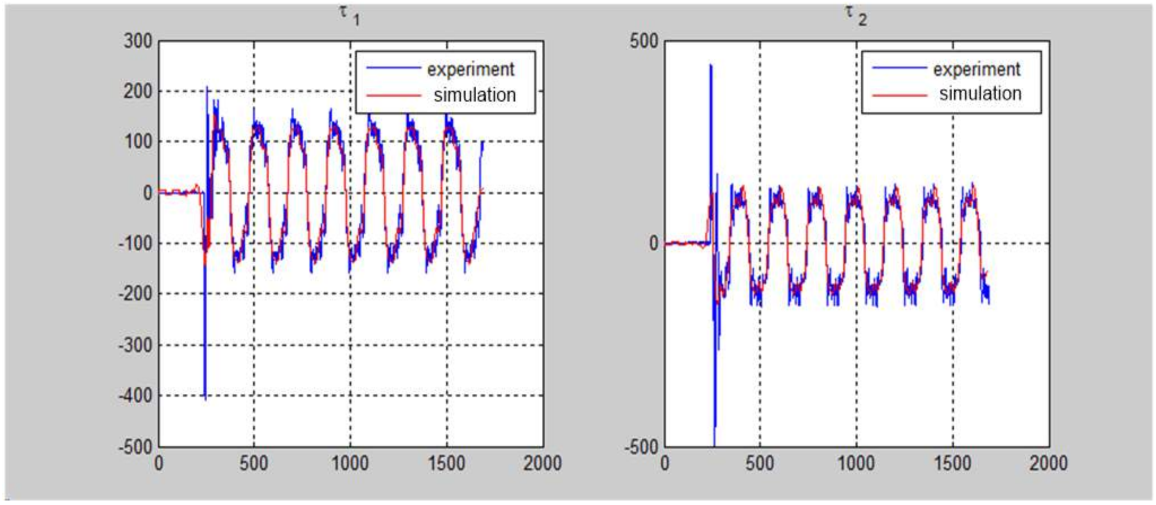


Figure 4.9: Identification with friction

In the experiment, the $\boldsymbol{Y}''$ data matrix needs to be well conditioned for good regression result. The ideal case is to use random motor angles, which is not possible for the actual robot. In order to find 'exciting' trajectories for regression[62], we need to maximize the minimum singular value of the $\boldsymbol{Y}''$ data matrix under dynamics constraints. The actual trajectory for identification experiment is chosen as shown in Fig. 4.10.

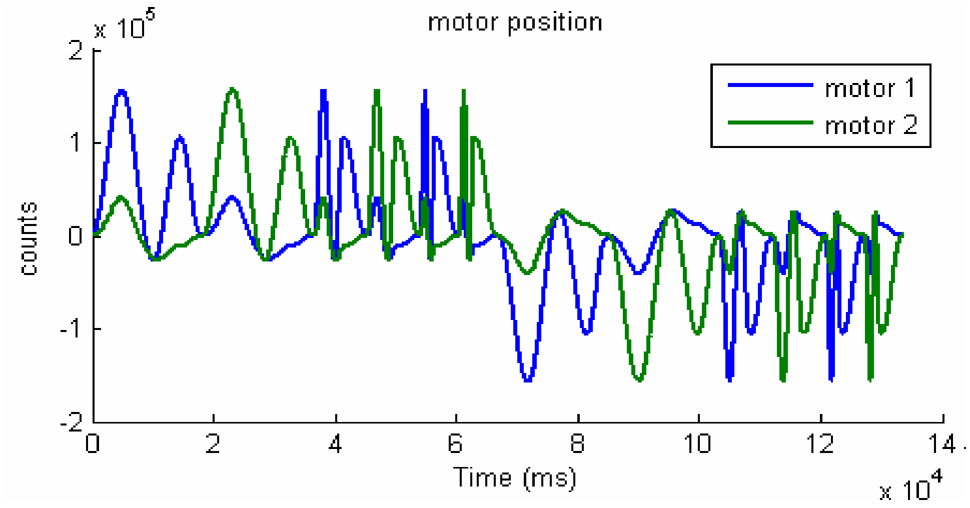


Figure 4.10: Trajectory for identification

Identification is then performed to obtain the mechanical parameters. Fig.4.11-Fig.4.13 show the validation results of identification with three different experiments. As shown in figures, the blue lines are the actual torque data, while the red lines are the computed torque using the model identified. The velocity after filtering is also shown in the upper part of the figure. Three different groups of data are chosen. The first one is a sinusoidal motion. The second one is a relatively arbitrary trajectory. The third one is similar to the data from identification, but with different amplitude. These figures indicate that the parameters identification procedure is effective, especially when the speed is not close to zero.

4.4 Feedforward Control

The feedforward controller for torque compensation is applied to the dual blade wafer handling robot using the result of dynamic parameters identification.

Experimental Configuration

The dual blade wafer handling robot adopts a tri-loop feedback control structure to stabilize the system for basic position servo. Torque and velocity feedforward are added for better

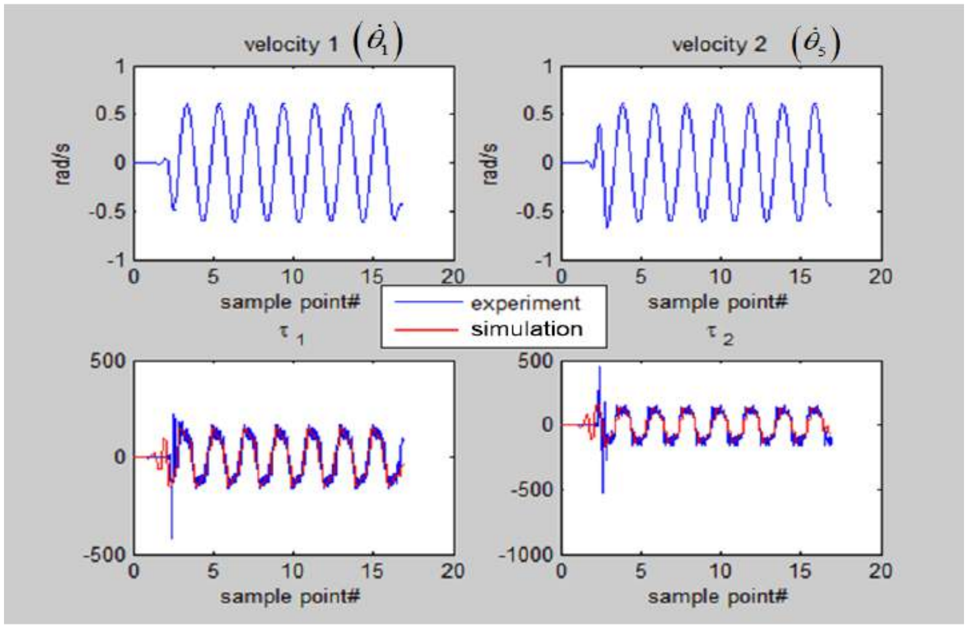


Figure 4.11: Verification of the ID- data #1

performance. The feedforward torque is computed offline now, and is injected to the drivers from target PC via EtherCAT communication. For safety, an Emergency Stop(E-stop) is used. A jig for mechanically adjusting the zeroing motor position is also used. The experiment configuration is shown in Fig.4.14.

In the experiment, joint angles and angular velocities are obtained by the driver feedback. Angular accelerations are obtained by differentiate velocities. Torque signal is obtained by the current feedback.

Experiment Result

Three trajectories are used for the feedforward control: two trajectories for extension and one trajectory for rotation. Each trajectory is separated into four parts. The first part is going to an initial position and the last part is going back to the zero position. The other two parts are the same, both of them include: 1) wait for start at the initial position 2) perform the desired trajectory 3) wait for start at the end position 4) go back to the initial position. The difference for these two parts is that the torque feedforward control is added only to the first desired trajectory.

The first extensional trajectory lasts for 2.69s. The end effecter moves from 304.8mm to 911.5mm in Cartesian space. The end effecter moves along a S-curve trajectory. The velocity limit, acceleration limit and jerk limit for this S-curve are: 554 mm/s, 9801 mm/s^2, 1000 mm/s^3. Figure 4.15 shows the tracking error during the experiment. In the upper

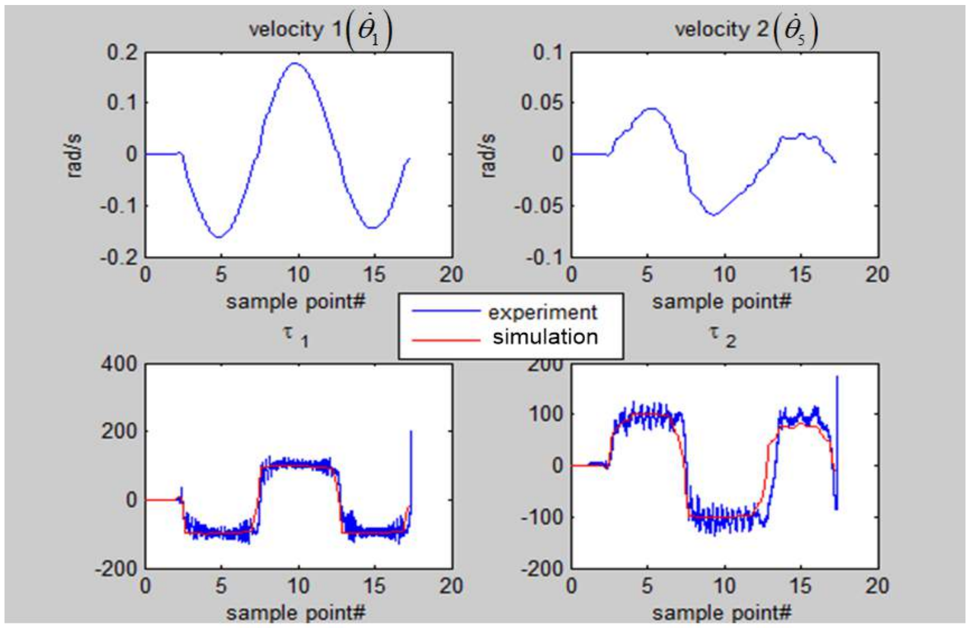


Figure 4.12: Verification of the ID- data #2

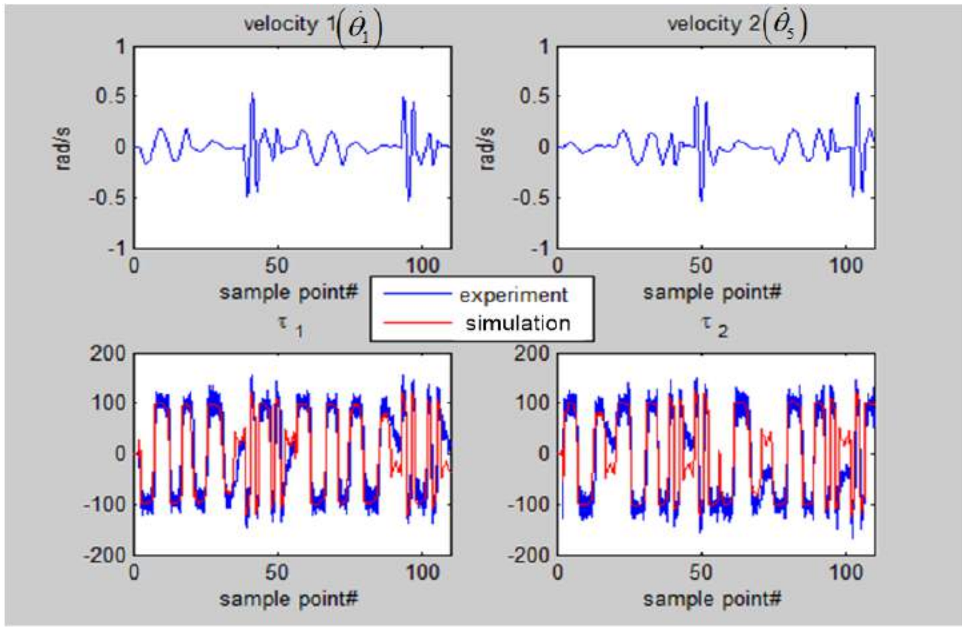


Figure 4.13: Verification of the ID- data #3

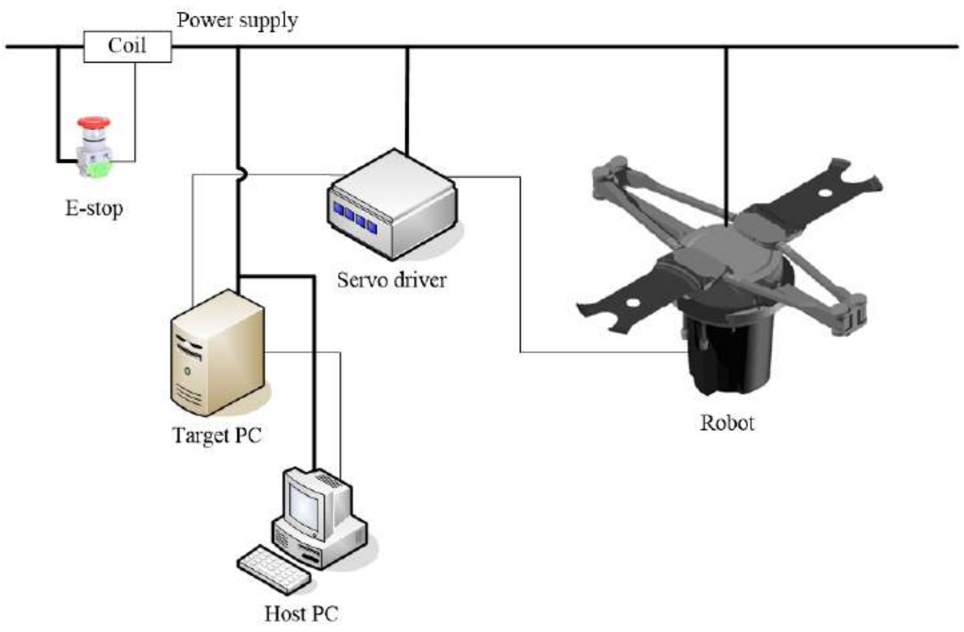


Figure 4.14: Experiment configuration

figure in Fig.4.15, the desired motor position and actual motor position are given. Two extensions are for comparison between with and without feedforward compensation. Middle figure shows the comparison result for tracking error. During the first extension (first red box in the middle figure), because the feedforward compensation is added as a torque offset in the torque commend (shown in the lower figure), the tracking error is largely reduced compared to the second extension (second red box in the middle figure).

The second extensional trajectory lasts for 1.76s, which is a faster motion for robot. The end effecter moves from 304.8mm to 911.5mm in Cartesian space. The end effecter moves along a S-curve trajectory. The velocity limit, acceleration limit and jerk limit for this S-curve are: 554 mm/s, 9801 mm/s^2, 5000 mm/s^3. Figure 4.16 shows the tracking error during the experiment. In the upper figure in Fig.4.16, the desired motor position and actual motor position are given. Two extensions are for comparison between with and without feedforward compensation. Middle figure shows the comparison result for tracking error. During the first extension (first red box in the middle figure), because the feedforward compensation is added as a torque offset in the torque commend (shown in the lower figure), the tracking error is largely reduced compared to the second extension (second red box in the middle figure).

The rotational trajectory rotates from 0 to 144 deg. Figure 4.17 shows the tracking error during the experiment. In the upper figure in Fig.4.17, the desired motor position and actual motor position are given. Two rotations are for comparison between with and without feedforward compensation. Middle figure shows the comparison result for tracking

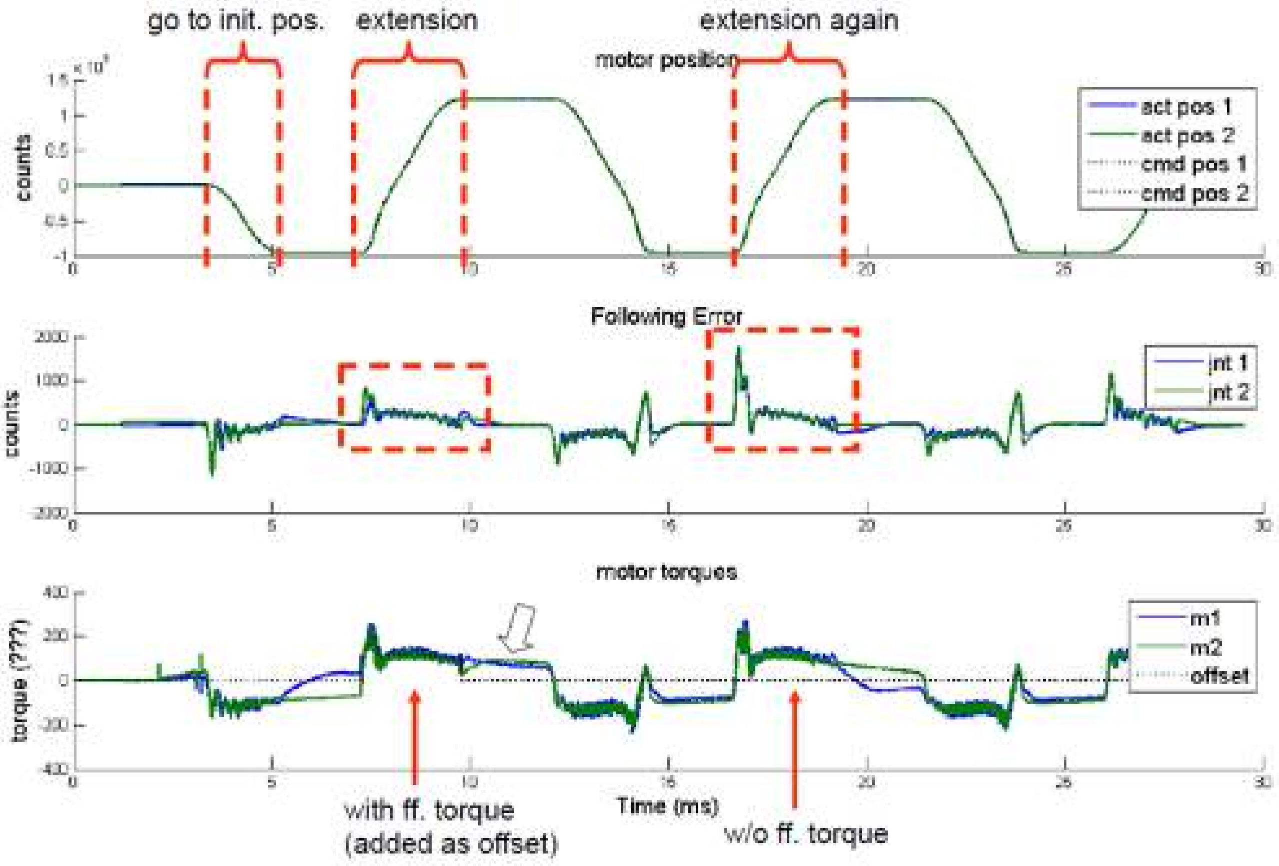


Figure 4.15: Torque feedforward control- extension 1

error. During the first rotation (first red box in the middle figure), because the feedforward compensation is added as a torque offset in the torque commend (shown in the lower figure), the large spike of tracking error is removed compared to the second rotation (second red box in the middle figure).

4.5 Chapter Summery

In this Chapter, the precision motion control for parallel robots performing accurate tracking in industrial applications is studied. The major challenge lies in the dynamic modeling of the parallel manipulators which have closed-loop chain structure. Due to the complex non-linear coupled multi-body dynamics, a model-based controller is thus hard to design.

One example is a dual blade 8-link wafer handling robot which transfers wafers among chambers in the IC machine. The dynamics modeling and dynamic parameter identification of the dual-blade wafer handling robot is studied. Both the explicit kinematics and dynamic

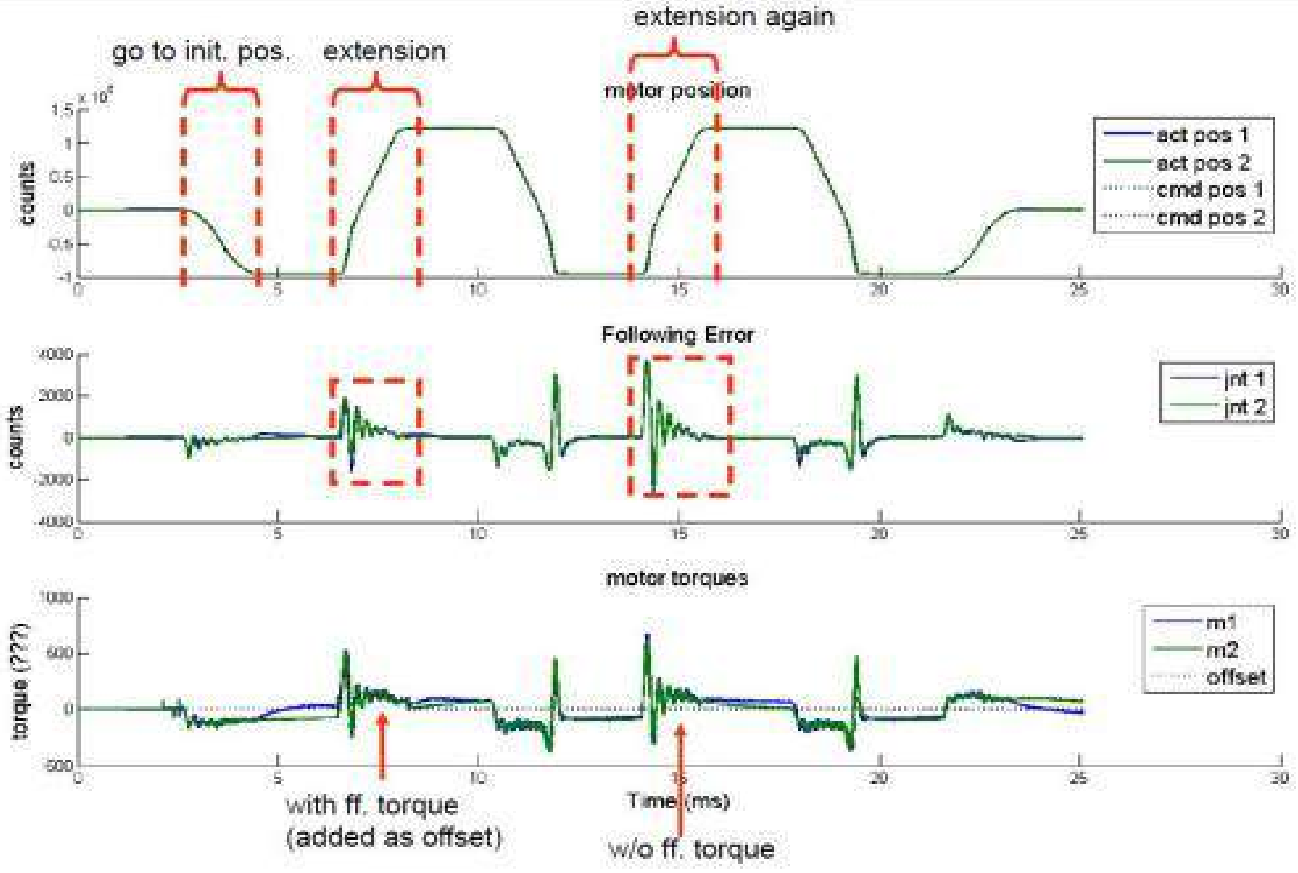


Figure 4.16: Torque feedforward control- extension 2

model for this parallel robot have been derived using Lagrangian method. A decoupled dynamic model is also formulated for dynamic identification. A data-driven approach is proposed for a refined decoupling which is in a favorable form for regression.

During identification, experiments on an actual robot indicates the major influence of friction. Viscous friction and Coulomb friction are then considered to improve the identification accuracy. The identified model is utilized in a model based controller: a feedforward controller. Multi experiments are performed to test the effectiveness of the feedforward controller. Tracking error is greatly reduced with this model based approach.

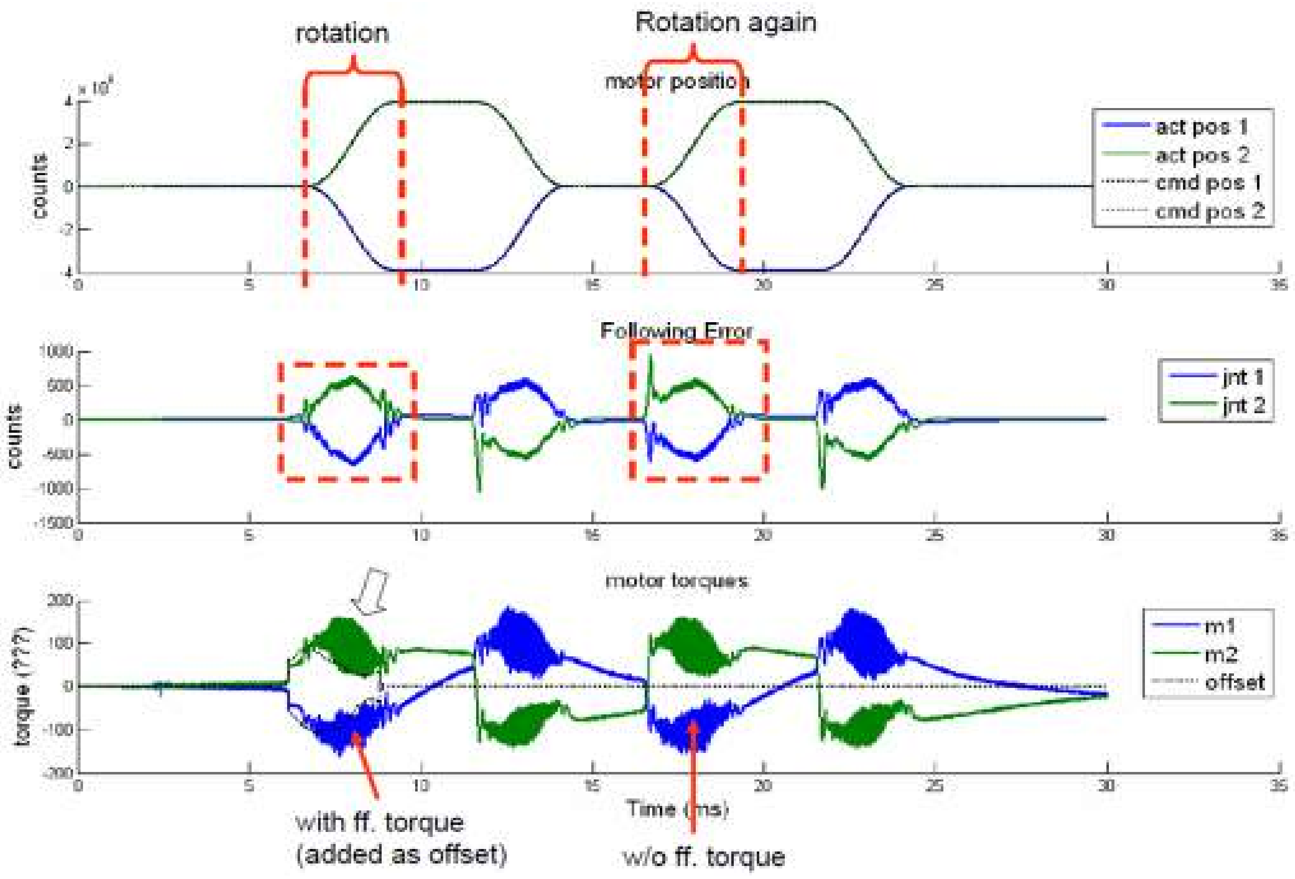


Figure 4.17: Torque feedforward control- rotation

Chapter 5

Optimal Gain Tuning

5.1 Introduction

This Chapter is a continuation of Chapter 4: precision motion control for parallel manipulators. For the motion control of industrial robot, the end-effector performance is always of the ultimate interest. However, only motor-side encoders are available in most cases. Thus controller gains, such as PID gains, are usually tuned based on motor-side performance, which is not desirable. In silicon wafer processing, with a non-optimal PID controller, the robot will exhibit considerable vibration and lots of particles will be produced. Because very few particle can cause wafer damage and tool downtime, a major concern of robot performance is the vibration on the load side. Thus vibration suppression is the main motivation of optimal gain tuning.

In order to obtain a better performance, in this Chapter, controller gains are proposed to be tuned based on end-effector performance with load-side sensing. Traditional tuning methodologies for the control gains include empirical methods such as Ziegler-Nichols [6] and analytical methods such as shaping of frequency response [7]. However, these methods are not very effective in practice. Their effectiveness for nonlinear systems such as robots and for reduction of vibration is not conclusive. To address this problem, an optimization based method for gain tuning is developed in this Chapter.

Optimization technique is ideally suited for gain tuning first because it is not a model based approach. In fact, the optimal gain tuning is an experiment based approach, which makes it practical for industrial applications. Second, the performance of optimal gain tuning is guaranteed by modern optimization theories [51]. These two features make the optimal gain tuning an attractive method for controller tuning of industrial robots.

In this Chapter, an experiment based guided gain search is proposed with direct end-effector sensing. An optimization scheme is utilized to guide the searching direction, which can provide fast gain tuning and in the mean time guarantee optimality. The rest of the Chapter is organized as follows: in Section 5.2, the tuning method is given in detail, including the description of the controller, and the optimization based tuning with direct end-effector sensing. Section 5.3 gives the result of the gain tuning, and Section 5.4 concludes the Chapter.

5.2 PID optimal gain tuning

Due to the importance of the vibration suppression on the load side, the optimal gain tuning procedure is based on the direct measurement of the load side acceleration. An accelerometer is mounted on the wafer center for vibration monitoring, as shown in Fig.5.1. Acceleration from both the "side-to-side" direction and the radial direction are measured.

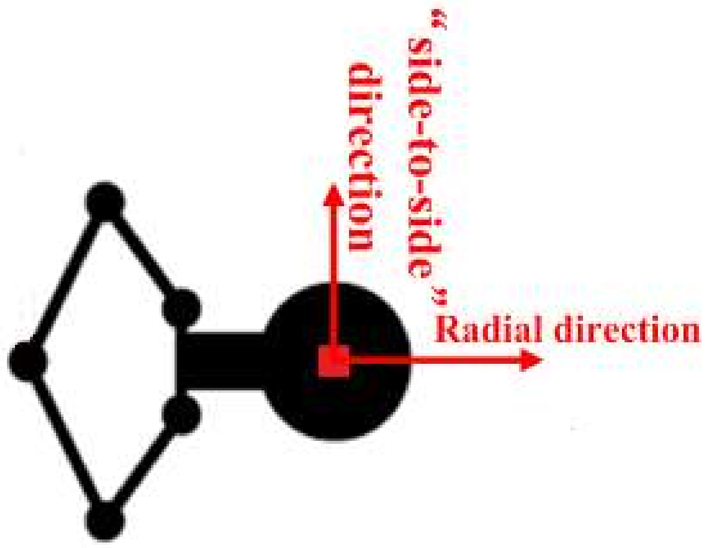


Figure 5.1: Load side wafer acceleration sensing

In the optimal gain tuning, three controller gains to be tuned are: proportional gain in the position loop (Pp), and proportional and integral gain in the velocity loop (Vp,Vi). The control loop is shown in Fig.5.2. It should be noted that the variables to be tuned are called design variables, and the space formed by design variables is called the design space.

A starting set of gains (P_p, V_p, V_i) should be given before the optimal gain tuning starts. A set of gains (P_p, V_p, V_i) is called a "point" in this Chapter. This is because the space formed by tuning variables is called the design space, and a set of gains can be viewed as a point in the design space. Usually, the start point is obtained manually, and it largely depends on the experience of engineers. Because the non-convexity nature of the gain tuning, the start point is assumed to be close to the optimal point to ensure the convergence to the global minimum. The optimal point means the point of the best performance, i.e. the global minimum. For each point, a score is given based on its performance, such as position tracking error from

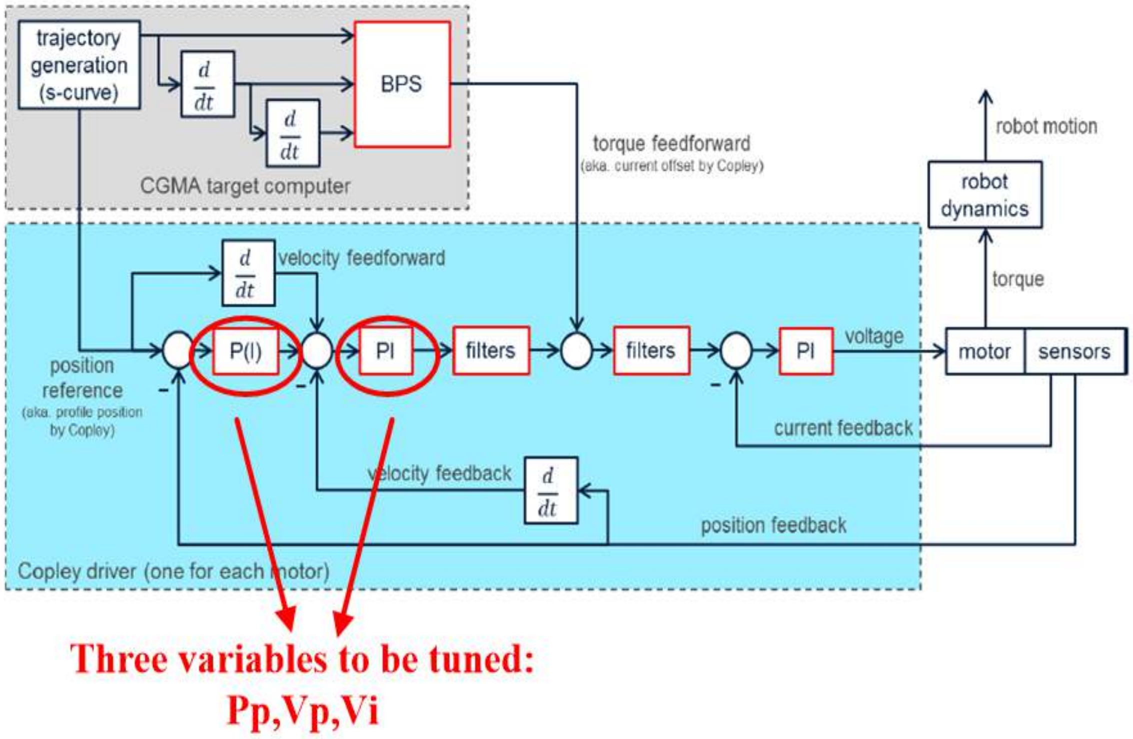


Figure 5.2: Control loop

motor side, average oscillation amplitude, acceleration from load side, etc, and it is graded in a way that the smaller the score, the better the performance. The optimal gain tuning is to search for the minimum score in the design space. There are two major steps of the optimal gain tuning: Gradient Estimation and Line Search.

Gradient Estimation

To find the optimal point in the design space, first we need to determine a searching direction in the design space from the start point. Thus we need to estimate the gradient of the start point based on performances of points that near the start point. This will give us the steepest descent direction of the performance score, as shown in Fig.5.3.

The points near the start point should be uniformly distributed to guarantee the estimation accuracy. Because the sobol sequence [63] can distribute points in the design space more uniformly than other distribution methods, it is used to perturb around the start point. 20 perturbation points should be enough with respect to the range of design space. After the distribution finished, 21 experiments are performed and scores based on their performances are generated. The score of each performance is a weighted sum of selected performance indices:

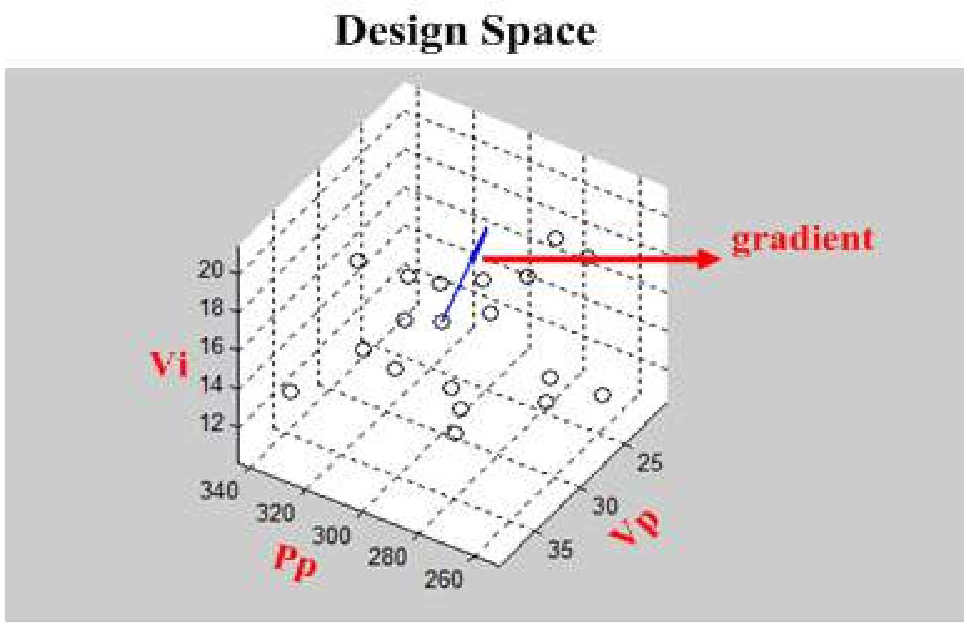


Figure 5.3: Gradient estimation in design space

a) Position errors(following errors) for both axes

b) Velocity errors for both axes

c) Average oscillation amplitudes for both axes

d) Maximum of average oscillation amplitudes of two axes

e) Maximum side to side oscillation amplitude from load side

f) Maximum radial oscillation amplitude from load side

The position error is related to the trajectory tracking error of both motors, and velocity error refers to the velocity tracking error of both motors. In the optimal gain tuning, we use the 1-norm of tracking error as the Position error index as it can give a value in the sense of average following error. Index c, Average oscillation amplitudes for both axes are measurements of oscillation amplitudes of two motors. It gives a reference on how far the tracking is from smooth tracking. This is a major index which indicates the magnitude of vibration. Index d, Maximum of average oscillation amplitudes of two axes is just to pick the larger value from index c (which has two values for two axes). Unlike other performance indices, e and f are measurements from the load side. The load side information is directly used when evaluate the performance of a point. Because the load side performance is the most important performance that we care about, the inclusion of load side information makes the optimal gain tuning more relevant. The weights are carefully chosen for all the performance indices:

- Position Error: 1.5×10^{-3}
- Velocity Error: 4.44×10^{-19}
- Average of Oscillation for both axes: 0.2×10^{-10}
- Maximum of average of oscillation of two axes: 0.2
- Maximum side to side oscillation from load side: 25
- Maximum radial oscillation from load side: 0.1

Using the sets of gains (noted as x, row vector, 21 sets) and their scores (noted as s, scalars, 21 values), we can determine the gradient of the start point by pseudo-inverse, which gives a least-square estimate:

$$G = \begin{bmatrix} \Delta s_1 \\ \vdots \\ \Delta s_{20} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_{20} \end{bmatrix}^{\dagger} \tag{5.1}$$

Line Search

Line search is to search along the steepest descent direction obtained by gradient estimation in the design space, as shown in Fig.5.4. The line is generated in a way that when it hits the boundary, it will turn direction until it reaches another boundary. The boundary of the design space is determined empirically.

First, 1000 points are distributed on this line. The number 1000 is chosen because it is dense enough to see the small performance change due to the change of gains. The optimal point will be obtained among these 1000 points. To make it efficient, a simple bisection search method is used to search along the line, and with this method, we are able to find the optimal point in less than 20 experiments.

Usually, Steps 1 and 2 (gradient estimation and line search) should be repeated. To start with, first round gradient estimation and line search should be performed. Then use the optimal point from the first round as the new start point to repeat the two steps until a satisfactory optimal point is obtained. However, in our experiment, there is no need to do the second round, because the score around the first optimal point are similar. This makes the process of line search even shorter.

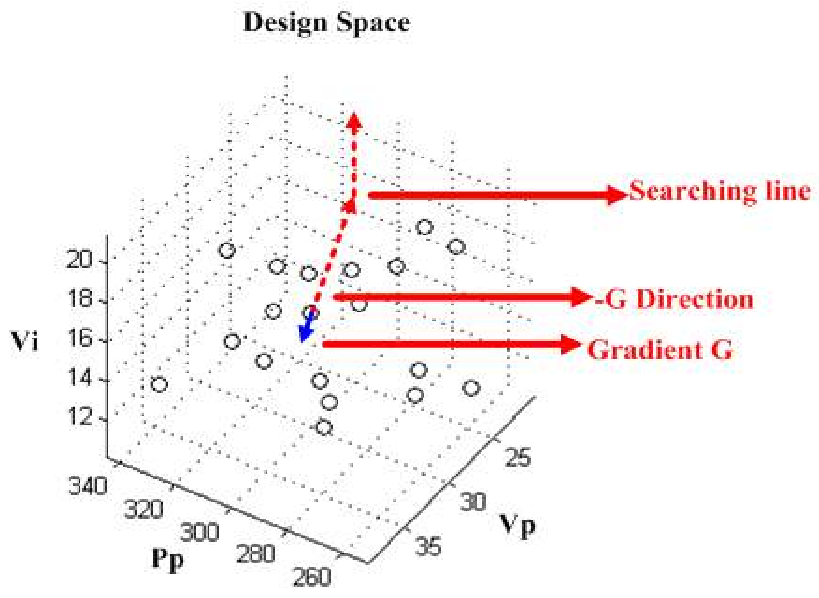
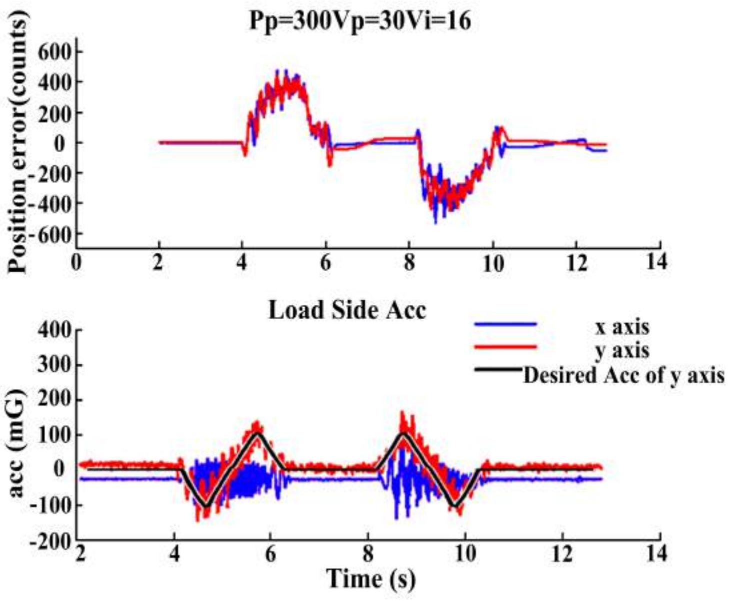


Figure 5.4: Line search

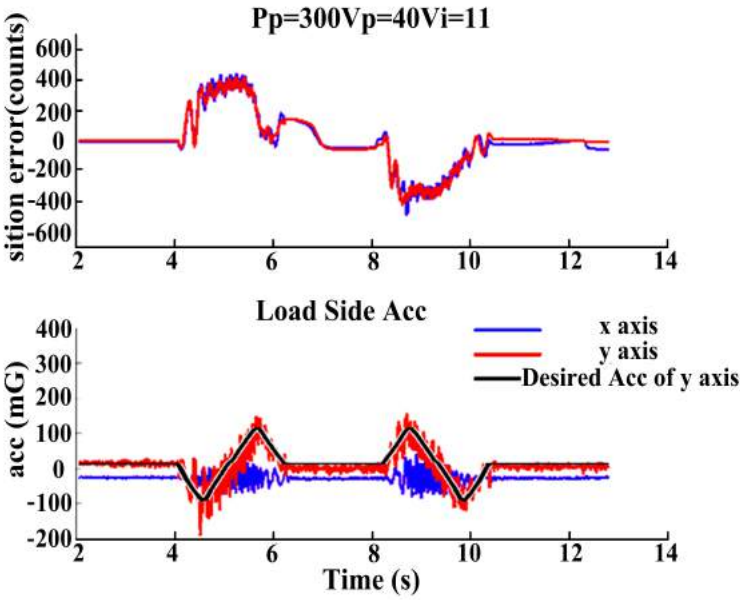
5.3 Experimental Result

It was confirmed by experiments that the optimal set of gains lowered the performance scores compared to the gains at the start point, which means a better performance and less vibration. Figure 5.5 shows the performances of both the start point and the optimal point. In each subfigure of Fig.5.5, the upper plot is the position tracking error of the robot's two motors, and the lower plot is the desired and actual acceleration (not the acceleration error) of the wafer center during the experiment. x is the "side-to-side" direction and y is the radial direction. The desired acceleration in x direction should be zero (which is not shown in the plot), and the desired acceleration in y direction is given in the lower plot of each subfigure. Because the vibration of the wafer center is of the utmost interest, the acceleration plots should be compared. The result is that, with the PID controller gains of the start point, the robot end effector suffers a lot vibration, while with the PID controller gains of the optimal point, the robot vibration of the wafer center is largely reduced in both directions.

Table 5.1 compares the scores of the most important performance indices at the start point and the end point of tuning process. It should be noted that all the scores were improved (lowered) after optimization.



a) Performance of the start point



b) Performance of the optimal point

Figure 5.5: Gain tuning result

Table 5.1: Comparison of performances of the start and optimal point

	Start Point	End Point
Max PosErr(count)	482.9/540.1	441.3/498.3
Max Load side Acc of x(mG)	115	68
Average of Oscillation(count)	45236	30263
Total Score	13931	10557

5.4 Chapter Summery

This Chapter continues the study on precision motion control for parallel manipulators. For the motion control of industrial robot, the end-effector performance is always of the ultimate interest. However, in most cases, robots are only equipped with motor side encoders, and the performance on the load side cannot be satisfied with controller gains tuned based on motor side performance. In silicon wafer processing, with a non-optimal PID controller, the robot will encounter considerable vibration and lots of particles will be produced. Because very few particle can cause wafer damage and tool downtime, a major concern of robot performance is the vibration on the load side.

In this Chapter, an experiment based guided gain search is proposed with direct end-effector sensing for vibration suppression. An optimization scheme is utilized to guide the searching direction with two steps: gradient estimation and line search. It is demonstrated by experiments that the optimal gain tuning improves the performance by reducing the vibrations on the load side.

Chapter 6

Conclusions and Future Work

This dissertation presented a series of statistical learning methods to improve the overall performance for industrial robots . Four aspects have been studied in detail including sensor fusion for robust visual servo, optimal trajectory planning, intelligent system identification for feedforward compensation, and optimal gain tuning.

Chapter 2 presented several statistical learning approaches to overcome the difficulties for fast, accurate and robust visual servo. The work is intended to improve the performance of an industrial manipulator for accurate glass piece handling using real-time visual feedback. Visual servo is favorable in this scenario since it can achieve accuracy based on the robot encoder resolution rather than the robot absolute accuracy by closing the control loop visually. Due to the hardware limitations, challenges come from several aspects, namely, object detection, Tool Center Point (TCP) calibration, visual feedback with large latency and low sampling rate due to limited quality of cameras, and system uncertainties. To address all the problems, first, a glass piece and slots are both detected robustly with an Eye-to-Hand camera with a statistical learning method: non-linear geometrical regression. Then, a closed loop visual servo controller is implemented to guide the glass piece into the slot. Instead of using point features of the object, line features are for the first time adopted for accurate placing requirement. To compensate for the cameras' very low frame rate and parameter uncertainties, a statistical learning method, which is a dual-rate Unscented Kalman Filter with dual-estimation is implemented. Experimental results show the effectiveness of the placing performance. Future work can be conducted for further increasing the placing efficiency. On the one hand, hardware, such as vision sensors can be improved for a better resolution and higher sampling rate. Furthermore, more unknown system parameters can be adaptively identified with intelligent adaptive estimator, which can address the problem of system uncertainties.

Chapter 3 attempted to solve the trajectory planning problem for industrial manipulators with operational constraints with combined statistical learning method and optimization strategy. One statistical approach for path planning is the sampling based method, which searches through the robot $\mathcal{C}$ space with a sampling scheme. However, it does not consider path smoothness and non-linear differential constraints. On the other hand, optimization based method, which is favorable for trajectory smoothness and optimal condition, requires heavy computational loads or only works for a predetermined duration. By combining the two approaches: sampling and optimization based method, the trajectory planning can be solved not only fast but also with certain optimality. One example given in this Chapter is a wafer handling robot working in a very limited workspace. The generated trajectory must be a safe one without colliding with any obstacles and is optimized with respect to the total execution time for working efficiency. For the obstacle avoidance, first a roadmap describing the connectivity of the robot's obstacle-free workspace is constructed by a probabilistic roadmap (PRM) method. The sampling scheme is to use multiple samplers each with a specific strength and update sampler adaptively. Then a shortest piece-wise linear path is searched from the roadmap for a given query, i.e, the initial and goal configuration of the robot. With the path searched from the roadmap, the trajectory is first smoothed by a cubic B-spline, which can guarantee the trajectory with a desired non-colliding shape and in the mean parameterized it with temporal information, which can be optimized later. B-spline also has nice properties that all kinematic constraints can be formulated easily. The problem is formulated as an optimization problem and solved by an "interior point" method and the results on a 3-axis wafer handling robot show the effectiveness of the proposed method. Future work will be devoted to consider trajectory outside the current work space, as well as the optimal control problem which involves the robot dynamics. Also, for a real-time application, it is desirable to reduce the computational time with more efficient computing structure.

In Chapter 4 and 5, statistical learning methods for the precision motion control for parallel robots performing accurate tracking in industrial applications were studied. On the one hand, the dynamic modeling of the parallel manipulators which have closed-loop chain structure is very challenging. Due to the complex non-linear coupled multi-body dynamics, a model-based controller for precision motion control is thus hard to design. On the other hand, for the motion control of industrial robots, the end-effector performance is always of the ultimate interest. However, in most cases, robots are only equipped with motor side encoders, and the performance on the load side cannot be guaranteed with controller gains tuned based on motor side performance. Chapter 4 designed a feedforward controller for a dual blade 8-link parallel wafer handling robot using model based approach. Both the explicit kinematic and dynamic model were derived using Lagrangian method. The dynamic parameter identification were studied with a decoupled dynamic model. A data-driven approach was proposed for a refined decoupling which is in a favorable form for regression. Then a linear regression was performed using data generated by experiments on the robot with designed trajectory. Tracking error is greatly reduced with this model based approach and verified by experiments. In Chapter 5, an experiment based guided gain search was proposed with direct end-effector sensing for vibration suppression. An optimization scheme was utilized to guide the searching direction with experiments and data learning. It was demonstrated by experiments that the optimal gain tuning improves the performance by reducing the vibrations on the load side. Future work can focus on the joint dynamics, and compliance induced by the magnetic coupler which is used for isolating the robot from atmosphere side. Also, the tuning method need to be expanded to other robots and improved to be a general gain tuning solution.

Bibliography

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[2] Xiaowen Yu, Thomas Baker, Yu Zhao, and Masayoshi Tomizuka. Fast and precise glass handling using visual servo with unscented kalman filter dual estimation. *Journal of Dynamic Systems, Measurement, and Control*, 140(4):041008, 2018.

[3] Xiaowen Yu, Thomas Baker, Yu Zhao, and Masayoshi Tomizuka. Visual servo for fast glass handling by industrial robot with large sensor latency and low sampling rate. *IFAC-PapersOnLine*, 50(1):4594–4601, 2017.

[4] Xiaowen Yu, Yu Zhao, Cong Wang, and Masayoshi Tomizuka. Trajectory planning for robot manipulators considering kinematic constraints using probabilistic roadmap approach. *Journal of Dynamic Systems, Measurement, and Control*, 139(2):021001, 2017.

[5] Xiaowen Yu, Cong Wang, Yu Zhao, and Masayoshi Tomizuka. Dynamics modeling and identification of a dual-blade wafer handling robot. In *ASME 2013 Dynamic Systems and Control Conference*, pages V003T39A004–V003T39A004. American Society of Mechanical Engineers, 2013.

[6] Aidan O'Dwyer. *Handbook of PI and PID controller tuning rules*. Imperial college press, 2009.

[7] Katsuhiko Ogata and Yanjuan Yang. *Modern control engineering*, volume 4. Prentice hall India, 2002.

[8] Xiaowen Yu, Cong Wang, Yu Zhao, and Masayoshi Tomizuka. Controller design and optimal tuning of a wafer handling robot. In *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, pages 640–646. IEEE, 2015.

[9] McKinsey Quarterly. Where machines could replace humans–and where they can't (yet). `http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/`, 2016. Accessed: 2016-7.

[10] Jim Shimano. Visual servoing–closing the position loop with vision.

[11] Kenton McHenry, Jean Ponce, and David Forsyth. Finding glass. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 973–979. IEEE, 2005.

[12] Ivo Ihrke, Kiriakos N Kutulakos, Hendrik PA Lensch, Marcus Magnor, and Wolfgang Heidrich. State of the art in transparent and specular object reconstruction. In *EUROGRAPHICS 2008 STAR–STATE OF THE ART REPORT*. Citeseer, 2008.

[13] Henri Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems, 2011.

[14] ABB. Programming and tesing-tools. `http://developercenter.robotstudio.com/BlobProxy/manuals/IRC5FlexPendantOpManual/doc99.html`, 2016. Accessed: 2016.

[15] Johan Hallenberg. Robot tool center point calibration using computer vision, 2007.

[16] Xiaowen Yu, Thomas Baker, Yu Zhao, and Masayoshi Tomizuka. Robot tool calibration in precise glass handling. In *ASME 2017 Dynamic Systems and Control Conference*, pages V002T16A001–V002T16A001. American Society of Mechanical Engineers, 2017.

[17] Cong Wang, Chung-Yen Lin, and Masayoshi Tomizuka. Statistical learning algorithms to compensate slow visual feedback for industrial robots. *Journal of Dynamic Systems, Measurement, and Control*, 137(3):031011, 2015.

[18] Keisuke Fujii. Extended kalman filter. *Refernce Manual*, 2013.

[19] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000.

[20] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997.

[21] Nikolaos P Papanikolopoulos, Bradley J Nelson, and Pradeep K Khosla. Six degree-of-freedom hand/eye visual tracking with uncertain parameters. *IEEE Transactions on Robotics and Automation*, 11(5):725–732, 1995.

[22] Hesheng Wang, Yun-Hui Liu, and Dongxiang Zhou. Adaptive visual servoing using point and line features with an uncalibrated eye-in-hand camera. *IEEE Transactions on Robotics*, 24(4):843–857, 2008.

[23] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011.

[24] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[25] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[26] Nicolas Andreff, Bernard Espiau, and Radu Horaud. Visual servoing from lines. *The International Journal of Robotics Research*, 21(8):679–699, 2002.

[27] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[28] Gaston H Gonnet. *Handbook of algorithms and data structures*. Addison-Wesley, 1984.

[29] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988.

[30] Bob Fisher. projective transformation. `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/BEARDSLEY/node3.html`, 1997. Accessed: 1997-11-7.

[31] Xiaowen Yu. Auto glass handling by robot. `https://youtu.be/E4K5p8eGl1Y`, 2017. Accessed: 2017-2-23.

[32] Yong K Hwang, Peter A Watterberg, Pang Chen, Christopher L Lewis, et al. General techniques for constrained motion planning. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1997.

[33] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[34] Vanni Zanotto, Alessandro Gasparetto, Albano Lanzutti, Paolo Boscariol, and Renato Vidoni. Experimental validation of minimum time-jerk algorithms for industrial robots. *Journal of Intelligent & Robotic Systems*, 64(2):197–219, 2011.

[35] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[36] David Hsu, Gildardo Sánchez-Ante, and Zheng Sun. Hybrid prm sampling with a cost-sensitive adaptive strategy. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3874–3880. IEEE, 2005.

[37] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[38] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.

[39] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

[40] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[41] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807. IEEE, 1993.

[42] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 489–494. IEEE, 2009.

[43] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574. IEEE, 2011.

[44] Kris Hauser and Victor Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2493–2498. IEEE, 2010.

[45] Marko Lepetič, Gregor Klančar, Igor Škrjanc, Drago Matko, and Boštjan Potočnik. Time optimal path planning considering acceleration limits. *Robotics and Autonomous Systems*, 45(3-4):199–210, 2003.

[46] Kevin Judd and Timothy McLain. Spline based path planning for unmanned air vehicles. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4238, 2001.

[47] C-H Wang and J-G Horng. Constrained minimum-time path planning for robot manipulators via virtual knots of the cubic b-spline functions. *IEEE transactions on automatic control*, 35(5):573–577, 1990.

[48] Yao-Chon Chen. Solving robot trajectory planning problems with uniform cubic b-splines. *Optimal Control Applications and Methods*, 12(4):247–262, 1991.

[49] A Gasparetto and V Zanotto. A new method for smooth trajectory planning of robot manipulators. *Mechanism and machine theory*, 42(4):455–471, 2007.

[50] Eric Larsen, Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. Technical report, Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[51] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[52] Fathi Ghorbel. Modeling and pd control of closed-chain mechanical systems. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 540–542. IEEE, 1995.

[53] Jean-Pierre Merlet. *Parallel robots*, volume 128. Springer Science & Business Media, 2006.

[54] Luis Angel Castañeda, Alberto Luviano-Juárez, and Isaac Chairez. Robust trajectory tracking of a delta robot through adaptive active disturbance rejection control. *IEEE Transactions on Control Systems Technology*, 23(4):1387–1398, 2015.

[55] Fathi H Ghorbel, Olivier Chételat, Ruvinda Gunawardana, and Roland Longchamp. Modeling and set point control of closed-chain mechanisms: Theory and experiment. *IEEE Transactions on Control Systems Technology*, 8(5):801–815, 2000.

[56] Jens Wittenburg. *Dynamics of systems of rigid bodies*, volume 33. Springer-Verlag, 2013.

[57] John J Uicker. Dynamic behavior of spatial linkages: Part 1exact equations of motion, part 2small oscillations about equilibrium. *Journal of Engineering for Industry*, 91(1):251–265, 1969.

[58] John J Murray and Gilbert H Lovell. Dynamic modeling of closed-chain robotic manipulators and implications for trajectory control. *IEEE Transactions on Robotics and Automation*, 5(4):522–528, 1989.

[59] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[60] H Mayeda, K Osuka, and A Kangawa. A new identification method for serial manipulator arms. *IFAC Proceedings Volumes*, 17(2):2429–2434, 1984.

[61] B Till. Dynamical model of a three-link remotelydriven vacuum robot. In *Proceedings of the 2010 Applied Materials ET Conference*, 2010.

[62] Brian Armstrong. On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics. *The International journal of robotics research*, 8(6):28–48, 1989.

[63] Hongmei Chi, Peter Beerli, Deidre W Evans, and Michael Mascagni. On the scrambled sobol̆ sequence. In *International Conference on Computational Science*, pages 775–782. Springer, 2005.