Data-Driven Approaches for Sensing and Control of Robot Manipulators

by

Cong Wang

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

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor J. Karl Hedrick
Professor Laurent El Ghaoui

Fall 2014

Data-Driven Approaches for Sensing and Control of Robot Manipulators

Abstract

Data-Driven Approaches for Sensing and Control of Robot Manipulators

by

Cong Wang

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

In a sensing rich system, a large amount of data can be obtained over time and utilized to improve the performance and functionality of a robotic system. Data-driven approaches emphasize on the utilization of auxiliary sensors, sensor fusion, and data learning. Real-time control systems of robotic systems often run at kilo-Hertz sampling frequencies. New data is obtained from a variety of feedback sources every one or a few milliseconds. Auxiliary sensors provide additional feedbacks and enable sensor fusion. This dissertation presents a series of data-driven approaches to improve the sensing and control of robot manipulators from several aspects, including sensor fusion for motion sensing, statistical learning for feedback compensation, nonparametric learning control, and intelligent modeling and identification.

In regard to the limited sensing capability of conventional indirect drive-trains of industrial robots, a sensor fusion approach based on auxiliary optical and inertial sensors is introduced for direct motion sensing of robot end-effectors. The approach is especially useful to applications where high accuracy is required for end-effector performance in real-time. Meanwhile, for the scenarios where auxiliary sensor are not allowed, a statistical learning algorithms is developed for sensing compensation so that control of systems with limited feedback capability can be significantly improved. A major application of the approach is vision guidance of industrial robots. The proposed learning approach can significantly increase the visual tracking bandwidth without requiring high-speed cameras.

Besides improving the sensing capability of robots, nonparametric learning control is developed to control systems with complex dynamics. A major motivation of the approach is robotic laser and plasma cutting. Furthermore, to obtain high-fidelity models more efficiently, planning and learning algorithms are discussed for intelligent system modeling and identification. The applications of the proposed approaches range from vision guided robotic material handling to precision robotic machining. Various tests are designed to validate the proposed approaches.

to all I've lost and gained.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Real-time motion control of robotic systems usually run at kilo-Hertz sampling frequencies. In a sensing rich system, new data is obtained from a variety of feedback sources every one or a few milliseconds. Over time, a large amount of data can be obtained and utilized to improve the performance and functionality of a robotic system. Auxiliary sensors provide additional feedbacks and enable sensor fusion. Data-driven approaches emphasize on the utilization of auxiliary sensors, sensor fusion, and data learning. This dissertation presents a series of data-driven approaches to improve the sensing and control of robot manipulators from several aspects, including sensor fusion for motion sensing, statistical learning for feedback compensation, nonparametric learning control, and intelligent modeling and identification.

In regard to the limited sensing capability of typical indirect drive-trains of industrial robots, Chapter 2 presents a sensor fusion approach based on auxiliary optical and inertial sensors to directly measure the motion of a robot end-effector. 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. Accurate estimation of the end-effector position and velocity is thus difficult due to complex drive-train dynamics. An optical sensor based on position sensitive detector (PSD), referred as PSD camera, is developed for direct end-effector position sensing. PSD features high precision and fast response while being cost-effective, thus is favorable for real-time control applications. In addition, to acquire good velocity estimation, a kinematic Kalman filter (KKF) is applied to fuse the measurement from the PSD camera with that from inertial sensors mounted on the end-effector. Based on the auxiliary optical sensor, inertial sensor, and the sensor fusion scheme, the motion of the robot end-effector can be directly measured and used to monitor and control the tool performance of a robot manipulator.

In the scenarios where a favorable sensing system such as the one introduced in Chapter 2 is not available, Chapter 3 presents a statistical learning algorithm to compensate feedback path with large delay and slow sampling rate. A major application of the approach is vision guidance of industrial robots. In most current industrial applications, vision guided robots are controlled by a look-then-move method. This method cannot support many new emerging demands which require real-time vision guidance. Challenge comes from the

speed of visual feedback. Due to cost limit, industrial robot vision systems are subject to considerable latency and limited sampling rate. Section 3.2 proposes a statistical learning algorithm to compensate the latency and slow sampling of vision feedback so that real-time vision guided robot control can be realized with satisfactory performance. A statistical learning method is developed to model the pattern of target's motion adaptively. The learned model is used to recover visual measurement from latency and slow sampling. Based on the compensation algorithm, Section 3.3 discusses the controller synthesis methods to utilize the compensated feedback path in closed-loop systems. In order to serve industrial applications, the limited sensing and actuation capabilities have to be considered properly. A cascade control structure is introduced. The sensor and actuator limits are dealt with by consecutive modules in the controller respectively. In particular, the kinematic visual servoing (KVS) module is discussed in detail. It is a kinematic controller that generates reference trajectory in real-time. Sliding control is used to give a basic Jacobian-based design. Constrained optimal control is applied to address the actuator limits.

In addition to improving the sensing capability of robots, data-driven approaches can also be utilized to compensate control signals of systems with complex dynamics that is hard to model precisely. Chapter 4 presents a nonparametric learning control method and its application to precision tracking control of robot manipulators. For robotic machining tasks such as laser and plasma cutting, the required tracking performance is much more demanding than that for material handling, spot welding, and machine tending tasks. Challenges in control come from the nonlinear coupled multi-body dynamics of robot manipulators, as well as the transmission error in the geared joints. The proposed method features data-driven iterative compensation of torque and motor reference. Motor side tracking and transmission error are handled by separate learning modules in a two-part compensation structure. Depending on the specific setup of end-effector sensing, the proposed approach can utilize either timed trajectory measurement or untimed two-dimensional contour inspection. Nonparametric statistical learning is used for the compensation. Considerations on incorporating analytical models and selecting data subsets for more efficient learning are discussed.

Throughout the work on sensing and control, proper modeling has always been playing an indispensable role. In order to obtain high-fidelity models more efficiently, Chapter 5 discusses planning and learning algorithms for intelligent system modeling and identification. Section 5.2 discusses the problem of planning well conditioned trajectories for learning a class of nonlinear models such as the imaging model of a camera and the multibody dynamic model of a robot. In such model learning problems, the model parameters can be linearly decoupled from system variables in the feature space. The learning accuracy and robustness against measurement noise and unmodeled response depend heavily on the condition number of the data matrix. A new method is proposed to plan well conditioned trajectories efficiently by using low-discrepancy sequences and matrix subset selection. Section 5.3 presents an approach for fast modeling and identification of robot dynamics. By using a data-driven machine learning approach, the process is simplified considerably from the conventional analytical method.

Chapter 2

Sensor Fusion for Direct End-effector Sensing

2.1 Introduction

In many applications of industrial robots, the primary objective is to control the end-effector to accurately track desired trajectories or quickly move to the target positions quickly and accurately. Accurate information of the end-effector position and velocity is important to achieve this objective. Joints of typical industrial robots are driven by motors with gear reducers, and are equipped with motor-side encoders only. Accurate estimation of the end-effector position and velocity is thus difficult due to flexibility, friction, and backlash in the gear mechanism. To overcome this difficulty, the idea of adopting end-effector sensors has been suggested [1]. Specifically, the capability of measuring the end-effector motion directly without physical contact is desirable. In addition, the application of inertial sensors to the end-effector sensing can provide supplemental measurements to achieve good velocity estimation [2].

Desirable features for end-effector position sensing are *non-contact* and *direct*. *Non-contact* sensing excludes any physical contact between the target and the sensor that may intertwine with the workpieces and the tools during the actual robot operation. *Direct* sensing means that the target position information should be acquired directly instead of being inferred based on the robot model. Candidate technologies range from laser trackers to ultrasonic range finders [3, 4, 5, 6]. Despite the broad variety, candidates for non-contact direct position measurement share two common features: a) all methods use light, sound, or radio waves; b) all methods use waves in one of the three ways, i.e., interferometry, time-of-flight (TOF), or projection.

Generally, the measurement precision is limited by the wavelength. For a desired precision at $0.1mm$ level in industrial robot applications, light waves will be necessary. Among the three ways to utilize waves, interferometry can achieve extremely high precision up to the nano scale, but is also expensive. TOF is most attractive in terms of cost, but can hardly give

precision beyond the centimeter level. In terms of cost and precision, projection provides the best balance. A straightforward choice combining light wave and projection may be vision camera in view that the advances of image processing algorithms can make vision systems capable of handling highly intelligent tasks. The slow response and large time latency, however, make it challenge to utilize vision cameras in the feedback loop for real-time motion control. This motivates to find alternatives for image sensors, featuring faster response and higher sampling rate while being cost-effective. Position sensitive detector (PSD) is such an option. In Section 2.2, a camera-like position sensing device, referred as PSD camera, is proposed. It senses the position of infrared markers attached to targets such as the robot end-effector with promising accuracy and precision. It also provides a much faster response than typical vision cameras, and can be sampled at a much higher sampling rate.

The position measurement from the PSD camera alone is not sufficient to provide good velocity information. To overcome this limitation, additional sensors (e.g., inertial sensors) can be installed on the robot end-effector. Section 2.3 introduces a sensor fusion scheme to obtain good velocity estimation by fusing the measurement from the PSD camera with that from the inertial sensors. The kinematic Kalman filter (KKF), a Kalman filter based on kinematic model, is a favorable option. It avoids using complex dynamic models and does not require parameters governing the system dynamics. KKF was originally applied to one-dimensional positioning systems, where the measurements from accelerometer and encoder were fused to provide a superior velocity estimate [7, 8, 9]. Recently it has been extended to the three-dimensional case, where the angular velocity measurement from Gyroscope has been utilized to capture the rotational motion [2].

2.2 PSD Camera for End-Effector Position Sensing

Position Sensitive Detector

Position sensitive detector (PSD), unlike CCD/CMOS image sensors which sense input light in image form, senses only the position of a light spot. There are generally two types of PSD, quadrant detector and lateral effect detector. The latter is chosen due to its superiority on effective sensing area. As shown in Fig. 2.1, the sensing plane of a two-dimensional lateral effect PSD consists of one single piece of photodiode with four anodes and a common cathode. When illuminated, a current generated by photo-electro effect will go through the anodes, with each channel's magnitude proportional to the magnitude of input illumination and the distance between the illuminated area and the anode. The relationship between the current magnitudes and the illumination position is

$$\frac{2x}{L} = \frac{I_A + I_D - I_B - I_C}{I_A + I_B + I_C + I_D}, \frac{2y}{L} = \frac{I_C + I_D - I_A - I_B}{I_A + I_B + I_C + I_D} \tag{2.1}$$

where $I_\bullet$ is the current through the corresponding anode. x and y are the illumination position on the sensing plane. L is the side length of the effective sensing area.

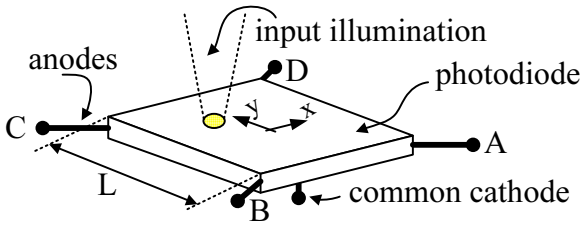


Figure 2.1: Two-dimensional lateral effect PSD

PSD has a long history of application to laser beam alignment. It has recently been adopted for spatial position sensing in entertainment applications. A successful example is the Wii game system by Nintendo, which equips a PSD at the front of the handheld controller. The illumination from the infrared LED lights located on the Sensor Bar of the Wii system is focused onto the sensing plane of the PSD by a pinhole structure. By sensing the position of the input illumination, the spatial position of the handheld controller can be determined. Inspired by the Wii system, PSD can be used in industrial applications for position sensing by focusing the light from the markers attached to the target onto the sensing plane of PSD with a lens. This forms a PSD camera (Fig. 2.2). The marker used can either be a retro-reflective one which reflects illumination from a light source, or an active beacon which emits light diffusively. Multiple markers can be sensed by a single PSD camera using multiplexing techniques.

Several advantages make PSD suitable for real-time feedback applications. First, it can achieve a response time at the level of microseconds. Even with the signal processing circuit, a response time as short as $30\mu s$ can be expected. This is much faster than that of typical vision cameras, which is generally at the level of milliseconds due to the time for exposure and image processing. In addition, the simple analogue output of PSD can be sampled easily by data acquisition devices with a high sampling rate. Moreover, the PSD camera provides high resolution[1], up to the level of $< 1\mu m$ when the sensing plane is $10mm \times 10mm$ large. Even under the influence of weak input, poor focusing, and noise, a resolution of $1.5\mu m$ can be easily achieved. This corresponds to a precision of $0.15mm$ for a $500mm \times 500mm$ measurement area. In addition, PSD is robust to focusing quality as it still measures the mass center of the illuminated area if the input is not focused. Furthermore, the peak spectral response of most PSD is around infrared. This makes it easy to remove the influence of environmental illumination by using an infrared pass filter. On the other hand, the sole drawback of PSD is the incapability of capturing images as what vision cameras do, and thus is incapable of sensing without marker. However, even for vision cameras, it is also a common practice to use markers of special features that are easy to identify by the image processing algorithms to expedite the response.

[1]Here as the analog resolution, i.e., the minimum position difference that can be distinguished on the sensing plane, equals to half of the sensing precision, and heavily depends on the signal to noise ratio.

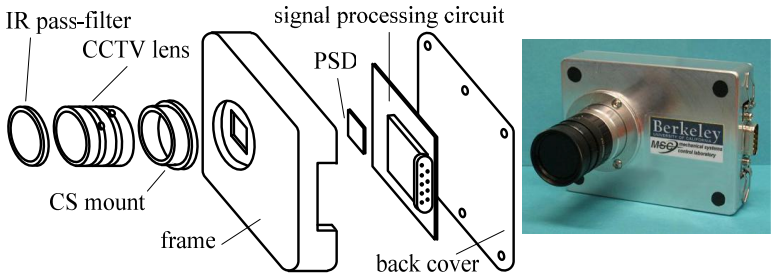


Figure 2.2: Structure of the PSD camera

PSD Camera

Fig. 2.2 shows the structure of the proposed PSD camera. A $9mm \times 9mm$ PSD (Hamamatsu, S5991-01) is adopted. It features a 2μs response time, and a typical 1.5μm resolution. A CCTV lens (FUJINON, HF9HA-1B) with $9mm$ fixed focal length is used. The lens offers a $48° \times 48°$ angle-of-view, or a $450mm \times 450mm$ field-of-view when the measurement plane locates $500mm$ in front of the camera. As PSD is robust to focusing quality, the focus adjustment does not matter much. The iris adjustment affects the intensity of total input illumination, and thus the signal-to-noise ratio and measurement precision. A maximum iris (F16) is selected in the experiments. An infrared pass filter (Anchor Optics, CR-39) is used to cut off all the input illumination except that from the markers.

The raw output signals of the PSD are weak current signals, whereas the final output of the PSD camera is the two-dimensional position of the input light spot on the sensing plane of the PSD, represented by two analogue voltage signals ranging from $-10V$ to $+10V$. A signal processing circuit is used for the conversion. An amplification unit first transforms the weak current signals to voltage signals, which will then be operated by the addition/subtraction and dividing units to perform the computation of (2.1). All operations are conducted by analog circuits (i.e., operational amplifiers and analog dividers) instead of digital processor after A/D conversion. This is to prevent the rounding error of the A/D process from being amplified. The output signals of the PSD camera are sampled by a data acquisition device (National Instruments, NI-6023E).

In the basic application setup (Fig. 2.3 (a)), the PSD camera is set on the ground, with an infrared LED beacon (Vishay, TSAL6400) mounted on the robot end-effector. With the end-effector's trajectory constrained in a plane, this setup provides two-dimensional measurement of the end-effector position. Advanced setup can be configured to achieve more measurement dimensions by using more markers and/or more PSD cameras (Fig. 2.3 (b)). The PSD camera can also be mounted on the end-effector, while the markers are mounted on the target.

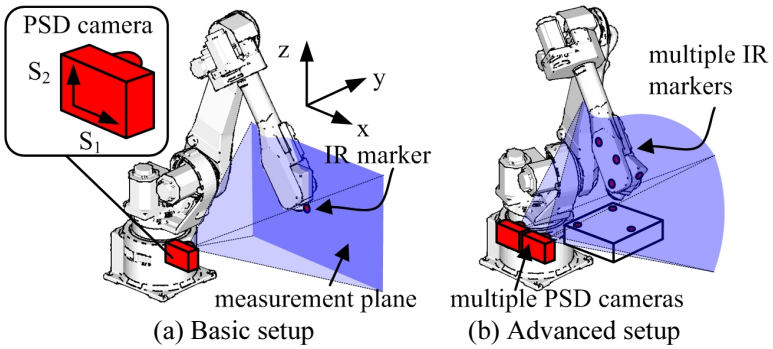


Figure 2.3: Setups of PSD camera

Calibration

Proper calibration is necessary to achieve good measurement accuracy. In the calibration process, two types of transformations are involved in the mapping between the spatial position of the target and the output signals of the PSD camera. They correspond to two parts of calibration. The first part, referred as linear calibration, corresponds to the perspective projection in an ideal camera model (Fig. 2.4), in which the spatial position of the marker and the position of the projected light spot on the sensing plane of PSD are related by

$$x_c = \frac{f}{Z_c} X_c, \qquad y_c = \frac{f}{Z_c} Y_c \tag{2.2}$$

$$\begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^{\mathrm{T}} = \boldsymbol{R}\left(\begin{bmatrix} X_s & Y_s & Z_s \end{bmatrix}^{\mathrm{T}} - \boldsymbol{T} \right) \tag{2.3}$$

where $(X_\bullet, Y_\bullet, Z_\bullet)$ and $(x_\bullet, y_\bullet, z_\bullet)$ denote positions of the marker P and its projection p respectively. The subscript c indicates the camera fixed coordinates, while s indicates the workspace coordinates of the robot. $\boldsymbol{R}$ and $\boldsymbol{T}$ are the rotation matrix and translation vector between the two coordinate systems. The camera outputs are thus $S_1 = K_1 x_c$ and $S_2 = K_2 y_c$, where K_1 and K_2 are the sensitivity gains of the PSD along its two axes. After the PSD camera is installed, linear calibration is conducted to determine K_1, K_2, $\boldsymbol{R}$, and $\boldsymbol{T}$. This is accomplished by measuring several points in the measurement plane by both the PSD camera and a reference instrument, then applying a least-squares fitting.

Besides linear calibration, nonlinear calibration is also necessary. It is to compensate the distortion in the overall sensing system, including the optical distortion of the lens, and the nonlinear response of the PSD and the signal processing circuit. Without considering these, it is very difficult to achieve an accuracy better than 2mm for a $450mm \times 450mm$ measurement area in the experiments. The common practice for nonlinear calibration is to model distortion with analytical models, which usually involve higher order polynomials [10, 11]. Such models, however, can only provide a rough approximation of the actual nonlinearity. The resulting accuracy of the PSD camera is still unacceptable. As an alternative, the distortion is recorded

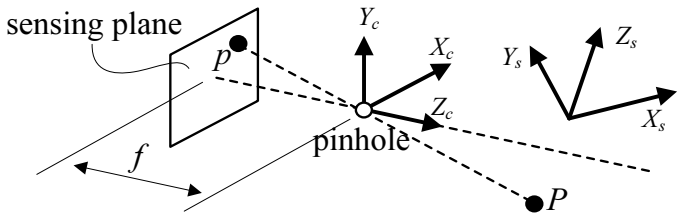


Figure 2.4: Perspective projection

by a lookup table built by densely scanning the sensing area point by point. The true values (obtained from a reference instrument) of every point and their corresponding PSD camera measurements (contaminated by distortion) are recorded and stored in the lookup table. Building such a lookup table can be time consuming. However, it needs to be conducted only once since the nonlinearity is the intrinsic characteristic of the PSD camera.

In the calibration process, the measurement plane is located parallel to the PSD sensing plane, $500mm$ in front of the PSD camera. An infrared marker is attached to the end-effector of an industrial robot, which is used to move the marker to specified locations in the measurement plane. Meanwhile, the CompuGauge system developed by Dynalog is used as a reference instrument. Its measurement of the end-effector position is regarded as the true value. CompuGauge senses the motion of the end-effector by measuring the motion of four strings attached to the end-effector. It gives a precision of $0.01mm$, and is widely used for robot calibration and performance analysis.

As shown in Fig. 2.5, during the scanning, the end-effector moves through the sensing area line by line without stop. The gap between adjacent lines is 1mm. Points are sampled along the lines with a time step corresponding to $1mm$ displacement. The actual path of the end-effector, however, will not be exactly the same as programmed. The sampled points are thus not uniformly aligned. This brings difficulty to the interpolation in the table lookup process. The problem is usually termed as scattered data interpolation [12]. One common step of various algorithms is to find the neighboring points around the inquired point. This induces significant computation load which is undesirable for real-time feedback applications. Thus, an alternative method is used. A new lookup table is built with ideally aligned grid. The value of each point on the grid is obtained by interpolation from the original table with the non-uniform grid. Any scattered data interpolation algorithm may be used. The result does not vary much as the grid is dense. This work needs to be done only once during the nonlinear calibration. The new lookup table can then be interpolated easily in real-time during the actual sensing process.

As illustrated in Fig. 2.6, in the actual operations of the PSD camera, the measured signals are first corrected to compensate distortion using the lookup table built in advance during the nonlinear calibration, then transformed through the perspective projection model. The final result represents the actual position of the target in the measurement plane.

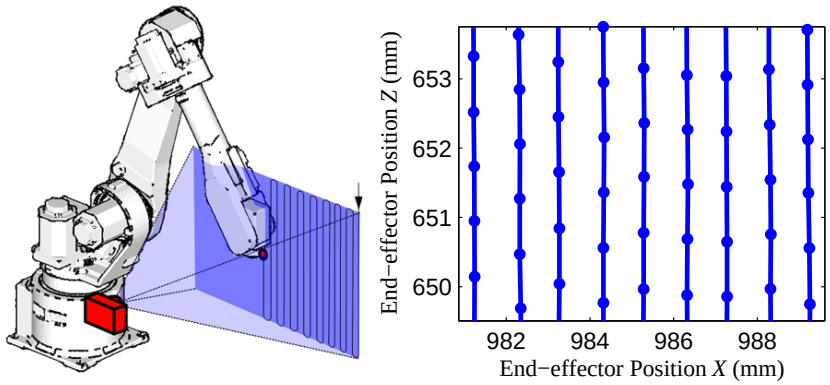


Figure 2.5: Sensing area scanning in nonlinear calibration

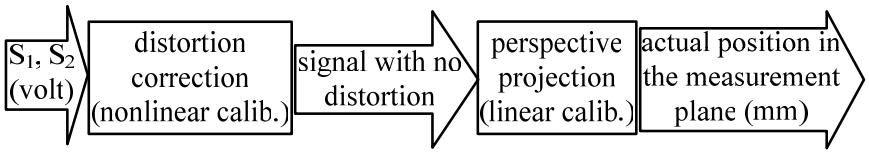


Figure 2.6: PSD camera calibration

Sensing Noise Mitigation

A low sensing noise level is essential to high precision sensing. In actual applications to industrial robots, the PSD camera suffers from strong environmental electro-magnetic interference due to the power supply of the robot servos. In experiments, the raw output is contaminated by noise with a distribution width of $15mV$, which corresponds to a $0.34mm$ precision when the measurement area is $450mm \times 450mm$ large. Shielded twisted pair cable with ferrite beads is used to avoid the common-mode noise and suppress the high frequency noise. A digital low-pass filter is also adopted after the A/D process. The filtering phase lag can be sufficiently small if the initial sampling rate is high enough (e.g. ten times of the desired sampling rate). The distribution width of the noise is then reduced to $6.8mV$. The corresponding precision levels on the measurement planes located at different distances in front of the PSD camera are summarized in Fig. 2.7.

2.3 Velocity Estimation with PSD Camera and Inertial Sensors

In addition to position feedback, velocity feedback is also important to motion control. The developed PSD camera has good capability on position sensing. Its measurement alone, how-

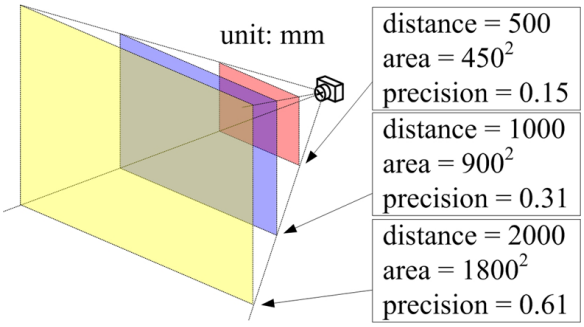


Figure 2.7: Precision levels for measurement planes at different distances

ever, is not sufficient to support fine velocity estimation. As shown in Fig. 2.11, the velocity estimation acquired by differentiating the PSD measurement is highly unacceptable. Methods have been studied to improve velocity estimation acquired from position measurement. Most of them, however, are based on dynamics models[9]. They require accurate parameters of system dynamics, and are difficult to apply due to the complexity of robot dynamics. One promising alternative is to use a sensor fusion algorithm to include measurements from additional inertial sensors (accelerometer and gyroscope) installed on the target (e.g. robot end-effector). No parameter of system dynamics is needed if the kinematic model is used to relate position to acceleration and angular velocity, where the latter two are treated as the model input and measured directly by accelerometer and gyroscope. This is the basic idea of kinematic Kalman filter (KKF). Here, a KKF for three-dimensional rigid body motion [2] is formulated to fuse the measurements from the PSD camera and inertial sensors.

Kinematic Kalman Filter for 3D Rigid Body Motion

The kinematic model of robot end-effector can be described as

$$\dot{\boldsymbol{p}}^s_{TCP} = \boldsymbol{v}^s_{wr} + \boldsymbol{R}[\boldsymbol{\omega}^b\times]\boldsymbol{r}^b_{TCP/wr} \tag{2.4}$$

$$\dot{\boldsymbol{v}}^s_{wr} = \boldsymbol{R}\boldsymbol{f}^b_{wr} + \boldsymbol{g}^s \tag{2.5}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}[\boldsymbol{\omega}^b\times] \tag{2.6}$$

where the subscript TCP indicates tool center point, and wr indicates the wrist point where the inertial sensors locate. s and b denote the workspace coordinates of the robot and the body coordinates of the end-effector respectively. $\boldsymbol{p}$ is position, $\boldsymbol{v}$ is velocity, $\boldsymbol{\omega}$ is angular velocity, $\boldsymbol{f}$ is acceleration, $\boldsymbol{g}$ is gravity, and $\boldsymbol{R}$ is the rotation matrix of the end-effector. $[\bullet\times] \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix, equivalent to the cross product operation of the corresponding vector. With this model, at the k-th time step, the open loop prediction can

be conducted as

$$\hat{\boldsymbol{R}}(k+1|k) = \hat{\boldsymbol{R}}(k|k)(\boldsymbol{I} + T[\bar{\boldsymbol{\omega}}^b(k+1)\times] + \tfrac{T^2}{2}[\bar{\boldsymbol{\omega}}^b(k+1)\times]^2) \tag{2.7}$$

$$\begin{bmatrix} \hat{\boldsymbol{p}}^s_{TCP}(k+1|k) \\ \hat{\boldsymbol{v}}^s_{wr}(k+1|k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & T\boldsymbol{I} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{p}}^s_{TCP}(k|k) \\ \hat{\boldsymbol{v}}^s_{wr}(k|k) \end{bmatrix}$$

$$+T \begin{bmatrix} -\hat{\boldsymbol{R}}(k+1|k)[\boldsymbol{r}^b_{TCP/wr}\times] & \frac{T}{2}\hat{\boldsymbol{R}}(k+1|k) \\ \boldsymbol{0} & \hat{\boldsymbol{R}}(k+1|k) \end{bmatrix} \times \begin{bmatrix} \bar{\boldsymbol{\omega}}^b(k+1) \\ \bar{\boldsymbol{f}}^b_{wr}(k+1) \end{bmatrix} + T \begin{bmatrix} \frac{T}{2}\boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix} \hat{\boldsymbol{g}}^s \tag{2.8}$$

where $\hat{\bullet}$ indicates estimated value, $\bar{\bullet}$ indicates measured value. T is the sampling time. The position measurement ($\bar{\boldsymbol{p}}^s_{TCP}$) from the PSD camera is then used to perform correction as follows

$$\delta\hat{\boldsymbol{x}}(k+1|k+1) = \delta\hat{\boldsymbol{x}}(k+1|k) + \boldsymbol{L}(k+1)[\bar{\boldsymbol{p}}^s_{TCP}(k+1) - \hat{\boldsymbol{p}}^s_{TCP}(k+1|k) + \delta\hat{\boldsymbol{p}}(k+1|k)] \tag{2.9}$$

where $\delta\hat{\boldsymbol{x}} = \begin{bmatrix} \delta\hat{\boldsymbol{p}}^T & \delta\hat{\boldsymbol{v}}^T & \hat{\boldsymbol{\psi}}^T \end{bmatrix}^T$ includes the correction terms for position, velocity, and rotation matrix. $\boldsymbol{L}$ is the Kalman filter gain determined from the estimation error propagation. Detailed derivation can be found in [2].

2.4 Results

Accuracy of the PSD camera

Tests are conducted to evaluate the performance of the PSD camera for end-effector position sensing. The measurement plane locates $500mm$ in front of the PSD camera. An industrial robot (FANUC, M-16iB) is programmed to move its end-effector along square and circle paths without rotation. The end-effector position is measured by both the PSD camera and a CompuGauge system (Fig. 2.8), with the measurement from the latter regarded as the true value.

Fig. 2.9 shows the comparison of the measurement results. The figures on the right column show the zoomed-in details of the figures on the left. Among repetitive experiments, an average accuracy[2] at the level of $0.05 \sim 0.1mm$ is achieved, which is quite promising considering the size of the measurement area ($450mm \times 450mm$). Higher accuracy can be achieved if a smaller measurement area is used.

Sensor Fusion

An inertial measurement unit (Analog Devices, ADIS16400) is installed on the end-effector of the robot manipulator shown in Fig. 2.8. It includes a triaxial gyroscope and a triaxial

[2]Represented by the root-mean-square difference between the measurements from the PSD camera and CompuGauge system.

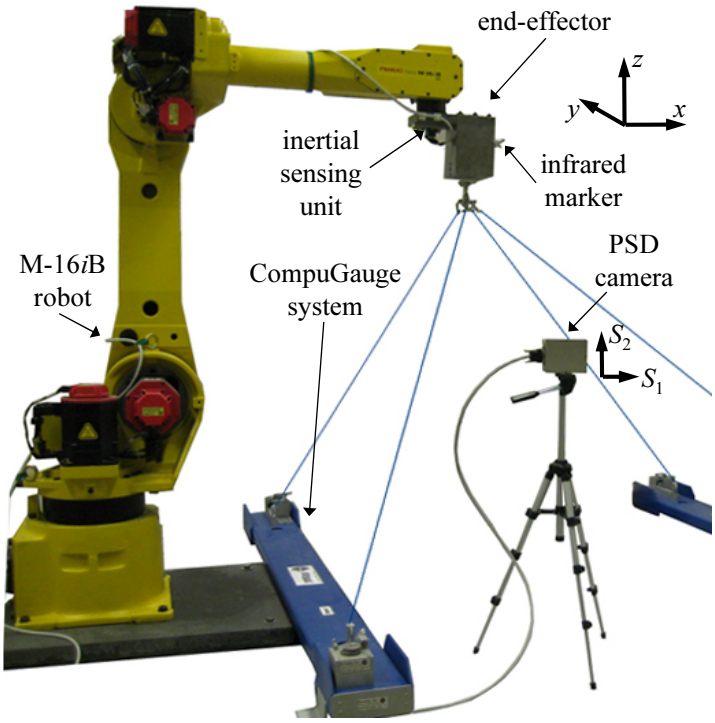
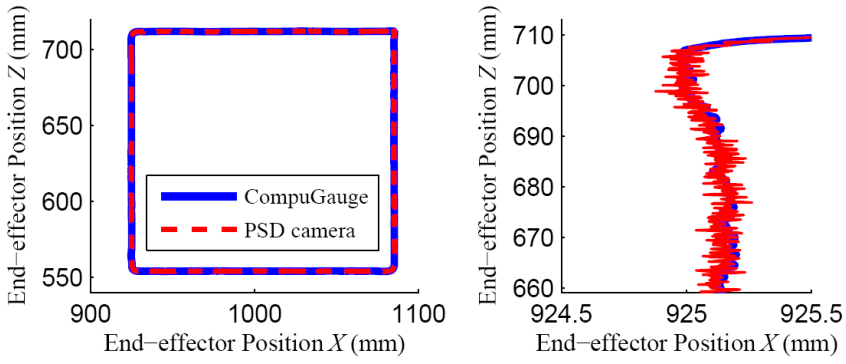


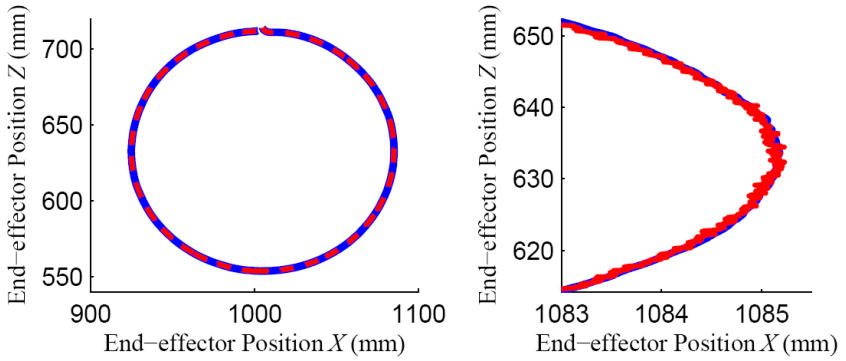
Figure 2.8: A FANUC M-16iB robot with auxiliary sensors

accelerometer. Again, the TCP position of the end-effector is measured by both the PSD camera and the CompuGauge system. The measurement from the latter and its differentiation are regarded as the true position and velocity. The sampling time is set as $1ms$. Two sets of experiments are conducted. In the first set, one marker is installed on the end-effector, providing two-dimensional sensing of position without rotation measurement. The trajectories used are the same with those in the PSD camera performance tests (Fig. 2.9). Fig. 2.11 (a) and (b) show the velocity estimation results using KKF, as well as a comparison with the results generated by differentiation of the position measurement from the PSD camera. The superiority of the sensor fusion scheme is evident.

In the second set of experiments, two markers are installed on the end-effector (Fig. 2.10), with a $120mm$ distance between them. The motion of the end-effector is still constrained in the x-z plane, with orientation change allowed only along the y axis. With two markers, the TCP position can be determined by triangulation based on the measured positions of the markers and their known locations on the end-effector. The markers are controlled by a micro-processor (Atmel, Mega328) to flash alternately, and thus sensed by the PSD camera alternately. The end-effector path is illustrated in Fig. 2.10. Fig. 2.11 (c) shows the velocity estimation result. Again, the result from KKF shows significant superiority. However, compared with the first set of experiments where no orientation change is involved, the



(a) Square path



(b) Circle path

Figure 2.9: Measurement comparison between PSD camera and CompuGauge

velocity estimation along the z axis shows some drifting phenomena. This is due to the calibration inaccuracy of the gyroscope. To address this, a more complicated sensor fusion algorithm with the consideration of sensor self-calibration may be necessary.

2.5 Chapter Summary

Non-contact direct motion sensing of end-effector is desirable to overcome the sensing limitations of conventional indirect drive-trains of industrial robots. This chapter first proposed the development of a PSD camera using position sensitive detector. While being cost-effective, the PSD camera provides much faster response than typical vision systems. With proper work on noise mitigation and calibration, the PSD camera demonstrated good precision and accuracy. These advantages make it favorable for real-time feedback systems. Furthermore, to obtain good velocity estimation, a kinematic Kalman filter based on a three-dimensional rigid body motion model was applied to perform sensor fusion, where the measurements from inertial sensors mounted on the robot end-effector were fused with that from the PSD

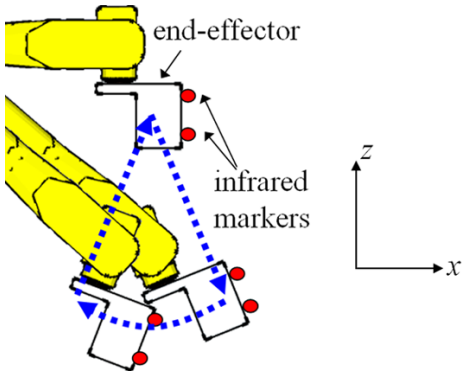
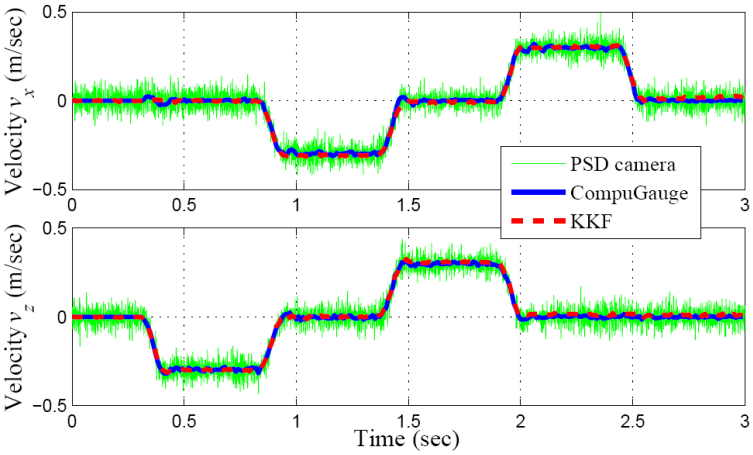
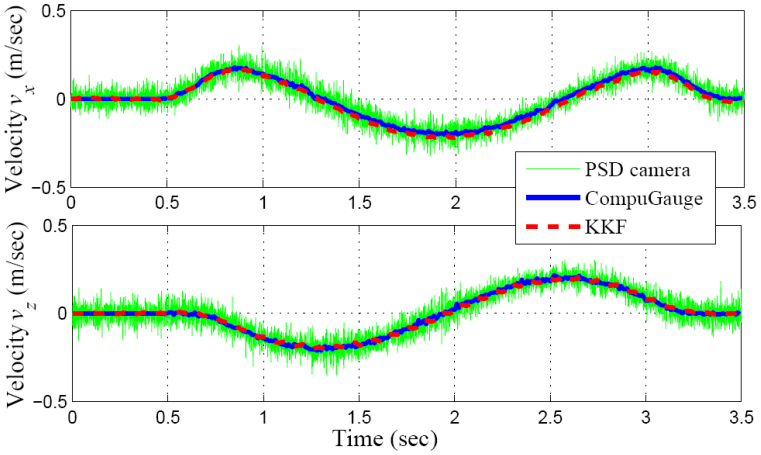


Figure 2.10: End-effector path with orientation change

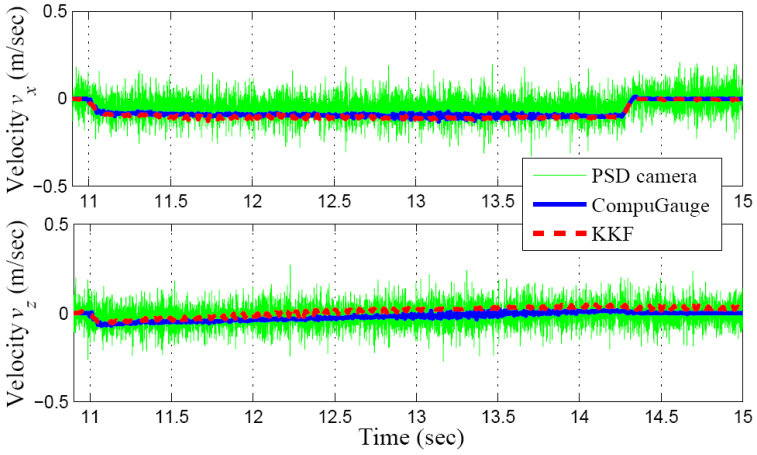
camera. Performance tests of the PSD camera and validation experiments of the sensor fusion scheme were conducted on an industrial robot. Promising results were demonstrated on both position sensing and velocity estimation of the robot end-effector.



(a) Square path with no orientation change



(b) Circle path with no orientation change



(c) Curve path with orientation change

Figure 2.11: Velocity estimation results

Chapter 3

Statistical Learning for Feedback Compensation

3.1 Introduction

In the scenarios where a favorable sensing system such as the one introduced in Chapter 2 is not available, this chapter presents a statistical learning algorithm to compensate feedback with large delay and slow sampling rate. A major application of the approach is visual guided control of industrial robots. Currently, vision feedback of industrial robots is mainly used in three types of tasks: (a) coordinate offset, (b) bin picking, (c) picking from belt conveyors. In those tasks, the vision system is used to locate the target before the robot starts to move. Then, the motion trajectory of the robot is offset or re-planned so that the robot can reach the target by following the trajectory. Such control method is called look-and-move or look-then-move. Because the target is in motion, the specific control method of (c) is a little more complicated than (a) and (b). Predictive action is needed to compensate the displacement of the target between the visual sensing action and the picking action. The prediction, however, can be as simple as a linear extrapolation because an object carried by a belt conveyor moves in straight line at a known constant speed [13].

Despite being widely used, the look-then-move method cannot support many emerging applications which require real-time vision guidance. The term "real-time" means that the visual feedback is updated and utilized to guide the robot at the same rate of the motion controller, which is usually at the order of $1kHz$. One example is picking a swinging object from a hanging conveyor (Fig. 3.1). The control method used in picking objects from a belt conveyor, i.e., look-then-move with prediction, cannot give satisfactory performance in this task because the motion of a swinging object is less predictable. Motion control based on real-time visual feedback is often called visual servoing. In many academic results on visual servoing, robots can track objects that are in agile and complex motion. The vision hardware used in such research, however, are specially made and too expensive for industrial users. Cost effectiveness has always been a key consideration in the manufacturing industry.

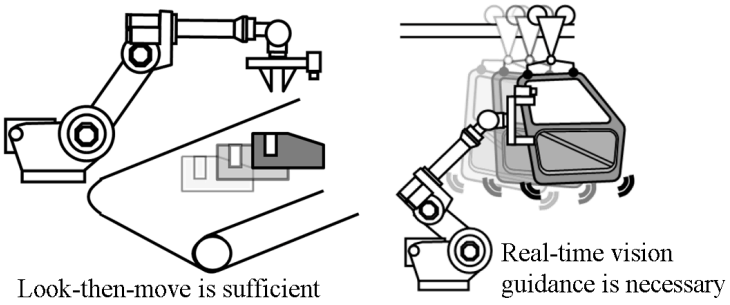


Figure 3.1: Motivation of real-time vision guidance

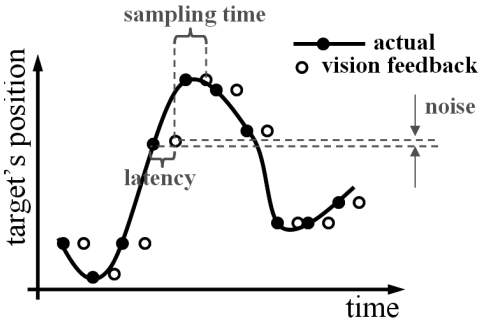


Figure 3.2: Visual sensing dynamics

Due to cost limit of hardware, typical robot vision systems used on industrial robots have a considerable latency caused by image transmission and processing at the order of $100ms$. The corresponding sampling rate has to be low enough so that image transmission and processing can be finished before the next image is taken; otherwise the transmission and processing tasks will queue up. It is unlikely that in the near future the speed of industrial robot vision products can be improved to match the desired sampling rate of motion controllers at the order of $1kHz$. The software compensation of visual sensing dynamics, however, has been addressed little. Academic research on visual servoing usually does not consider cost effectiveness for industrial applications and employs high-end visual sensing and processing hardware, for which visual sensing dynamics is not a limiting factor.

There are two key aspects in realizing real-time vision guidance for industrial robots:

1) Estimating the motion of the target using vision feedback: Due to cost considerations, typical industrial machine vision systems have low sampling rate and large latency. Usually, the preferred rate for robot motion controller is at least $1kHz$, whereas the sensing latency introduced in image acquisition and processing is on the order of $10ms$ to $100ms$, and the corresponding sampling rate has to be low enough to avoid queuing of the image acquisition and processing tasks for consecutive images. These factors are sometimes called *visual sens-*

ing dynamics [14] (Fig. 3.2). It makes real-time tracking of a moving target difficult. For this reason, *visual sensing dynamics compensation (VSDC)* is needed.

2) Guiding the robot to approach and track the target based on vision feedback: This task is often called *visual servoing*. It differs from conventional tasks such as welding and palletizing in terms of no reference trajectory is known in advance. Instead, the motion of the target is measured in real-time. In addition, at the beginning of a task, the end-effector of the robot is at a distance from the target. These factors require a control method different from the conventional trajectory tracking algorithm. Meanwhile, in order to benefit the end users, it is still desirable to preserve the trajectory tracking control algorithm that is available on most robots, and make the new control algorithm an add-on. This add-on is called *kinematic visual servoing (KVS)*.

In regard to the two aspects introduced above, in the rest of this chapter, Section 3.2 presents a statistical learning algorithm to compensate slow feedback signals. Section 3.3 discusses the control schemes to utilize the compensated feedback signals for real-time tracking control.

3.2 Compensating Slow Feedback

One way to realize real-time vision guidance affordably is by using alternative camera-like sensors other than the conventional image cameras. One type of such devices uses multiple linear CCD sensors to measure active markers (usually LED infrared beacons) installed on the target. Each linear CCD camera provides one dimensional measurement of the marker's position. By designing the layout of the sensors properly, the three dimensional positions of the markers can be obtained. Cylindrical lenses can be used to focus the illumination from the markers onto the linear sensors. A representative of this type of devices is the Optotrak system developed by Northern Digital. In Chapter 2, Section 2.2 presented another type of alternative vision system. A two-dimensional analog position sensitive detector (PSD) is used to measure the position of infrared markers. As an analog device, a PSD sensor provides a response time as short as $30\mu s$ and can be sampled at very high frequencies. A common limit of these alternative vision systems is the requirement of markers, which is not applicable in many manufacturing tasks.

In the situation where alternative sensors are not applicable, a visual sensing dynamics compensation (VSDC) algorithms can be applied to directly compensate the latency and low sampling rate of visual feedback so that it can be used for real-time motion control [15]. In order to compensate latency, the actual move of the target must be predicted from the delayed measurements. To deal with slow sampling, the target's motion between consecutive image sampling actions needs to be estimated as well. In addition, noise comes from quantization and artifacts in image processing. The general framework of Bayes filtering accommodates all the factors mentioned above. In particular, there are three key elements: (a) a stochastic model describing the motion of the target, (b) an estimation law (e.g. a Kalman filter), and (c) a statistical learning law to tune the parameters in the model and the estimation law. In

the rest of this section, the proposed VSDC algorithm is presented by describing its specific configurations of those three elements.

The motion of the target can be viewed as a vector signal of multiple dimensions. Each dimension of the motion, such as the target's horizontal or vertical position in the image, is a scalar time-varying signal $s(t)$. The image processing software acquires the values of $s(t)$ from the images with a sampling period and a latency that are much longer than the basic sampling period of the algorithm. A VSDC algorithm estimates the true value of $s(t)$ from the delayed and slowly sampled measurements.

Modeling the Target Motion

A preferred model should be able to describe various motions with little a-priori information of the target. One counter-example is the vision guided robotic grasping presented in [16] and [17]. In the papers, a projectile model is used to predict the motion of flying objects. The model cannot describe motions other than projectile flying. Models that describe the motion of a target are needed also in many other areas such as radar signal processing [18]. One popular choice is a so called kinematic model [19, 7, 8, 20]. The model uses the position and velocity of the target as the state variables. Dynamics of the motion is as simple as a double-integrator with acceleration as the input. Inspired by the kinematic model, the Taylor expansion of $s(t)$ is considered:

$$s\left(t_0+t\right)=\sum_{c=0}^{r}s^{(c)}\left(t_0\right)\frac{t^c}{c!}+s^{(r+1)}\left(t_0+\xi\right)\frac{t^{r+1}}{(r+1)!} \tag{3.1}$$

where r is the order of the expansion, and $\xi\in(0,t)$. Rewrite Eq. (3.1) in the form of a discrete state space model:

$$x\left(i+1\right)=\begin{bmatrix}1 & T & \cdots & \frac{T^r}{r!}\\ & 1 & \cdots & \frac{T^{r-1}}{(r-1)!}\\ & & \ddots & \vdots\\ & & & 1\end{bmatrix}x\left(i\right)+\begin{bmatrix}\frac{T^{r+1}}{(r+1)!}\\ \frac{T^r}{r!}\\ \vdots\\ T\end{bmatrix}u\left(i\right) \tag{3.2}$$

where the state vector $x=\begin{bmatrix}s & s' & \cdots & s^{(r)}\end{bmatrix}^{\mathrm{T}}$ contains s and its derivatives. i is the index of time step. T is the sampling time of the algorithm, which can be much shorter than the sampling period of the visual feedback. u comes from the residual term in Eq. (3.1). It is treated as an unknown input. An *equivalent noise approach* is used to handle u. The approach has long been applied in the field of radar signal processing [18]. It suggests that a stochastic process w can be designed to replace the actual input u to provide a similar effect of excitation. The spectral characteristics of the stochastic process is designed based on the specific type of the target. For the general case, white noise can be used. Such implementation is called *input whitening*. In order to reduce the influence of the difference between the fictitious stochastic input and the actual input, a higher order model is desired.

In practice, without introducing too much computation load, a whitening depth of $r = 4$ is used. Further increasing r can improve the estimates of higher order derivatives of s, but not much the estimates of s and s', which are the two state variables primarily used in visual servoing.

The sampling period and latency of the visual feedback are assumed to be N and L times the basic time period of the algorithm respectively. Without loss of generality, N and L are assumed to be integers. This is reasonable because N and L are typically large and their decimal parts can be safely ignored by rounding N and L to the next integers. The visual feedback is then described by

$$y(j) = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} x(j-L) + v(j) \tag{3.3}$$

where y is the visual feedback signal. v is the sensing noise with a covariance V. $j = N, 2N, 3N, ...$ is the index of the sampling actions of the camera. Equations (3.2) and (3.3) give a dual-rate state space model:

$$\begin{cases} x(i+1) = Ax(i) + B_w w(i) & (i = 0, 1, 2, 3, ...) \\ y(j) = Cx(j-L) + v(j) & (j = N, 2N, 3N, ...) \end{cases} \tag{3.4}$$

where A, B_w, and C are as shown in Eq. (3.2) and (3.3). w is the fictitious stochastic input that replaces the actual input u in Eq. (3.2). The value of its covariance W is adjusted by the statistical learning algorithm discussed later in Section 3.2.

Dual-rate Kalman Filtering for Prediction and Interpolation

Rewrite Eq. (3.4) with a unified time step:

$$\begin{cases} x_v(k+1) = A_v x_v(k) + w_v(k) \\ y_v(k) = Cx_v(k) + v_v(k) \end{cases} \quad (k = 1, 2, 3, ...) \tag{3.5}$$

where

$$A_v = A^N \tag{3.6}$$
$$x_v(k) = x(i = Nk - L) \tag{3.7}$$
$$y_v(k) = y(j = Nk) \tag{3.8}$$
$$v_v(k) = v(j = Nk) \tag{3.9}$$
$$w_v(k) = \sum_{c=0}^{N-1} A^{N-1-c} B_w w(Nk - L + c) \tag{3.10}$$

Note that the time steps of x, y, w, and v are counted by i or j, whereas the time steps of x_v, y_v, w_v, and v_v are counted by k. Using Eq. (3.5), a dual-rate Kalman filter can be

formulated. The prediction action runs at the basic sampling rate of the motion controller (e.g., $1kHz$).

$$\begin{aligned}
\hat{x}\left(Nk-L+m\,|Nk\right) =& A^m\hat{x}\left(Nk-L\,|Nk\right) \\
=& A^m\hat{x}_v\left(k\,|k\right) \qquad (m=1 \text{ to } N+L) \qquad (3.11)\\
\hat{x}_v\left(k+1\,|k\right) =& \hat{x}\left(Nk-L+N\,|Nk\right) \\
=& A_v\hat{x}_v\left(k\,|k\right) \qquad (3.12)\\
\hat{y}_v\left(k+1\,|k\right) =& C\hat{x}_v\left(k+1\,|k\right) \qquad (3.13)
\end{aligned}$$

where $\hat{\bullet}\left(a|b\right)$ denotes the estimate of $\bullet(a)$ given all feedback information up to the time step b. The correction action runs at the sampling rate of the camera (e.g., $30Hz$).

$$\begin{aligned}
&\hat{x}_v\left(k+1\,|k+1\right) = \hat{x}\left(Nk-L+N\,|Nk+N\right) \\
&= \hat{x}_v\left(k+1\,|k\right) + K\left(k+1\right)\left(y_v\left(k+1\right) - \hat{y}_v\left(k+1\,|k\right)\right)
\end{aligned} \qquad (3.14)$$

where

$$K\left(k+1\right) = M\left(k+1\right)C^{\mathrm{T}}\left(CM\left(k+1\right)C^{\mathrm{T}}+V\right)^{-1} \qquad (3.15)$$

is the correction gain determined by the propagation dynamics of the error covariance

$$M\left(k+1\right) = A_vM\left(k\right)A_v^{\mathrm{T}} + W_v - A_vM\left(k\right)C^{\mathrm{T}}\left(CM\left(k\right)C^{\mathrm{T}}+V\right)^{-1}CM\left(k\right)A_v^{\mathrm{T}} \qquad (3.16)$$

where

$$\begin{aligned}
W_v &= \sum_{c=1}^{N} A^{N-c}B_wW\left(A^{N-c}B_w\right)^{\mathrm{T}} \\
&= W\sum_{c=1}^{N} A^{N-c}B_w\left(A^{N-c}B_w\right)^{\mathrm{T}} \\
&= W\Sigma
\end{aligned} \qquad (3.17)$$

Figure 3.3 explains the dual-rate Kalman filtering. The correction action takes place every N prediction steps. Because of the sensing latency, the correction is applied to a state in the past, whereas the present state is estimated by predicting from the corrected past state.

Statistical Parameter Learning

A learning algorithm is needed to adjust the parameter W, which is the covariance of the fictitious stochastic input in the model of target's motion. Consider Eq. (3.5) and a non-optimal correction with W_v improperly set as Q. The prediction and correction actions are still conducted as in Eq. (3.11) to (3.14). The correction gain, however, is different from Eq. (3.15) and is given by

$$K'\left(k+1\right) = P\left(k+1\right)C^{\mathrm{T}}\left(CP\left(k+1\right)C^{\mathrm{T}}+V\right)^{-1} \qquad (3.18)$$

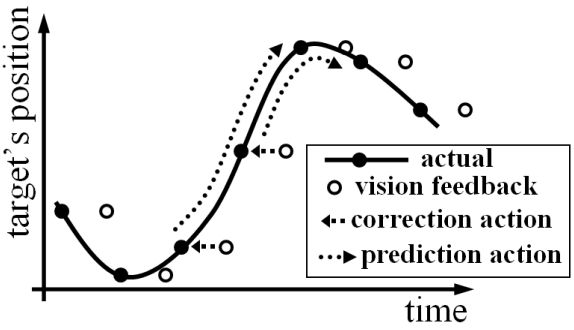


Figure 3.3: Dual-rate Kalman filtering to compensate latency and low sampling rate

where

$$\begin{aligned} P(k+1) &= A_v P(k) A_v^{\mathrm{T}} + Q - A_v P(k) C^{\mathrm{T}} \left(CP(k) C^{\mathrm{T}} + V\right)^{-1} CP(k) A_v^{\mathrm{T}} \\ &= (A_v - A_v K'(k) C) P(k) (A_v - A_v K'(k) C)^{\mathrm{T}} + Q + A_v K'(k) V (A_v K'(k))^{\mathrm{T}} \end{aligned} \tag{3.19}$$

Because $Q \neq W_v$, P is not the actual covariance of the prediction error $\tilde{x}_v$, which is given by

$$\begin{aligned} \tilde{x}_v(k+1|k) &= x_v(k+1) - \hat{x}_v(k+1|k) \\ &= A_v \tilde{x}_v(k|k-1) + w_v(k) - A_v K'(k) (C\tilde{x}_v(k|k-1) + v_v(k)) \end{aligned} \tag{3.20}$$

The propagation of the corresponding error covariance M' is

$$\begin{aligned} M'(k+1) &= E\left\{\tilde{x}_v(k+1|k)\, \tilde{x}_v^{\mathrm{T}}(k+1|k)\right\} \\ &= A_v K'(k) V (A_v K'(k))^{\mathrm{T}} + W_v + (A_v - A_v K'(k) C) M'(k) (A_v - A_v K'(k) C)^{\mathrm{T}} \end{aligned} \tag{3.21}$$

Subtracting Eq. (3.21) from Eq. (3.19) gives

$$P(k+1) - M'(k+1) = Q - W_v + \Phi(k) \quad (P(k) - M'(k)) \Phi^{\mathrm{T}}(k) \tag{3.22}$$

where $\Phi(k) = A_v - A_v K'(k) C$. At steady state

$$P_{ss} - M'_{ss} = \Phi_{ss} (P_{ss} - M'_{ss}) \Phi_{ss}^{\mathrm{T}} + Q - W_v \tag{3.23}$$

where the subscript ss stands for the steady state values. $W_v = \Sigma W$ and $Q = \Sigma q$, where W and q are scalars. Σ is a matrix as defined in Eq. (3.17). Equation (3.23) is a Lyapunov equation whose solution is

$$P_{ss} - M'_{ss} = \bar{\Sigma}(q - W) \tag{3.24}$$

where

$$\bar{\Sigma} = \sum_{c=1}^{N_a} \Phi_{ss}^{N_a - 1} \Sigma \left(\Phi_{ss}^{N_a - 1}\right)^{\mathrm{T}} \tag{3.25}$$

Equation (3.24) is utilized to estimate W. N_a is set to a sufficiently large number instead of infinity to give a truncated solution of the Lyapunov equation. Then, in Eq. (3.24), $\bar{\Sigma}$, P_{ss}, and q are known. M'_{ss} is unknown, but can be approximated as follows. The prediction error of y_v is

$$\begin{aligned}\tilde{y}_v\left(k\left|k-1\right.\right) &= y_v\left(k\right) - \hat{y}_v\left(k\left|k-1\right.\right) \\ &= Cx_v\left(k\right) + v_v\left(k\right) - C\hat{x}_v\left(k\left|k-1\right.\right) \\ &= C\tilde{x}_v\left(k\left|k-1\right.\right) + v_v\left(k\right)\end{aligned} \tag{3.26}$$

Its covariance is

$$E\left\{\tilde{y}_v\left(k\left|k-1\right.\right)\tilde{y}_v^{\mathrm{T}}\left(k\left|k-1\right.\right)\right\} = CM'_{ss}C^{\mathrm{T}} + V \tag{3.27}$$

where $CM'_{ss}C^{\mathrm{T}} = m_{11}$ is the (1,1) component of M'_{ss}. Approximate the expectation of $\tilde{y}_v\left(k\left|k-1\right.\right)\tilde{y}_v^{\mathrm{T}}\left(k\left|k-1\right.\right)$ with its average η over a short time period. In this way, $m_{11} \approx \eta - V$. Substituting m_{11} into the (1,1) component of Eq. (3.24) gives an estimate of W:

$$\hat{W} = q - \frac{1}{\bar{\sigma}_{11}}\left(p_{11} - \eta + V\right) \tag{3.28}$$

where p_{11} is the (1,1) component of P_{ss}. $\bar{\sigma}_{11}$ is the (1,1) component of $\bar{\Sigma}$.

3.3 Utilizing the Compensated Feedback in Real-Time Control

Assuming a VSDC algorithm is used to compensate the low sampling rate and large latency introduced in image acquisition and processing, Fig. 3.4 shows probably the most straightforward method for visual servoing. First, the measured motion of the target is converted from the image coordinates to the robot joint coordinates. Then, the motion of the target expressed in robot joint coordinates is fed directly as the reference to a conventional trajectory tracking controller. At the beginning of a task, however, the end-effector of the robot is at a distance from the target. This approach equals to performing trajectory tracking under a very large initial error without homing the robot, whereas a trajectory tracking controller is designed to track optimized trajectories with a proper homing action at the beginning. The performance would be unsatisfactory. The robot might even go unstable.

A Cascade Structure via Multi-sliding Surface Control

A vision guided robot manipulator of eye-in-hand configuration can be described by

$$\begin{cases} \boldsymbol{\tau} = \mathbf{M}\left(\mathbf{q}\right)\ddot{\mathbf{q}} + \mathbf{C}\left(\mathbf{q},\dot{\mathbf{q}}\right)\dot{\mathbf{q}} + \mathbf{F}\left(\dot{\mathbf{q}}\right) + \mathbf{G}\left(\mathbf{q}\right) \\ \mathbf{X}_{ee} = f_{rbt}\left(\mathbf{q}\right) \\ \mathbf{s}_{tgt} = f_{img}\left(\mathbf{X}_{ee}, \mathbf{X}_{tgt}\right) \end{cases} \tag{3.29}$$

where $\boldsymbol{\tau}$ is the torque exerted by the motors, $\mathbf{q}$ is the rotation of the motors, $\mathbf{M}$ is the inertia matrix, $\mathbf{C}$ gives the centrifugal and Coriolis effect, $\mathbf{F}$ is friction, and $\mathbf{G}$ is gravity.

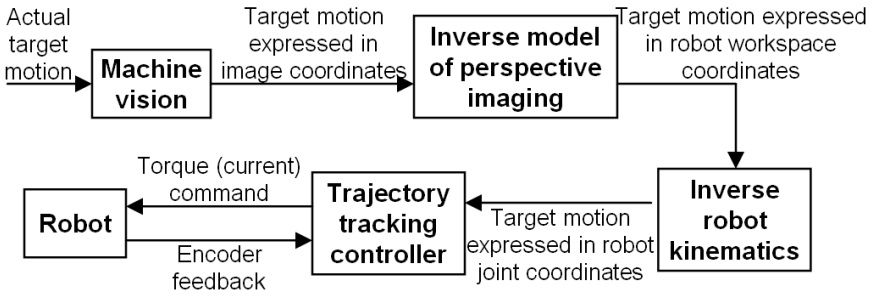


Figure 3.4: A simple method for visual servoing

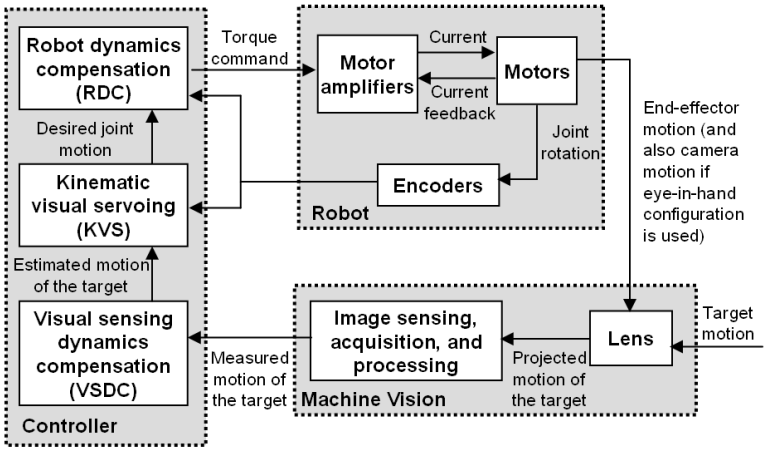


Figure 3.5: A cascade control structure

$\mathbf{X}_{ee} = \left(\mathbf{p}_{ee}^T, \mathbf{r}_{ee}^T\right)^T$ includes the position and orientation of the end-effector in the workspace of the robot, f_{rbt} is the robot kinematics. $\mathbf{s}_{tgt}$ includes the position and orientation of the target in the image space, $\mathbf{X}_{tgt} = \left(\mathbf{p}_{tgt}^T, \mathbf{r}_{tgt}^T\right)^T$ includes the position and orientation of the target in the workspace of the robot, f_{img} is the imaging kinematics. The control goal is to regulate $\mathbf{s}_{tgt}$ to a set-point (e.g., zero).

Using sliding control, a sliding surface can be built as $\mathbf{S} = \dot{\mathbf{e}} + \lambda \mathbf{e}$ ($\lambda > 0$). The derivative of $\mathbf{S}$ is

$$\dot{\mathbf{S}} = \ddot{\mathbf{e}} + \lambda \dot{\mathbf{e}} = \ddot{\mathbf{s}}_{tgt} + \lambda \dot{\mathbf{s}}_{tgt} \tag{3.30}$$

where $\ddot{\mathbf{s}}_{tgt}$ is a function of $\ddot{\mathbf{q}}$, $\dot{\mathbf{q}}$, $\mathbf{q}$, $\mathbf{X}_{tgt}$, $\dot{\mathbf{X}}_{tgt}$, and $\ddot{\mathbf{X}}_{tgt}$. $\dot{\mathbf{s}}_{tgt}$ can be estimated from the vision feedback. Because $\ddot{\mathbf{q}}$ is related directly to the torque $\boldsymbol{\tau}$, $\ddot{\mathbf{s}}_{tgt}$ is also related directly to $\boldsymbol{\tau}$. This allows a design of $\boldsymbol{\tau}$ (the control of the plant) to satisfy the sliding condition

$$S_i \dot{S}_i \leq -\eta \left| S_i \right| \tag{3.31}$$

where S_i's are the components of $\mathbf{S}$. $\eta > 0$ governs the convergence speed. Equation (3.31) is a sufficient condition for error convergence. This method, however, is difficult to realize. The relationship between $\ddot{\mathbf{s}}_{tgt}$ and $\boldsymbol{\tau}$ involves both the dynamics and the second order differential kinematics of the robot. An analytical form of the control law, if can be derived, is very complicated.

Instead of a single sliding surface, consider two sliding surfaces in cascade. The first one has the simple form of $\mathbf{S}_1 = \mathbf{s}_{tgt}$. Its derivative is

$$\dot{\mathbf{S}}_1 = \dot{\mathbf{s}}_{tgt} = \mathbf{J}_{img}\mathbf{J}_{rbt}\dot{\mathbf{q}} + \frac{\partial \mathbf{s}_{tgt}}{\partial t} \tag{3.32}$$

where $\mathbf{J}_{rbt}$ is the robot Jacobian. $\mathbf{J}_{img} = \nabla_{\mathbf{x}_{ee}} f_{img}$ is called a image Jacobian. It is evaluated as if the target stays still and only the camera is moving, whereas $\frac{\partial \mathbf{s}_{tgt}}{\partial t}$ is evaluated as if only the target is moving. It can be evaluated from $\mathbf{q}$, $\dot{\mathbf{q}}$, $\mathbf{s}_{tgt}$ and $\dot{\mathbf{s}}_{tgt}$ using Eq. (3.32). The actual control of the robot, the torque $\boldsymbol{\tau}$, does not appear explicitly in $\dot{\mathbf{S}}_1$. Instead, $\dot{\mathbf{q}}$ is used as a synthetic control. The desired value of $\dot{\mathbf{q}}$ can be designed as below to make $\mathbf{S}_1$ converge.

$$\dot{\mathbf{q}}_d = -\mathbf{J}_{rbt}^{\dagger}\mathbf{J}_{img}^{\dagger}\left(\frac{\partial \mathbf{s}_{tgt}}{\partial t} + \mathbf{K}_{kvs}\,\mathrm{sign}\left(\mathbf{S}_1\right)\right) \tag{3.33}$$

where $\bullet_d$ denotes a desired value. $\bullet^{\dagger}$ denotes a generalized or augmented inverse [21]. The gain matrix $\mathbf{K}_{kvs}$ controls the convergence speed of $\mathbf{S}_1$. In order to make tracking error converge, $\dot{\mathbf{q}}_d$ needs to be tracked tightly. This requires a second sliding surface $\mathbf{S}_2 = \dot{\mathbf{q}} - \dot{\mathbf{q}}_d$. Its derivative is

$$\dot{\mathbf{S}}_2 = \ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d = \mathbf{M}^{-1}(\mathbf{q})\left(\boldsymbol{\tau} - \mathbf{C}\left(\mathbf{q}, \dot{\mathbf{q}}\right)\dot{\mathbf{q}} - \mathbf{F}\left(\dot{\mathbf{q}}\right) - \mathbf{G}\left(\mathbf{q}\right)\right) - \ddot{\mathbf{q}}_d \tag{3.34}$$

The actual control of the robot, the torque $\boldsymbol{\tau}$, appears explicitly in $\dot{\mathbf{S}}_2$. It can be designed as below to make $\mathbf{S}_2$ converge.

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{invdy} + \boldsymbol{\tau}_{rbst} \tag{3.35}$$

where

$$\boldsymbol{\tau}_{rbst} = -\mathbf{M}\left(\mathbf{q}\right)\mathbf{K}_{rdc}\,\mathrm{sign}\left(\mathbf{S}_2\right) \tag{3.36}$$

is called a robust term, $\mathbf{K}_{rdc}$ is the gain matrix controlling the convergence speed of $\mathbf{S}_2$, and

$$\boldsymbol{\tau}_{invdy} = \mathbf{M}\left(\mathbf{q}\right)\ddot{\mathbf{q}}_d + \mathbf{C}\left(\mathbf{q}, \dot{\mathbf{q}}\right) + \mathbf{F}\left(\dot{\mathbf{q}}\right) + \mathbf{G}\left(\mathbf{q}\right) \tag{3.37}$$

gives a computed torque using the inverse dynamics of the robot. Equations (3.33) and (3.35) form a cascade controller. Equation (3.33) gives the desired angular velocities of the motors using a model of robot kinematics and a model of camera imaging. It is a kinematic controller conducting the task of *kinematic visual servoing* (KVS). In addition, the desired angular acceleration and rotation can be obtained by differentiating and integrating the desired velocity with the help of dynamic surface control [22]. In this sense, the KVS control law can be considered as an online trajectory generator. It differs from conventional

trajectory planning in terms of using the feedback information in real-time. Meanwhile, Eq. (3.35) determines the torque to actuate the robot to follow the synthetic control given by Eq. (3.33). It can be considered as doing *robot dynamics compensation* (RDC).

A sign function is used for the robustness terms in Eq. (3.33) and (3.36). This is the classic design of sliding mode control. When the system enters the sliding phase, the sign function is known to cause chattering. For smoothing, a popular practice is to replace the sign function with a saturation function or a proportional function [23]. Further, the proportional function can be extended to a general PID form. The RDC control law then becomes similar to a conventional trajectory tracking controller. The only difference is that the computed torque part uses the actual velocity and position feedback instead of the desired values. This will satisfy many end users' preference of utilizing the trajectory tracking controller currently available in their robots.

Addressing Motor Limits

The Jacobian-based KVS control law does not handle the torque and speed limits of the actuators well. In Eq. (3.43), the feedback gain $\mathbf{K}_{kvs}$ determines the convergence speed of the tracking error. Larger gain can shorten the convergence time, but may cause actuator saturation and lead to unexpected move of the robot. This is especially true for industrial robots, whose actuator capability is limited with respect to the inertia of the robot and the mobility of the target. In actual applications, the gain can only be tuned by trial-and-error in an ad-hoc manner. Achieving a good balance is in general difficult.

In order to address actuator limits in a general manner, more sophisticated KVS control law is desired. As introduced, despite its origin from sliding control, the KVS control law can be considered as an online trajectory generator. As a matter of fact, in the few publications discussing actuator limits for visual servoing, online trajectory planning is a dominant approach. The goal of standard trajectory planning for robot manipulators is to drive the end-effector from a home point to a final point. Via-points might also be specified. In order to apply online trajectory planning for visual servoing, it is important to choose a proper final point at each planning period. In [24], point-to-point polynomial trajectories are planned online. Each time, a point on the predicted trajectory of the target is chosen as the final point. The method requires long-term prediction of the target motion, because the duration from the current point to the final point needs to be long enough so that the required motion is not too intense to cause actuator saturation. In many actual applications, however, long-term motion prediction is not possible. In [25] and [26], a fast planning algorithm is used. The current position and velocity of the target are used as the ending states of the trajectory in each planning action. The limit of joint acceleration is used as a constraint. This constraint, however, can hardly assure the avoidance of actuator saturation as robot manipulators feature nonlinear dynamics.

In order to avoid the difficulty of choosing a final point for trajectory planning, the task is formulated into a constrained optimal control problem. As mentioned, each time when a new visual measurement becomes available, the VSDC algorithm introduced in Section 3.2

is used to compensate the sensing latency and estimates the target motion until the next vision measurement becomes available. This action is a blind reckoning based on the past measurement. The estimation for the whole interval can be done at the beginning of each vision sampling period. Using inverse robot kinematics and the inverse model of perspective imaging, the estimated target motion can be converted from the image coordinates to the robot joint coordinates (denoted as $\mathbf{q}_{tgt}$). At the beginning of the k^{th} vision sampling period, an optimal control task is formulated as

$$\min_{\ddot{\mathbf{q}}_d(i)} \lambda \mathbf{e}^T(kN{+}N)\,\mathbf{S}\,\mathbf{e}\,(kN{+}N) + \sum_{i=kN+1}^{kN+N} \mathbf{e}^T(i)\,\mathbf{S}\,\mathbf{e}\,(i) \tag{3.38}$$

subject to

$$\begin{gathered} \begin{bmatrix} \mathbf{q}_d\,(i) \\ \dot{\mathbf{q}}_d\,(i) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & T\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{q}_d\,(i{-}1) \\ \dot{\mathbf{q}}_d\,(i{-}1) \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2}\mathbf{I} \\ T\mathbf{I} \end{bmatrix} \ddot{\mathbf{q}}_d\,(i{-}1) \\ -\bar{\mathbf{v}} \le \dot{\mathbf{q}}_d\,(i) \le \bar{\mathbf{v}} \\ -\bar{\boldsymbol{\tau}} \le \hat{\boldsymbol{\tau}}\,(i) \le \bar{\boldsymbol{\tau}} \\ (i = kN+1 \sim kN+N) \end{gathered} \tag{3.39}$$

where the index i counts the basic control cycles. One vision sampling period is assumed to be N times of the basic control cycle. $\mathbf{I}$ is a 3×3 identity matrix. $\ddot{\mathbf{q}}_d$ is the synthetic control (i.e., the reference motion) to be followed by the RDC control law. $\mathbf{e}{=}\begin{pmatrix} \mathbf{q}_d^T & \dot{\mathbf{q}}_d^T \end{pmatrix}^T{-}\begin{pmatrix} \mathbf{q}_{tgt}^T & \dot{\mathbf{q}}_{tgt}^T \end{pmatrix}^T$ is the tracking error. $\hat{\boldsymbol{\tau}}\,(i) = \mathbf{M}\,(i)\,\ddot{\mathbf{q}}_d\,(i) + \mathbf{C}\,(i)\,\dot{\mathbf{q}}_d\,(i) + \mathbf{G}\,(i)$ is an estimate of the required torque. $\mathbf{S} = \mathrm{diag}\,(\mathbf{I},\ \mu\mathbf{I})$ is a weight matrix. λ and μ are adjustable weights. $\bar{\boldsymbol{\tau}}$ and $\bar{\mathbf{v}}$ are the limits of torque and velocity respectively.

Note that one vision sampling period is not very long. The pose and velocity of the robot, and thus the matrices $\mathbf{M}$, $\mathbf{C}$, and the gravity term $\mathbf{G}$ in the dynamic model, do not change much during the interval. The problem can then by simplified by replacing $\hat{\boldsymbol{\tau}}$ in the constraints with a rough estimate

$$\tilde{\boldsymbol{\tau}}\,(i) = \mathbf{M}\,(kN)\,\ddot{\mathbf{q}}_d\,(i) + \mathbf{C}\,(kN)\,\dot{\mathbf{q}}_d\,(i) + \mathbf{G}\,(kN) \tag{3.40}$$

This is an affine expression. It changes Eq. (3.38) from a nonlinear programming problem to a quadratic programming (QP) problem which features global convexity.

In general, two kinds of solution can be obtained for an optimal control problem [27]. One is the *batch solution*, where the problem is solved offline in advance for the whole control horizon. The solution is a function of only the initial states. The batch solution of a QP problem can be obtained easily using many efficient algorithms (e.g., the conjugate gradient method). The drawback is that the solution is an open-loop control law and is not robust against disturbance and uncertainty. The alternative is the *recursive solution* via repeatedly solving dynamic programming. The result is a closed-form feedback control law that features good robustness. When constraints exist, however, an universal closed-form solution does not exist as in the classic unconstrained linear quadratic case. Parametric optimization has to be used in advance in an ad hoc manner to determine a piece-wise linear feedback law.

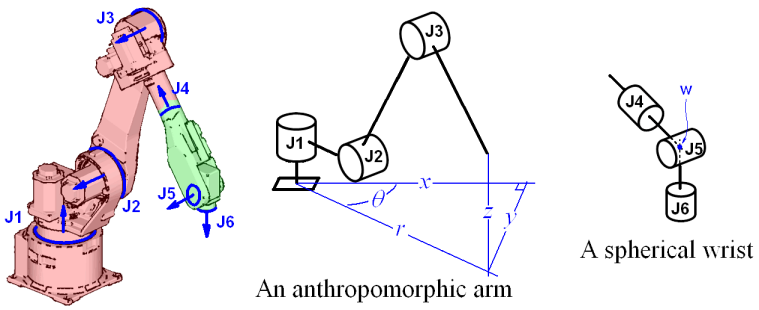


Figure 3.6: The kinematic feature of typical six-axis industrial robot manipulators

The large analysis load is difficult to be implemented in real-time, because the task needs to be done at the beginning of every vision sampling period.

In order to obtain the robustness brought by feedback as well as keep a light computation load, a revised batch solution is developed. It is reasonable to assume that the actual trajectory, even under disturbance and uncertainty, will not deviate significantly from the optimal trajectory. Thus, the batch solution $\ddot{\mathbf{q}}_{d,b}$ can be used as a feedforward control. Meanwhile, a simple state feedback is added to provide robustness, i.e.,

$$\ddot{\mathbf{q}}_d(i) = \ddot{\mathbf{q}}_{d,b}(i) + K \begin{bmatrix} \mathbf{q}_{d,b}(i) - \mathbf{q}(i) \\ \dot{\mathbf{q}}_{d,b}(i) - \dot{\mathbf{q}}(i) \end{bmatrix} \tag{3.41}$$

$\mathbf{q}_{d,b}$, $\dot{\mathbf{q}}_{d,b}$, and $\ddot{\mathbf{q}}_d$ are fed to the RDC control law as the reference (Fig. 3.5).

Tailored Formulation for a Six-axis Robot Manipulator

Special considerations are needed when the control scheme is applied to six-axis robot manipulators such as a FANUC M-16iB robot (Fig. 2.8). For the Jacobian-based KVS control law, a major concern in the formulation is about computing the robot Jacobian in real-time. Usually the robot Jacobian can be computed through differential kinematics. It relates the rotation speed of n joints ($\dot{\mathbf{q}} \in \mathbb{R}^n$) to the linear and angular velocity of the end-effector ($\dot{\mathbf{p}}_{ee} \in \mathbb{R}^3$ and $\dot{\mathbf{r}}_{ee} \in \mathbb{R}^3$). The matrix can be partitioned into two sub-matrices, and

$$\begin{bmatrix} \dot{\mathbf{p}}_{ee} \\ \dot{\mathbf{r}}_{ee} \end{bmatrix} = \mathbf{J}_{rbt}\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J}_{rbt,p} \\ \mathbf{J}_{rbt,r} \end{bmatrix} \dot{\mathbf{q}} \tag{3.42}$$

where $\mathbf{J}_{rbt,p} \in \mathbb{R}^{3\times n}$ relates $\dot{\mathbf{q}}$ to $\dot{\mathbf{p}}_{ee}$, $\mathbf{J}_{rbt,r} \in \mathbb{R}^{3\times n}$ relates $\dot{\mathbf{q}}$ to $\dot{\mathbf{r}}_{ee}$. For a six-axis robot, $n = 6$.

In Eq. (3.33), the robot Jacobian in the Jacobian-based KVS control law needs to be evaluated in every control cycle. This brings much additional computation load. In addition, the singularity of six-axis motion is complicated. When singularity occurs, special

augmentation is needed to find a generalized inverse of the robot Jacobian. In regard to this concern, the kinematic feature of typical six-axis industrial robot manipulators helps. A typical six-axis industrial robot manipulator can be viewed as the combination of an *anthropomorphic arm* and a *spherical wrist* (Fig. 3.6). The first three joints (J1, J2, J3) and links form an anthropomorphic arm, which mainly provides translational motion to the end-effector. The axes of joint J4, J5, and J6 have a single intersection point W (the wrist point). These three joints form a spherical wrist that mainly provides rotational motion to the end-effector. If the wrist point W is chosen to represent the position of the end-effector[1], then $\mathbf{p}_{ee}$ is determined only by the motion of J1, J2, and J3. This makes the last three columns of $\mathbf{J}_{rbt,p}$ zero[2]. Using a cylindrical coordinate system, the kinematics of the wrist point and the corresponding Jacobian $\mathbf{J}_{rbt,p}$ can be easily computed. In the regular workspace of a FANUC M-16iB robot, singularity will not occur for those three joints. As for J4, J5, and J6, they mainly control the rotation of the end-effector. The inverse kinematics of end-effector rotation is a little trickier, but is still straightforward. Given a desired rotation of the end-effector and the joint angles of J1, J2, and J3, finding the joint angles of J4, J5, and J6 can be considered as of finding a set of 1-2-3 Euler angles between two given rotations Q_1 and Q_2, where Q_1 is the rotation caused by J1, J2, and J3, Q_2 is the desired rotation of the end-effector.

With the idea introduced above, a tailored version of the Jacobian-based KVS control law is:

$$\dot{\mathbf{q}}_{d,123} = -\left[\mathbf{J}_{rbt,p}^{-1}\ \mathbf{0}_{3\times 3}\right]\mathbf{J}_{img}^{\dagger}\left(\frac{\partial \mathbf{s}_{tgt}}{\partial t} + \mathbf{K}_{kvs}\mathbf{s}_{tgt}\right) \tag{3.43}$$

where $\dot{\mathbf{q}}_{d,123}$ is the desired velocity of J1, J2, and J3. Meanwhile, the Jacobian-based KVS control law is not used for J4, J5, and J6. Instead, the estimated orientation of the target is used directly as the reference for end-effector rotation, from which the corresponding rotation of J4, J5, and J6 is determined. As introduced earlier, such a direct-tracking strategy is not appropriate for controlling the joints providing large translational motion (J1, J2, and J3). It is, however, fine for controlling J4, J5, and J6, because those joints bear relatively small inertia, and the target rotation tends to be slow in actual applications. This tailored KVS scheme features light computation load, and has no problem of singularity within the regular workspace of the robot.

For the constrained optimal control approach, considerations focus on the relatively complex dynamics of a six-axis robot manipulator. In particular, gravity needs to be included. Even using the recursive Newton-Euler algorithm, computing the matrices $\mathbf{M}$ and $\mathbf{C}$, and the gravity vector $\mathbf{G}$ in Eq. (3.40) requires considerable computation load and can hardly be done in real-time at the desired $1kHz$ sampling rate. In order to reduce computation load, an approximation of the full Lagrangian dynamic model is used. The approximation assumes that the motion of J4, J5, and J6 contributes little to the torque of J1, J2, and J3,

[1]Meanwhile, the actual tool-center-point (TCP) is determined from the wrist point with an offset.

[2]Hereafter in this section, $\mathbf{J}_{rbt,p}$ is used to represent a 3×3 matrix without the zeros.

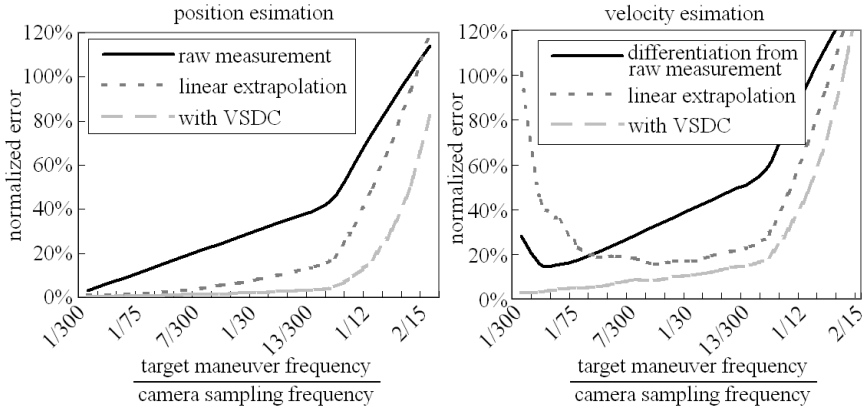


Figure 3.7: Bandwidth evaluation

and their rotation can be set to fixed values when estimating the torques for J1, J2, and J3. In this way, the model can be simplified considerably.

3.4 Results

Bandwidth of the VSDC Algorithm

The bandwidth of the VSDC algorithm introduced in Section 3.2 is examined. The VSDC algorithm is applied to track sinusoidal signals with different frequencies. Since the algorithm is nonlinear, Bode plots are not applicable. Instead, the estimation error versus frequency is plotted. The errors are normalized against the amplitude of the sinusoidal input signals. Instead of the absolute values of the input frequency and the camera sampling frequency, it is the ratio of the two that matters and is used in the plots. Figure 3.7 shows the frequency response represented in this manner. For comparison, the plots also show the performance of a linear extrapolation method, which estimates the target's position using linear extrapolation from the delayed measurements. This method is widely used for picking objects from belt conveyors. The advantage of using a VSDC algorithm is obvious. Compared to linear extrapolation, the proposed method can keep the normalized position error under than 5% in a much wide frequency range. Meanwhile, for velocity estimation, only the propose method can achieve a normalized error lower than 10%.

VSDC algorithm in closed-loop control

Visual servoing tests are conducted to test the effect of the proposed VSDC algorithm in a closed-loop control system. The output of the VSDC algorithm, i.e., the compensated feedback, is fed to the KVS control law introduced in Section 3.3 to drive the robot. The

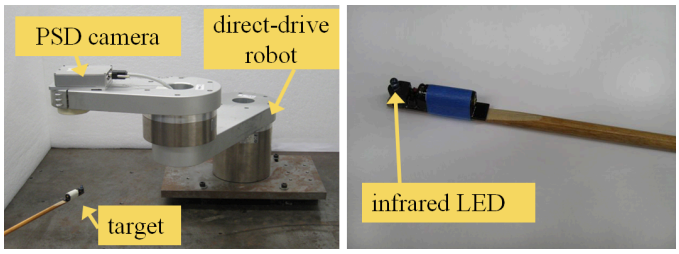


Figure 3.8: A two-axis direct drive robot with a PSD camera and an infrared marker

PSD camera introduced in Section 2.2 is mounted on the end-effector of a two-axis direct-drive planar robot to form an eye-in-hand setup (Fig. 3.8). With proper calibration, it measures the position of an LED infrared beacon with sub-mm accuracy. The output of the PSD camera consists of two analogue voltage signals, representing the horizontal and vertical positions of the marker's projection on the sensing plane. The analogue signals can be sampled at high frequency with little processing. The corresponding sensing latency is insignificant compared to the sampling period. This vision measurement with fast sampling and little latency is used as the baseline for evaluation purpose, whereas the signal feed to the VSDC algorithm is down-sampled to $30Hz$, and delayed by $33ms$. This represents the performance of typical industrial robot vision products. The sampling rate of the control loop is $1kHz$. The control system employs the xPC Target platform developed by MathWorks, with the target computer equipped with a quad-core 3.4GHz CPU and National Instruments 7831R reconfigurable I/O.

The robot is controlled to track an LED infrared beacon moved by a human. The operating person moves the beacon freely at moderate speed. The strong direct-drive motors give the robot a fast dynamic response. In addition, the move of the planar robot is not affected by gravity. Thus, visual sensing dynamics is the major bottleneck in the control loop. In this manner, the effect of a VSDC algorithm can be clearly examined. Figure 3.9 shows the convergence of tracking errors without and with using a VSDC algorithm. The tracking error is significantly reduced. The tracking is also much more stable.

Avoiding Motor Saturation

The constrained optimal control approach introduced in Section 3.3 is tested on the setup shown in Fig. 3.8. Figure 3.10 shows the torque command and tracking error. The torque command still exceeds the limit occasionally. This is due to the replacing of $\hat{\boldsymbol{\tau}}$ in Eq. (3.39) with an approximation. The excess, however, is minor. In general, the torque stays within the limits.

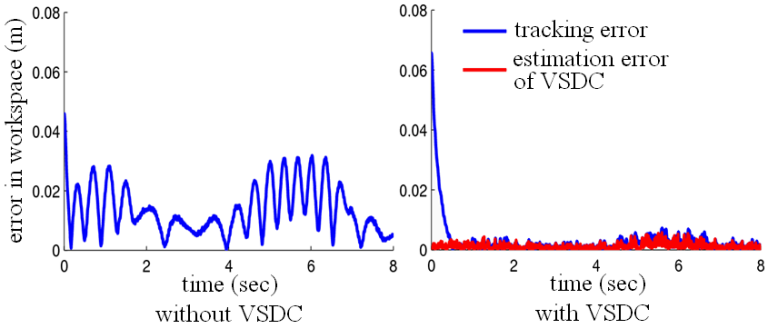


Figure 3.9: Effect of a VSDC algorithm in closed-loop control

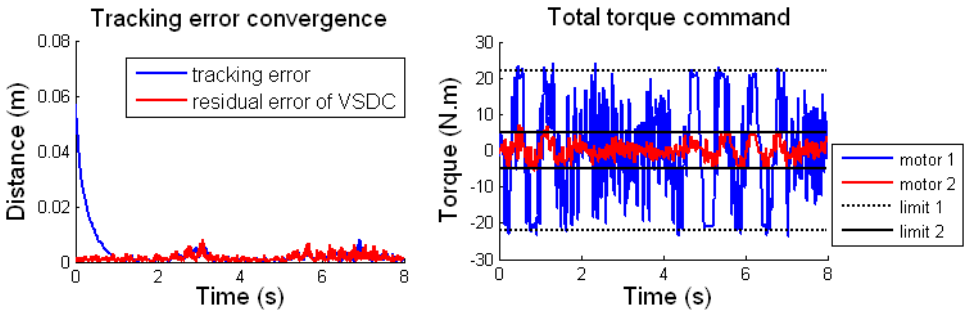


Figure 3.10: Avoiding motor saturation (a two-axis robot)

Extension to a Six-axis Robot Manipulator

The tailored formulation for six-axis robot manipulators is tested using a high-fidelity simulator of a FANUC M-16iB robot (Fig. 2.8). The SimMechanics toolbox in MATLAB is used to build a multibody simulator with detailed dynamic characteristics. High fidelity has been reported in different applications of this simulator [28, 29]. Some dynamic characteristics in the simulator, such as the joint flexibility introduced by indirect drives, are not included in the dynamic model used by the controller, and become a test on the robustness of the proposed scheme. The basic control cycle is $1ms$. In order to achieve this control rate, in addition to the tailoring of the KVS control law, the recursive Newton-Euler algorithm is used to evaluate the computed torque $\boldsymbol{\tau}_{invdy}$ in Eq. (3.35).

The control task is to approach and track a damped swinging target in front of the robot. This mimics a task of picking a swinging workpiece. Figure 3.11 shows the dimensions of the setup. The position of the target is represented by the point that will coincide with the wrist point of the robot when the end-effector goes to the target. Note that this point, though fixed in the corotational frame of the target, may not be on or within the surface of the target. Its position can be computed using the known geometry of the end-effector

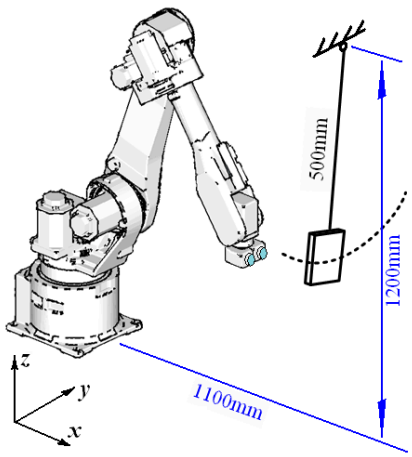


Figure 3.11: A six-axis robot manipulator and a swinging target

and the tool. An eye-in-hand stereo vision sensor is assumed to measure the position and rotation of the target. The large sensing latency ($33ms$) and low sampling rate ($30Hz$) of the vision sensor are compensated by the VSDC algorithm introduced in Section 3.2. Figure 3.12 shows the simulation result. The end-effector approaches and tracks the moving target successfully on both translational and rotational dimensions. The steady state error comes mainly from the residual error of the VSDC algorithm.

One issue of concern in a picking task is controlling the relative approaching direction, so as to prevent undesirable intervention between the tool and the workpiece. When the target is swinging in a y-z plane in the workspace of the robot (Fig. 3.11), a preferred relative approaching direction might be along x direction. This can be realized by adjusting the gain matrix $\mathbf{K}_{kvs}$ in Eq. (3.43). In the simulation shown in Fig. 3.12, $\mathbf{K}_{kvs}$ is a matrix with three identical diagonal elements. That means the approaching velocity in x, y, and z directions will roughly be identical, resulting in an angled relative approaching direction as shown in Fig. 3.13(a). To improve, set the gain corresponding to the x direction smaller than the other two. This setting means the approaching velocity is larger in y and z directions. The end-effector will first move synchronously with the target in y and z directions, then approaches the target along x direction (Fig. 3.13(b,c)). The risk of collision is attenuated.

On the issue of motor saturation, Fig. 3.14 shows an example of undesirable response when the Jacobian-based KVS control law is used and the gain matrix is set blindly without careful tuning. The torque limits of the motors are set to $4Nm$, $4Nm$, $2Nm$, $2Nm$, $1Nm$, and $1Nm$ for J1 to J6 respectively.

Using the simplified Lagrangian dynamic model, the constrained optimal KVS approach is applied. Figure 3.15 shows the result. Because of the error brought by the simplification of the dynamic model, the torque command exceeds the limits to a small extent when large torque is desired, but is still well bounded in general.

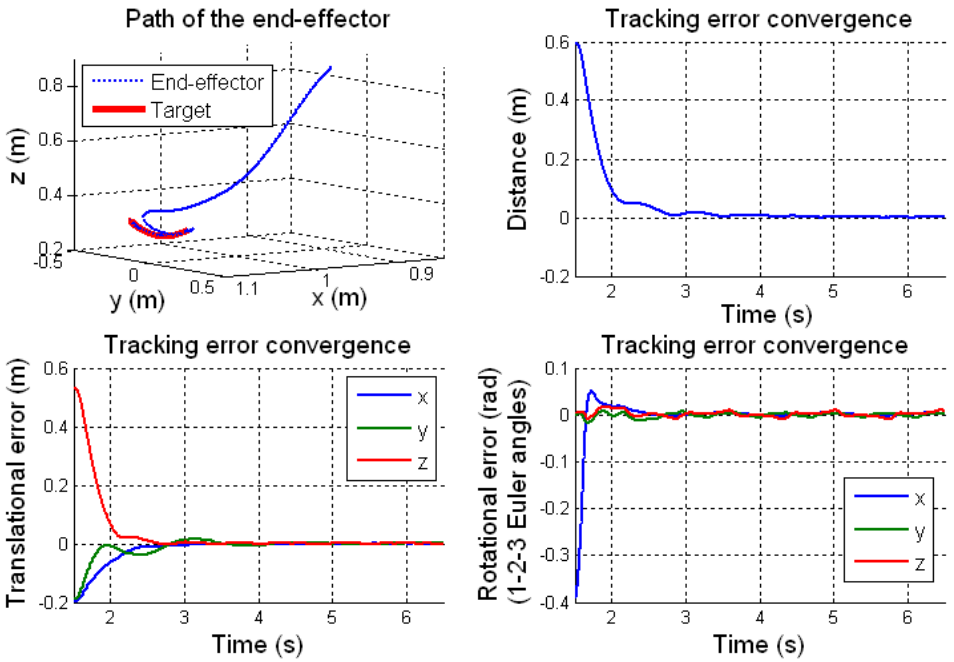


Figure 3.12: Approaching and tracking a swinging target

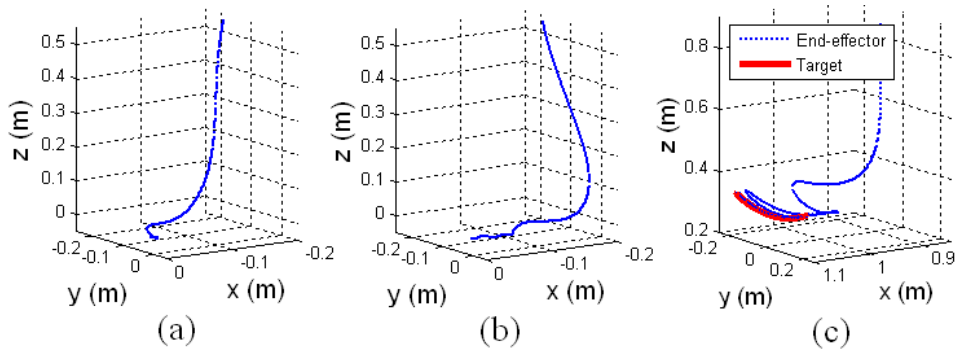


Figure 3.13: Relative approaching directions - (a) The original relative path, (b) The improved relative path, (c) The improved path in the workspace

3.5 Chapter Summary

This chapter presented a statistical learning algorithm to compensate slow feedback signals for real-time motion control. The work is intended to help industrial robots utilize visual feedback for real-time guidance. Due to limit of hardware cost, common vision systems equipped on industrial robots have large latency and low sampling rate, i.e., visual sensing

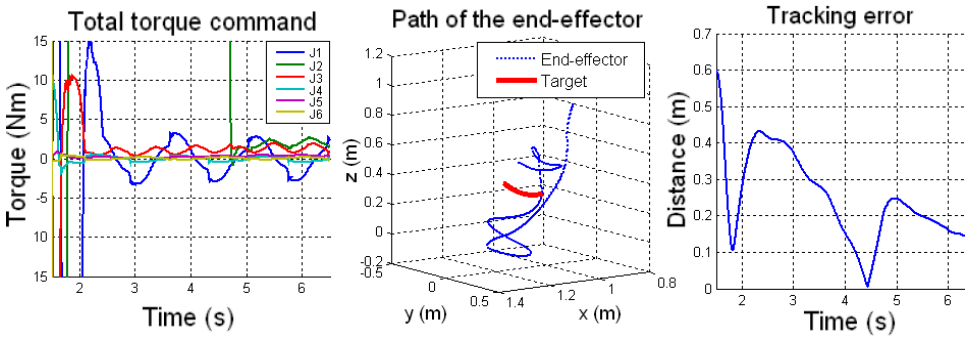


Figure 3.14: Undesirable response caused by motor saturation

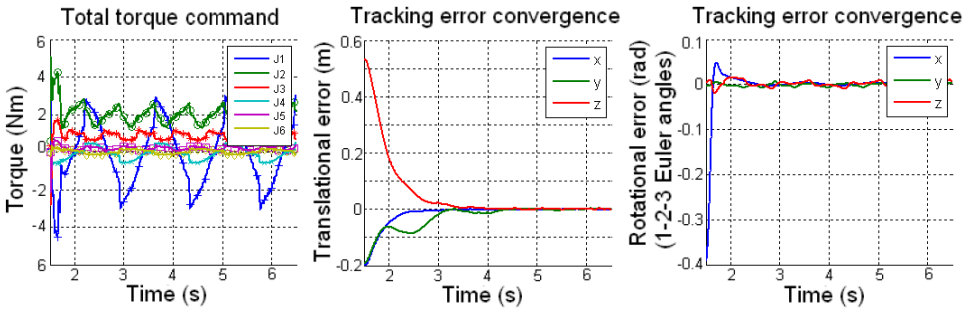


Figure 3.15: Avoiding motor saturation (a six-axis robot)

dynamics. In order to utilize the slow visual feedback to guide real-time motion control, a visual sensing dynamics compensation (VSDC) algorithm was introduced in Section 3.2. A VSDC algorithm compensates sensing latency by using stochastic models to predict the current move of the target from the delayed measurements. The low sampling rate is compensated by interpolation using the model. A statistical method was developed for adaptive model learning.

Section 3.3 discussed the control schemes to utilize the compensated feedback signals for real-time tracking control. A cascade controller structure was proposed. It includes a kinematic visual servoing (KVS) module and a robot dynamics compensation module (RDC). A basic Jacobian-based KVS control law was formulated using sliding control. Then, the speed and torque limits of actuators were addressed using a constrained optimal control approach. The proposed schemes were first applied to a two-axis robot manipulator, then extended to six-axis robot manipulators after special tailoring.

Chapter 4

Nonparametric Learning for Control Compensation

4.1 Introduction

Chapter 2 and Chapter 3 discussed about improving the sensing capability of robots. Data-driven approaches can also be utilized to compensate the control signals of systems with complex dynamics that is hard to model precisely. The method is especially useful for robotic machining. In factory automation systems, robot manipulators and machine tools serve different roles. Although a robot manipulator and a machine tool are both multi-axis servo systems and share similar control hardware, it is significantly more difficult to realize precision contouring control with robot manipulators. This is mainly because machine tools have much higher drive-train stiffness. In addition, the degrees of freedom of a machine tool are arranged in a manner such that basic contours like straight lines and circular arcs can be easily realized by moving as few as one or two motors, whereas even a simple straight line requires that all axes of a robot manipulator move coordinately. Due to these fundamental differences, robot manipulators are conventionally used for material handling, spot welding, and machine tending, but not for machining.

Recent years, however, have witnessed a fast growing trend of robotic machining. Many advantages of robot manipulators, such as large work range and relatively lower cost, are favorable to flexible manufacturing. Although using robot manipulators for turning and milling still suffers fundamental limitations brought by the large cutting force, robotic laser and plasma cutting are becoming increasingly popular. The tracking performance required by laser and plasma cutting tasks is more demanding than that by material handling, spot welding, or machine tending tasks. In the latter cases, there are typically few requirements on contour tracking. The main emphasis is on final positioning, which can be easily achieved based on the high repeatability and careful calibration of the robot. Precision contour tracking, on the other hand, requires advanced real-time control.

The multi-body dynamics of a robot manipulator is very nonlinear and has strong cou-

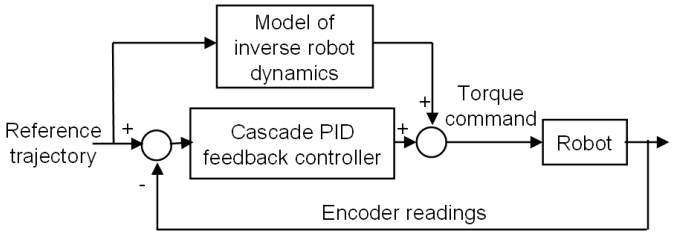


Figure 4.1: Computed torque method

pling among individual axes. The computed torque method [30] has become a major technique for controlling robot manipulators. As illustrated in Fig. 4.1, the computed torque method combines model-based torque feedforward control with decentralized linear feedback control. Accurate torque feedforward control is essential to precision tracking. Dynamics of a robot manipulator, however, is difficult to model and identify precisely. A major challenges lie in the transmission error of the drive-train. As also mentioned in Section 2.1, industrial robot manipulators usually have high gear ratio reducers in the drive-train. This allows the use of high speed and low torque motors which have lower costs and lighter weights. Compliance, backlash, and manufacturing inaccuracy of the reducers, however, introduce transmission error, and the classic multi-rigid body dynamics model cannot describe the actual system behavior perfectly. In addition, the encoders measuring joint rotation are commonly installed on the motor shafts instead of on the output shafts of the reducers. In this way, the resolution of the encoders (in terms of measuring joint rotation) can be effectively scaled up by the gear ratio of the reducer. Due to the large reduction ratio, however, deflection caused by the reducer compliance can barely be perceived by the encoders on the motor side [31]. While contour tracking usually requires sub-millimeter accuracy, transmission error and other uncertainties in the kinematic chain can easily cause several millimeters of deviation at the tool. According to [32], 8% to 10% of the end-effector tracking error comes from drive-train compliance. Precision measurement of the tool motion therefore requires additional sensing such as a laser tracker or a machine vision sensor.

This chapter presents a method to realize precision robot tracking control based on trajectory or contour inspection. The method features data-driven iterative compensation that benefits from the repetitive nature of industrial robots' motion. Nonparametric statistical learning is used to compensate for both the feedforward torque and the motor reference. Motor side tracking and uncertainties in kinematic chain are handled by separate learning modules in a two-part compensation structure. Compensation methods based on different setups of end-effector sensing are discussed. In the following sections, different aspects of the method are introduced in detail. Previous work related to each aspect of the proposed method is reviewed in their respective sections. Experimental validation using a benchmark hole cutting trajectory is discussed at the end.

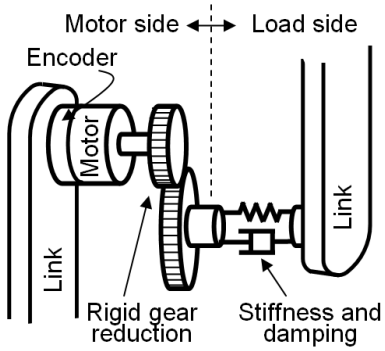


Figure 4.2: A spring-damper model for geared joints of robot manipulators

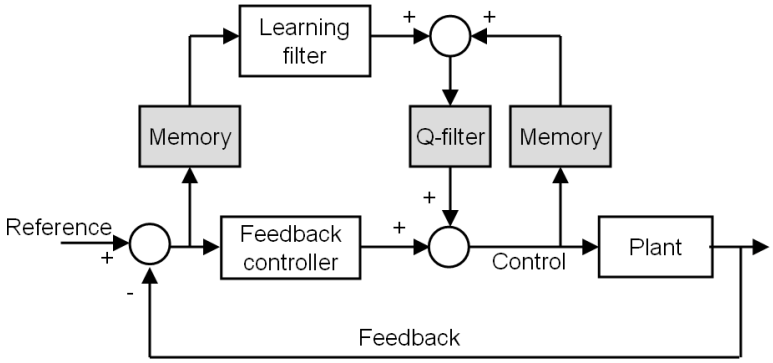


Figure 4.3: Iterative learning control

4.2 Data-driven Compensation through Learning

Like a human worker refines his skills through experience, machines that execute repetitive tasks can improve its performance by learning from data collected in previous executions. A classic method based on this idea is iterative learning control (ILC). As shown in Fig. 4.3, the feedforward control is iteratively learned from previous tracking errors by using a learning filter, whose design is based on a convergence condition. Specifically, the convergence conditions of most ILC methods have a similar form as [33, 34, 35]

$$\rho\left(Q\left(I - LP\right)\right) < 1 \tag{4.1}$$

where $\rho(\bullet)$ is a norm function, I is an identity matrix. L, P, and Q are the descriptions (either in time domain or frequency domain) of the learning filter, the actual system dynamics, and a Q-filter respectively. The low-pass Q-filter improves the transient learning behavior and robustness by restraining the learning activity within finite frequency range. A convergence condition in the form of Eq. (4.1) tolerates uncertainty of the actual system

behavior to a certain extent. Effectiveness of disturbance rejection, however, still depends much on how well the actual system behavior is known. Instead of refining feedforward control based on available knowledge of the system response, adaptive control techniques refine the system model online. The dynamic model of a tree-like multi-body system is linear with respect to the model parameters [36]. This property allows data-driven online estimate of the model parameters using least squares regression [37]. The multi-rigid body dynamics model, however, cannot accommodate many complex characteristics of the drive-train, such as the transmission error caused by joint compliance, backlash, and manufacturing inaccuracy. Meanwhile, a more sophisticated model that attempts to include those factors always turns out to be overly complicated and difficult to identify.

Nonparametric Statistical Learning

The limitation of parameterized models motivates the application of nonparametric statistical learning methods. Rather than indicating that the methods are parameter-free, the term nonparametric means that the mapping from input to output (also called a target) is learned without assuming a model with specific parameterized structure. To avoid confusion, the parameters used in nonparametric learning are called hyper-parameters. A representative method in this class is locally weighted regression (LWR) [38]. [39] presents a comprehensive discussion on using LWR for learning the inverse dynamics of robot manipulators. One disadvantage of LWR is the use of a large number of hyper-parameters which are difficult to tune [40].

In recent years, Gaussian process regression (GPR) is becoming increasingly popular due to simple implementation and reliable hyper-parameter tuning. Consider a mapping f from a vector input $\mathbf{x}$ to a scalar output $y = f(\mathbf{x}) + \varepsilon$, where ε is the sensing noise with a covariance σ_ε^2. Note that the concepts of input and output used here should not be confused with those of general dynamic systems. Rather than assuming a parameterized structure for f, GPR assumes f to be random, and can be characterized using a mean function and a covariance function (also called a kernel) [41]:

$$\begin{aligned} m(\mathbf{x}) &= E\{f(\mathbf{x})\} \\ k(\mathbf{x}_i, \mathbf{x}_j) &= E\{(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))\} \end{aligned} \tag{4.2}$$

The kernel characterizes the correlation among input variables. If little a priori information is known about the mapping, a zero mean and a Gaussian kernel are often assumed. Then, given a set of training data points $\{(\mathbf{x}_{t,i}, y_{t,i}) | i = 1 \sim n\}$ collected from previous measurement, and a set of query points $\{(\mathbf{x}_{q,i}, y_{q,i}) | i = 1 \sim m\}$ whose output $y_{q,i}$'s are to be inferred, the joint distribution of the training data and query points is

$$\begin{pmatrix} \mathbf{y}_t \\ \mathbf{f}_q \end{pmatrix} \sim N\left(\mathbf{0}, \begin{pmatrix} K_{t,t} + \sigma_\varepsilon^2 I & K_{t,q} \\ K_{q,t} & K_{q,q} \end{pmatrix}\right) \tag{4.3}$$

where $K_{\bullet,*}$ denotes the covariance matrix of $X_\bullet$ and X_*. X_t, X_q, $\mathbf{y}_t$, and $\mathbf{f}_q$ are aggregations of $\mathbf{x}_{t,i}$'s, $\mathbf{x}_{q,i}$'s, $y_{t,i}$'s, and $f(\mathbf{x}_{q,i})$'s respectively. Given X_q, X_t, and an actual observation of

$\mathbf{y}_t$, the conditional distribution of $\mathbf{f}_q$ has a mean

$$\bar{\mathbf{f}}_q = K_{q,t}\big(K_{t,t} + \sigma_\varepsilon^2 I\big)^{-1}\mathbf{y}_t \tag{4.4}$$

and a conditional covariance

$$\operatorname{cov}(\mathbf{f}_q) = K_{q,q} - K_{q,t}\big(K_{t,t} + \sigma_\varepsilon^2 I\big)^{-1}K_{t,q} \tag{4.5}$$

The conditional mean $\bar{\mathbf{f}}_q$ serves as the estimate of the output of the query points. The conditional covariance $\operatorname{cov}(\mathbf{f}_q)$ gives the confidence interval of the estimate. Note that if the kernel function indicates that a query point is barely correlated to any training point, GPR still gives an estimate, however, with a wide confident interval. In this sense, GPR does not require the execution to be repetitive (whereas ILC does). Ideally, if the data is collected from all regions in the space spanned by the motion variables, all motion of the robot can be covered. Learning with GPR mainly includes determining a proper kernel and tuning the corresponding hyper-parameters. Tuning can be done either by cross validation or by optimizing the marginal likelihood function of the estimation. Because a servo system always follows certain trajectories, the collected data often features nonuniform and continuous distribution in the input space, making it difficult to find a good partitioning of the training data for effective cross validation.

Some related work of applying nonparametric statistical learning to robot control includes learning inverse robot dynamics [40, 39], and learning kinematic controllers [42, 43]. Compared to these work, the approach introduced here features applying GPR in a two-part compensation structure, where learning is conducted on both the dynamic and kinematic level. Moreover, in addition to learning based on timed trajectory measurement, a learning scheme based on untimed two-dimensional contour inspection is presented. Such a learning scheme is more favorable for the typical inspection systems in actual industrial setups.

Two-part Compensation for Low-stiffness Servo Systems

The control of flexible motion systems has long been in the spolight, especially in the area of controlling construction machines such as cranes and loaders. A widely adopted method is input shaping [44, 45]. Input shaping utilizes a model of the flexible drive-train to modify the reference signal to the actuators so that the end of the drive-train moves as desired without vibration. As an example, Fig. 4.2 shows a simplified model of a geared robot joint. The relationship between motor side motion and load side motion can be described as

$$k_J\left(\frac{\theta_M}{r} - \theta_L\right) + d_J\left(\frac{\dot{\theta}_M}{r} - \dot{\theta}_L\right) = \tau_L \tag{4.6}$$

where θ_M is rotation angle on the motor side, θ_L is the rotation angle on the load side, r is the gear ratio, k_J is the stiffness, d_J is the damping ratio, and τ_L is the torque applied on

the load side. Rearrange Eq. (4.6) to the form of a first order system with θ_M as the state variable:

$$\dot{\theta}_M = \frac{k_J}{d_J}\theta_M + u_{\theta_M} \tag{4.7}$$

where

$$u_{\theta_M} = r\left(\frac{k_J}{d_J}\theta_L + \dot{\theta}_L + \frac{\tau_L}{d_J}\right) \tag{4.8}$$

is the input of the first order system. Given a desired trajectory $\theta_{L,d}(t)$ on the load side, the load side torque τ_L can be computed using the inverse multi-body dynamics model of the robot. The shaped motor reference $\theta_{M,d}(t)$ can be obtained by integrating Eq. (4.7) with respect to time. This method works very well in computer simulations. However, implementing model-based input shaping on industrial robots faces two limitations. First, shaping the reference of the motors does not guarantee a perfect tracking of it. Due to complex multi-body dynamics, it is difficult for the servo system to track the shaped reference perfectly. Second, it is difficult to model and identify the characteristics of the joints of an industrial robot accurately. For many systems whose response is dominated by pure flexibility, such as the arm of a crane or a loader, a classic spring-dumper model can effectively cover the majority of the drive-train behavior. For the joints of an industrial robot, however, their response is a mix of pure flexibility and many other characteristics such as backlash and assembly misalignment. It is difficult to model and identify the overall behavior accurately using analytical models.

To overcome these limitations, iterative compensation can be used. [46] proposes a two-stage iterative learning control (ILC) method, where in one stage the torque signal is updated to make the plant behave like a nominal model, and in the second stage the motor reference is updated to address the transmission error. A somewhat similar structure is adopted here. As shown in Fig. 4.4, the motor reference compensation module compensates (shapes) the motor reference, whereas the torque compensation module corrects the feedforward torque so that the compensated motor reference can be tracked closely. The compensations are updated iteratively based on data collected through repetitive executions.

Data Selection for Efficient Learning

The computation load of GPR grows with respect to the size of the dataset. In particular, computing the matrix inverse in Eq. (4.4) introduces a basic complexity of $\mathrm{O}(N^3)$, where N is the size of the dataset. As a robot runs for a longer time and more data is collected, it becomes impractical to conduct learning using all the accumulated data. GPR predicts the output of a query point based on the correlation between the input vectors of the data points and the query point. From this point of view, for a given query point, it is reasonable to select a certain number of data points (often called an active set) whose input vectors are close to the input vector of the query point. This can be realized by storing the data points using a space-partitioning data structure such as a kd-tree and applying a k-nearest neighbors (k-NN) search algorithm [47]. In addition, because factors such as the environment

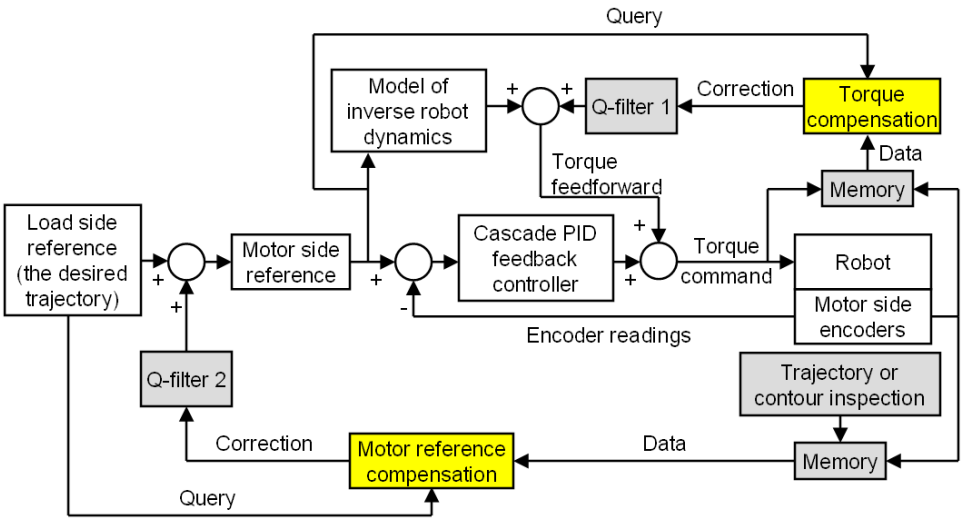


Figure 4.4: A two-part compensation structure

temperature might change considerably during a day, the mechanical characteristics of a robot varies through time. Thus, it is reasonable to use the data collected in more recent iterations instead of those in older iterations.

4.3 Compensation of Feedforward Torque

In the computed torque method, usually a model of inverse multi-rigid body dynamics (often in the form of Lagrange's equation of motion) of the robot is used to compute the feedforward torque of the motors:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{4.9}$$

where $\boldsymbol{\tau}$ is the torque exerted by the motors, $\mathbf{q}$ is the rotation of the motors, $\mathbf{M}$ is the inertia matrix, $\mathbf{C}$ gives the centrifugal and Coriolis effect, $\mathbf{F}$ is friction, and $\mathbf{G}$ is gravity. Such a model of multi-rigid body dynamics has limited fidelity to the actual system response. GPR can be applied to learn a nonparametric mapping from the motion variables $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ to the torque $\boldsymbol{\tau}$. The measurements from previous executions are used as training data, with the motion variables as input and torques as output. The motion variables of the desired trajectory are the inputs of the query points. An anisotropic exponential kernel is used to characterize the correlation between input variables:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_k^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^{\mathbf{T}} W (\mathbf{x}_i - \mathbf{x}_j)\right) \tag{4.10}$$

where the input $\mathbf{x} = \begin{pmatrix} \mathbf{q}^{\mathbf{T}} & \dot{\mathbf{q}}^{\mathbf{T}} & \ddot{\mathbf{q}}^{\mathbf{T}} \end{pmatrix}^{\mathbf{T}}$ contains the motion variables. W is a weight matrix that allows the characterization of anisotropic relevance among the different dimensions of

$\mathbf{x}$. Automatic relevance determination (ARD) [48] can be used to set W. σ_k is a scaling factor. It can be tuned by optimizing the marginal likelihood function of the estimation using gradient-based numerical optimization. A good discussion can be found in [49].

It is, however, unrealistic to expect perfect learning of the full robot dynamics using an exponential kernel. In Section 5.4.3 of [41], an example shows the poor performance of using an exponential kernel to learn from the samples of a step function, which is the basic model of Coulomb friction. More sophisticated kernel functions such as a neural network kernel might give better learning results, but also introduces significantly more hyper-parameters and thus makes the tuning process tricky. It is therefore desirable to incorporate available analytical models into GPR. Instead of assuming a zero mean, the nominal model of multi-rigid body robot dynamics with gravity and friction compensation (as described by Eq. (4.9) is used as a mean function in GPR to give a rough prediction at the query points. In this manner, GPR is used to learn the difference between the nominal model and the actual response. If a query point is far away from all training data, instead of giving zero (as in the case of assuming a zero mean), the prediction will degenerate to the value evaluated by the nominal model.

Instead of applying the learned feedforward torque directly to the next execution, as shown in Fig. 4.4, a low-pass Q-filter is used to restrain the compensation in a finite frequency range. The application of Q-filters has been a common practice in iterative learning control and disturbance observer. It improves the robustness of the compensation action [35]. Here, a fourth-order Butterworth filter with a cutoff frequency of 35Hz is used for torque compensation.

4.4 Compensation of Motor Reference

Because of the complex drive-train behavior, it is infeasible to estimate the motion of the tool accurately using only motor encoders. Thus, the compensation of motor reference requires additional sensing to inspect the tracking performance at the tool. Depending on the inspection method, the technique for compensation differs. In terms of assisting control, the ideal inspection sensor is a position tracker such as the CompuGauge cable measure system developed by Dynalog or the laser tracker system developed by FARO. A position tracker records the motion of the robot end-effector with timestamp. Despite its usefulness, a motion tracker is often not applicable when a robot is in actual production activities. The tool and the workpiece will interfere with the cables of a CompuGauge system or block the laser beam of a laser tracker. A commonly available option is after-run inspection such as that provided by a coordinate measuring machine or a machine vision system[1].

[1]Actually, in addition to providing single-frame inspection, a machine vision system can also capture consecutive images and potentially provide continuous timed measurement to serve as a tool tracker. The sampling rate of current industrial machine vision systems, however, is often too limited to serve the purpose. The fast motion of a robot and the sparks in laser or plasma cutting also make it very difficult to capture a clear picture.

Compensation Based on Timed Trajectory Measurement

For the sake of clarity, it is necessary to briefly review a few concepts of paths and trajectories of robot manipulators. The term path refers to a geometric description of the motion, whereas a trajectory refers to a path with a specified timing law [30]. In the field of machining, the term contour and profile often refer to a path. Paths and trajectories can be defined either in the joint space to describe the rotating motion of the joints, or in the Cartesian space to describe the translation and rotation of the tool. In the joint space, due to transmission error, the paths and trajectories on the motor side do not simply equal to their counterparts on the load side multiplied by gear ratios. For the purpose of clear indication, the terms motor path/trajectory and tool path/trajectory are used hereafter. The term motor-equivalent (M-E) tool path/trajectory is used to indicate a tool path/trajectory inferred from the motor path/trajectory using the forward kinematic model of the robot. For an ideal multi-rigid body robot, the M-E tool path/trajectory is the same as the tool path/trajectory. For an actual robot, however, due to model inaccuracy and transmission error, they are different. In summary, there are three types of paths/trajectories used – 1) motor path/trajectory, 2) tool path/trajectory, and 3) M-E tool path/trajectory. For each of them, a reference can be specified. Due to imperfect tracking, the actual paths/trajectories always deviates from the reference paths/trajectories.

In the case where a motion tracker is available to give timed measurement of the actual tool trajectory, GPR can be applied to update the reference M-E tool trajectory by learning from the difference between the actual M-E tool trajectory and the actual tool trajectory [42]. The configuration of the training data and query points is less straightforward than the case of torque learning. Table 4.1 gives a summary. Unlike the case for torque compensation, the motion variables used here are the position, velocity, and acceleration of the tool in the Cartesian space. In every iteration of robot motion, the updated reference M-E tool trajectory is converted to the reference motor trajectory using inverse robot kinematics. Cascade PID feedback control, torque feedforward control, and the torque compensation method introduced in Section 4.3 are used to drive the motors to track the reference motor trajectory. If the resulting actual motor trajectory is close to the reference motor trajectory, the corresponding actual tool trajectory should be close to the reference tool trajectory. A fourth-order Butterworth filter with a cutoff frequency of 10Hz is applied as a Q-filter.

Compensation Based on Untimed Two-dimensional Contour Inspection

Assuming that a machine vision system is used, after the machining task, a camera takes a picture of the finished workpiece. Then, an image processing algorithm identifies the two-dimensional contour of the actual tool path and returns the two-dimensional coordinates of sampling points distributed along the contour. Unlike the timed measurement provided by a motion tracker, the measurements from an after-run inspection only include the coordinates of the selected measurement points. Without time information, the measured points along

Table 4.1: Motor reference compensation using trajectory measurement

	Training	Query
Input	Motion variables of the actual tool trajectory in previous executions	Motion variables of the reference tool trajectory
Output (target)	Motion variables of the actual M-E tool trajectory in previous executions	Motion variables of the reference M-E tool trajectory

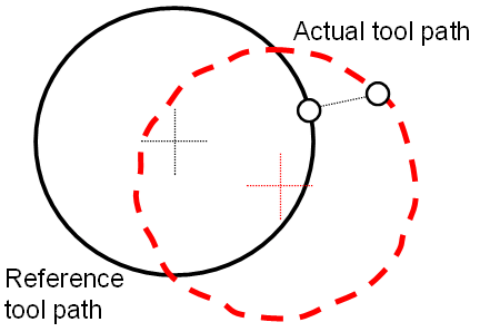


Figure 4.5: Improper matching of paths

tool path cannot be matched to the sampling points of the trajectory, and the motor reference cannot be compensated directly. [50] presents a method of correcting the tool path of repetitive machining tasks for machine tools. The correction is based on the finish inspection of the previous machined workpiece using a coordinate measuring machine. The tool path is corrected by simply flipping the contour errors to the opposite side of the desired path and adding them to the motor reference. Inspired by this method, statistical learning is applied first to compensate the tool path. Then, the trajectory is regenerated from the compensated tool path.

In order to apply the proposed learning scheme, it is first necessary to match the points on the reference tool path, actual tool path, and M-E tool path. One way of matching is to project the sampling points on the reference tool path to the other paths along the normal direction. The distance between the matched points on the reference path and the actual path is called contour error or profile error, as opposed to the tracking error computed from the timed trajectory measurement. It is one of the primary evaluation indices in the field of machining[51]. Matching directly with normal projection, however, might give misleading results under workpiece positioning error or inaccurate kinematic calibration. As shown in Fig. 4.5, in terms of shape accuracy, the actual path actually resembles the reference

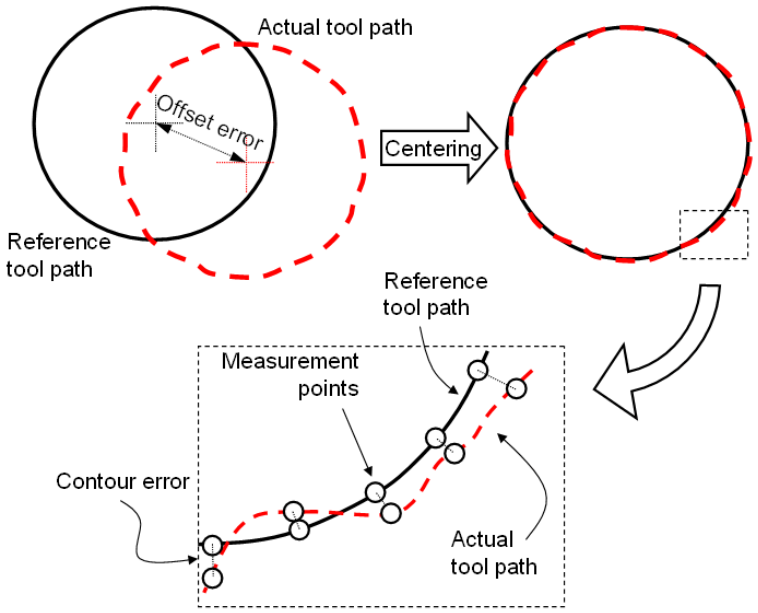


Figure 4.6: Matching the actual path to the reference path after centering

path closely. A direct matching of the points on the normal direction, however, indicates very large contour errors. A better way is to introduce offset error as the general deviation between the paths, and matching the points using normal projection after centering the two paths. The offset error is another important index besides contour error. The center of a path can be located by computing the centroid of a polygon whose vertices are defined by the sampling points of the path. An introduction on computing the centroid of polygons can be found in [52]. After the paths are matched point by point, GPR can be applied to shape the reference M-E tool path. The training data and query points are configured as in Table 4.2.

The reference M-E tool path is shaped by subtracting the predicted contour error from the reference tool path. Then, the M-E reference tool trajectory can be re-generated by parameterizing the path using its arc length l. For machining tasks, often a constant feed rate along the path is desired. As shown in Fig. 4.7, this is realized by timing the arc length with an S-curve. Depending on the given limits of acceleration and jerk, the S-curve features an accelerating and a decelerating phase before and after the constant speed portion. A fourth-order Butterworth filter with a cutoff frequency of $10Hz$ is applied as a Q-filter to the tool trajectory. Finally, the tool trajectory generated in this manner is converted to motor trajectory using the inverse kinematic model of the robot.

Table 4.2: Motor reference compensation using contour inspection

	Training	Query
Input	Coordinates of the actual tool path in previous executions	Coordinates of the Reference tool path
Output (target)	Contour error between the actual M-E tool path and the actual tool path in previous executions	Contour error between the reference M-E tool path and the reference tool path

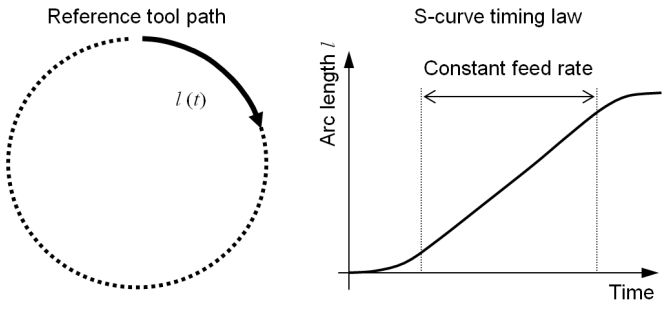


Figure 4.7: Generating trajectory from path

4.5 Results

A FANUC M-16iB robot (Fig. 2.8) is used for experimental validation. A payload of $16kg$ is mounted on the end-effector to represent the weight of a tool. The CompuGauge cable measure system developed by Dynalog is used to directly measure the motion of the end-effector. Figure 4.8 depicts the setup of the control system. The xPC target system developed by MathWorks is used to implement the proposed control scheme. All real-time control algorithms are deployed on the target PC, which uses FANUC PCI interface to communicate with a FANUC digital servo adapter (DSA). The sampling rate for control and sensing is $1kHz$. The robot is controlled to finish a circular tool path with one-inch diameter in one second. This is one of the commonly used benchmark tests to evaluate a robot's capacity for laser and plasma cutting.

Figure 4.9 shows the result when only torque compensation is applied. The tracking performance of the motors is evaluated by computing the difference between the reference tool trajectory and the actual M-E tool trajectories (Plot (a)), where the latter is computed from the actual encoder readings. Without motor reference compensation, the reference tool

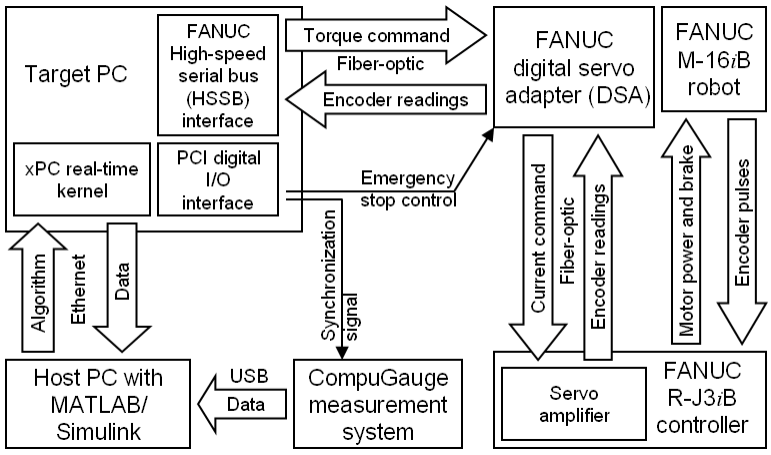


Figure 4.8: Deployment of the control system

trajectory is the same with the reference M-E tool trajectory. The tracking performance at the tool-center-point is evaluated by computing the distance between the reference and actual tool trajectories (Plot (b)), where the latter is measured by the CompuGauge system. Plot (c) compares the convergence of the average and maximum tracking errors on the motor side and load side. It can be seen that with torque compensation alone, the motor reference can be tracked closely after a few iterations of learning, whereas the actual tool trajectory still deviates from the reference to a nontrivial extent because of the uncompensated transmission error.

Figure 4.10 shows the result when the complete two-part compensation (Fig. 4.4) is applied. The motor reference is compensated based on trajectory measurement provided by CompuGauge. Within a few iterations, the tracking error at the tool-center-point can be reduced from above two millimeters to sub-millimeter levels. Meanwhile, compensation of the motor reference results in a nontrivial difference between the reference tool trajectory and the reference M-E tool trajectory.

Instead of using timed trajectory measurement, Fig. 4.11 shows the result when only untimed two-dimensional contour inspection is available for motor reference compensation. The contour measurement includes 50 points uniformly distributed along the circular path. The two-dimensional coordinates of the 50 points are obtained from the CompuGauge measurement (around 1000 points in total) by dropping the time information and depth measurement, and interpolating the left two-dimensional coordinate data at the 50 selected locations. Within a few iterations, both the contour error and the offset error converge to a much reduced level.

Figure 4.12 shows the finished tool path without compensation. Fig. 4.13 and Fig. 4.14 show the finished tool path with compensation based on timed trajectory measurement and untimed two-dimensional contour inspection respectively. It can be seen that when only

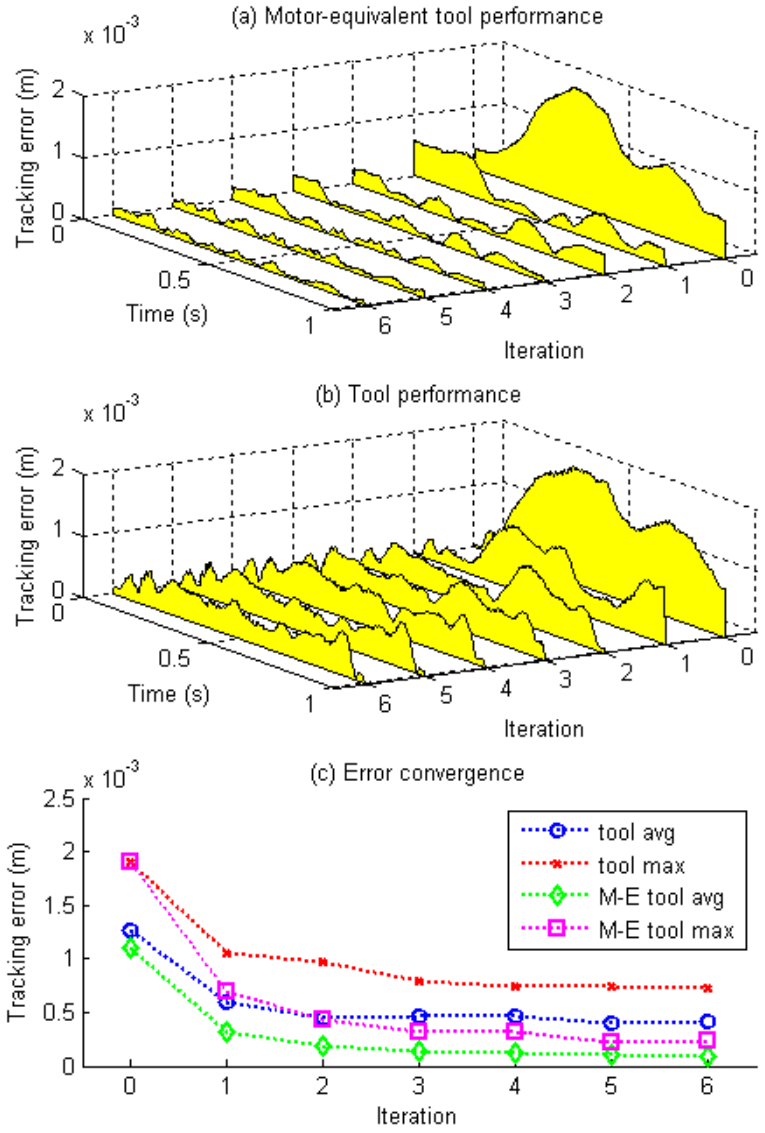


Figure 4.9: Tracking performance with torque compensation only

untimed two-dimensional contour inspection is available, the residual error on the depth direction (the x direction in the figure) is larger than that when timed trajectory measurement is available. Without depth measurement, the motion on the depth direction is not compensated directly. The improvement on the depth direction comes from the coupling between Cartesian dimensions in the inverse robot kinematics.

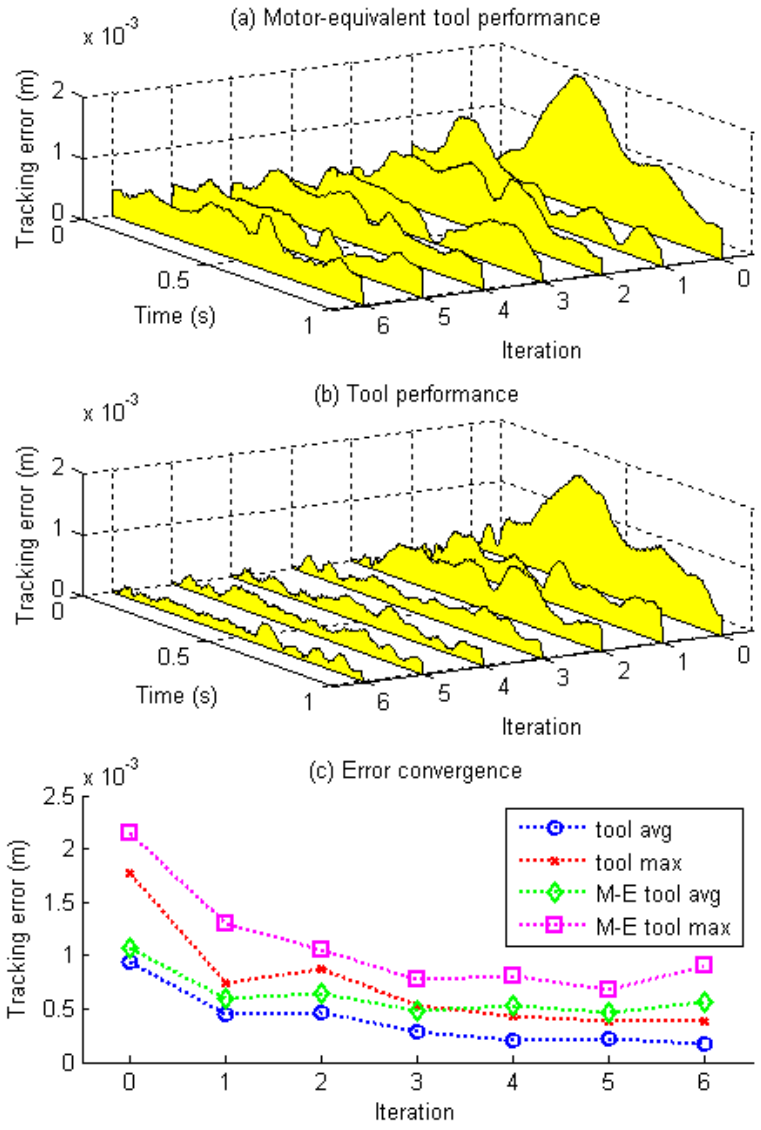


Figure 4.10: Tracking performance with two-part compensation - with trajectory measurement

4.6 Chapter Summary

This chapter presented an approach of nonparametric learning control and its application to precision tracking control of robot manipulators. The approach features data-driven iterative compensation that benefits from the repetitive nature of industrial robots' motion. A two-part compensation structure is proposed, in which motor side tracking and transmission error are handled by separate learning modules. Depending on the specific setup of end-

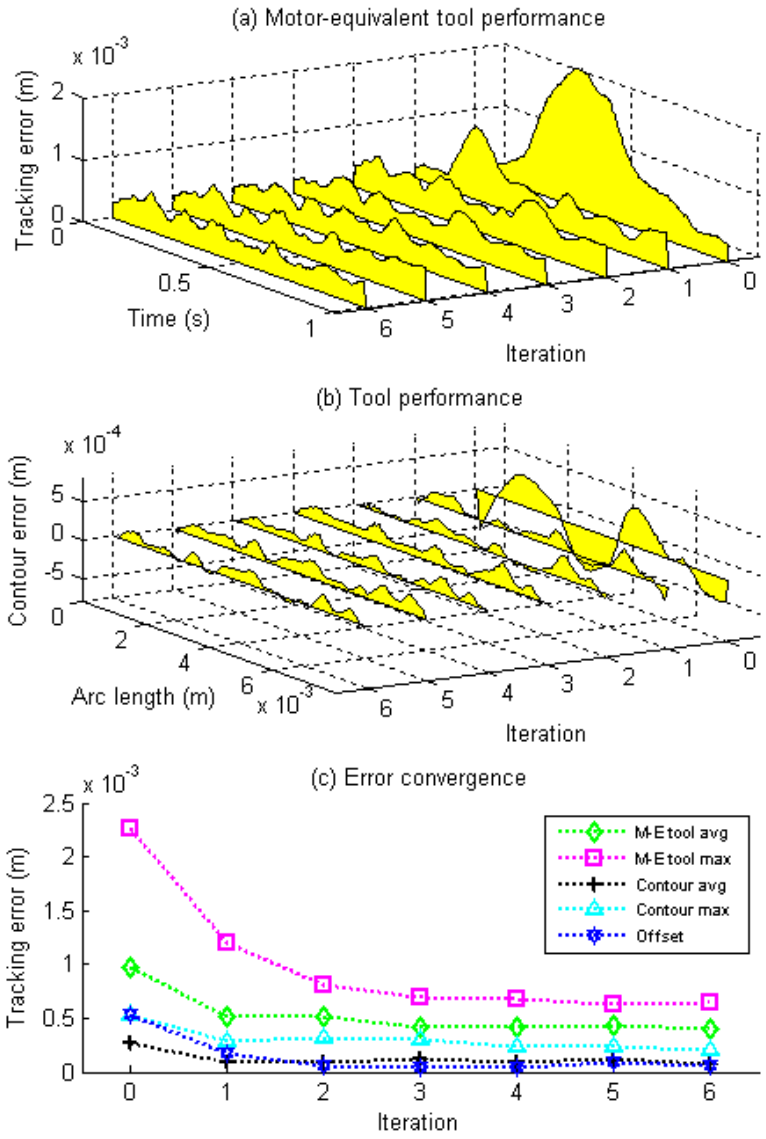


Figure 4.11: Tracking performance with two-part compensation - with contour inspection

effector sensing, either timed trajectory measurement or untimed two-dimensional contour inspection can be used for the compensation. Nonparametric statistical learning was applied using Gaussian process regression. Considerations on incorporating analytical models and selecting data subsets for more efficient learning were discussed. Experimental validation was conducted using a six-axis robot manipulator. Experimental results showed that the proposed method for torque compensation can significantly improve motor side tracking performance. Meanwhile, the motor reference compensation scheme can handle transmission error effectively. For a benchmark hole cutting trajectory with a one-inch diameter and

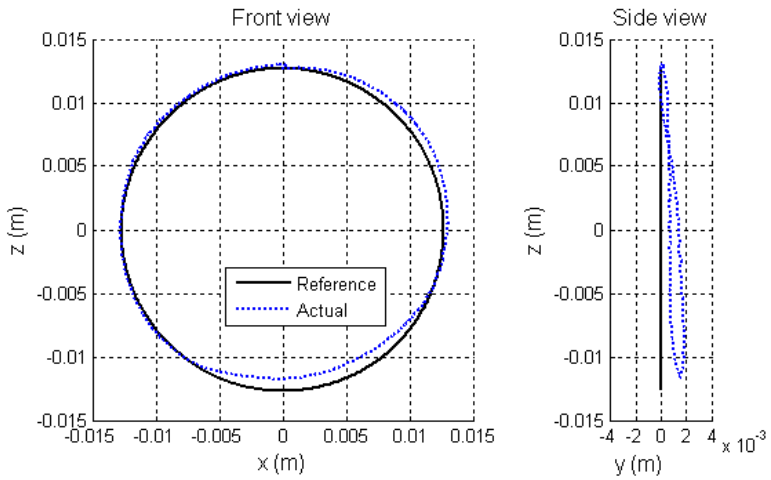


Figure 4.12: Actual tool path with no compensation

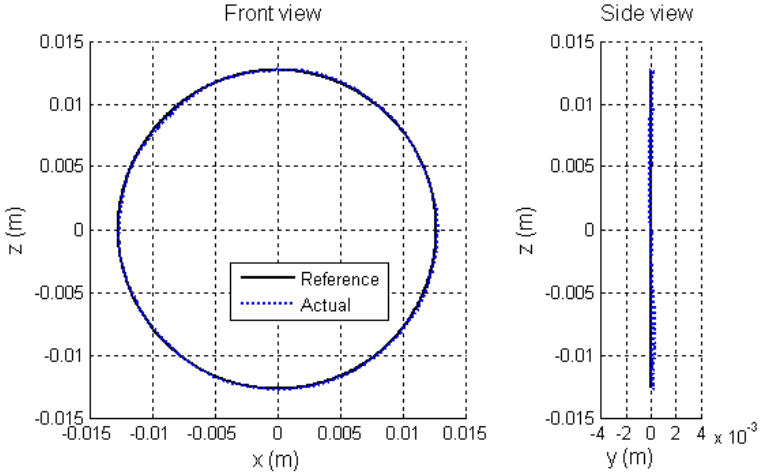


Figure 4.13: Actual tool path with compensation based on timed trajectory measurement

one-second duration, the tracking error can be reduced from above two millimeters to sub-millimeter levels within a few iterations.

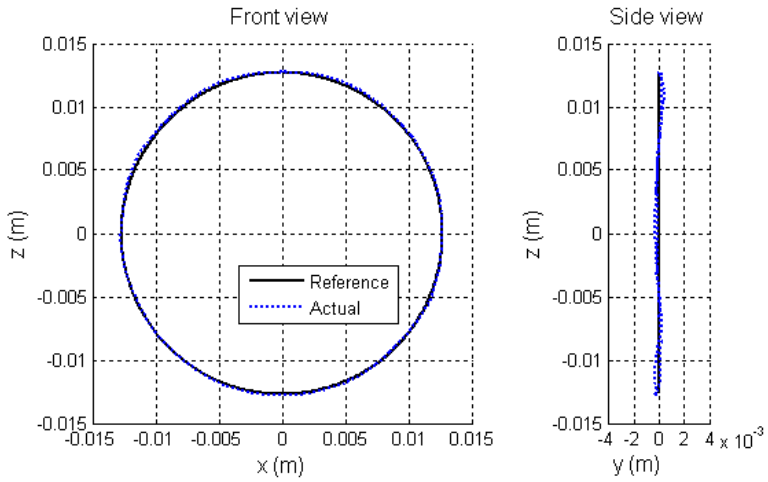


Figure 4.14: Actual tool path with compensation based on untimed 2D contour measurement

Chapter 5

Intelligent Modeling and Identification

5.1 Introduction

Throughout various work on sensing and control of robot manipulators, proper modeling has always been playing an indispensable role. Torque feedforward control based on an accurate dynamic model can greatly reduce the trajectory tracking error of servo systems. In iterative learning control and disturbance observer control, the bandwidth of effective disturbance rejection is heavily determined by the accuracy of the model. In visual servoing, an inaccurate camera model might lead to undesired response and even instability. In system diagnose, model learning with high accuracy is needed for telling the health of the system. A model with high fidelity is also indispensable to virtual development based on computer simulations.

In order to obtain high-fidelity models more efficiently, this chapter discusses data-driven planning and learning approaches for intelligent modeling and identification. Section 5.2 discusses the problem of planning well conditioned trajectories for learning a class of nonlinear models that can be linearized in the feature space. Section 5.3 presents an approach for fast modeling and identification of robot dynamics.

5.2 Fast Planning of Well Conditioned Trajectories for Model Learning

In general, system behavior can be modeled as $y = g(x, Z)$, where $x \in \mathbb{R}^s$ includes the minimum number of system variables that can fully describe the system status (e.g., position, velocity, and acceleration of a point mass), $y \in \mathbb{R}^o$ includes the system variables that are determined or required by x, and $Z \in \mathbb{R}^p$ consists of p model parameters to be learned. Besides the linear models that can be transformed into transfer functions, many important

models are nonlinear and cannot be identified using frequency response. Such models include, but not limited to the imaging model of a camera and the multibody dynamic model of a robot. There are often two major steps in learning those models. First, the model is transformed to the form of

$$y = f(x) Z \tag{5.1}$$

where the system variables and model parameters are decoupled linearly. $f \in \mathbb{R}^{o \times p}$ is called a regressor. Such transformation can be done in a lot of cases. A conventional challenge in this step is deriving the symbolic expression of f. The second step is designing a trajectory of x. The response of the system is measured when it is driven through the trajectory. Then, model learning can be formulated as an optimization problem

$$\min_{Z} \|F(X) Z - Y\| \tag{5.2}$$

with

$$F(X) = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{pmatrix} \tag{5.3}$$

where x_i's and y_i's are the measurements of the system variables. Depending on the sensing noise, unmodeled response, and the way that data are collected, different learning methods can be applied to estimate Z. In general, all methods require the data matrix $F(X)$ to be well conditioned. Otherwise, some parameters might even become not identifiable.

Various approaches have been adopted to plan trajectories that give well conditioned data matrices. In [53], an optimal trajectory is designed for learning camera parameters. The trajectory is derived symbolically by requiring the data matrix to be orthonormal. The method requires nontrivial effort of ad hoc analysis and can be hard to be applied to different problems. In [54], optimal trajectories for learning the multibody dynamic models of robots are generated by solving an optimal control problem, with the cost function as the condition number of the data matrix. Numerical optimization is used to solve the optimal control problem. [55] also deals with generating well conditioned trajectories for learning multibody dynamic models. The trajectory is parameterized by a certain number of via-points in the robot workspace. Numerical optimization is used to adjust the positions and velocities of those points so that the data matrix has a minimized condition number. In [56], the optimal trajectory is parameterized as a finite Fourier series, with the Fourier coefficients optimized numerically. A recent work in [57] follows a similar method with the trajectory parameterized as B-splines. These representative works have drawn much attention over the years. Their success, however, depends heavily on a good initial design of the trajectory because numerical optimization based on gradient estimation often gets trapped at local optima. In regard to these limits, this section presents a different type of method using low-discrepancy sequences and matrix subset selection. The method can be easily applied to various model learning problems.

Learning Error and Condition Number of Data Matrix

The error of model learning comes from three sources - unmodeled response, sensing noise, and rounding error in digital computing. In numerical analysis, the condition number of a matrix is more related to the third source. When studying learning under unmodeled response and sensing noise, however, the influence of rounding error is relatively trivial.

The actual system response with unmodeled response and sensing noise can be described as

$$f\left(\bar{x}-n_x\right)Z+u=\bar{y}-n_y \tag{5.4}$$

where $\bar{x}=x+n_x$ and $\bar{y}=y+n_y$ are the measurements of x and y, n_x and n_y are sensing noises, and u is the unmodeled response. Considering the sensing noise, a maximum-likelihood estimate (MLE) of Z is obtained by maximizing the likelihood function $\prod_{k=1}^{N} p_{\bar{x}}\left(\bar{x}_k\right) p_{\bar{y}}\left(\bar{y}_k\right)$, where $\bar{\bullet}_k$ is the $k^{\mathbf{th}}$ measurement of $\bullet$, $p_a\left(b\right)$ is the probability density function of a. This problem is intractable exactly because the actual values of $x=E\left(\bar{x}\right)$ and $y=E\left(\bar{y}\right)$ are unknown. Some would replace x with the value in the designed trajectory. The corresponding difference between x and $\bar{x}$, however, is not Gaussian anymore because usually the designed trajectory cannot be tracked perfectly. In addition, because f is often nonlinear, the probability distribution of y can be complicated. In regard to this problem, a common practice assumes that the sensing noise n_x is small and can be neglected (at least compared to n_y). This assumption is reasonable with most modern mechatronic systems. Then, the MLE of Z can be obtained in the form of a weighted least squares estimate:

$$\hat{Z}=\left(F^{\mathrm{T}}WF\right)^{-1}F^{\mathrm{T}}WY \tag{5.5}$$

where W is a diagonal weight matrix representing the different noise levels of different components in y. If the noise levels of all components in y are about the same, the optimal estimate as a solution to Eq. (5.2) can be as simple as

$$\hat{Z}=\left(F^{\mathrm{T}}F\right)^{-1}F^{\mathrm{T}}Y \tag{5.6}$$

Because of the unmodeled response, the above solution gives a learning error $\tilde{Z}=\hat{Z}-Z$ with an expectation

$$E\left(\tilde{Z}\right)=\left(F^{\mathrm{T}}F\right)^{-1}F^{\mathrm{T}}\left[E\left(U\right)\right] \tag{5.7}$$

where the matrix U is the stack of the unmodeled response u. In order to reduce the learning error, the smallest eigenvalue of $F^{\mathrm{T}}F$, which corresponds to the smallest singular value of F, should be maximized. This mainly corresponds to minimizing the condition number κ of F defined by l_2-norm. In addition, consider the influence of sensing noise:

$$F\left(Z+\tilde{Z}\right)+U=Y+N_Y \tag{5.8}$$

where N_Y is the perturbation due to sensing noise n_y. The corresponding deviation in the learning result is related to the condition number of the data matrix as

$$\left\| \frac{\tilde{Z}}{Z} \right\| \leqslant \kappa \left\| \frac{N_Y}{Y} \right\| \tag{5.9}$$

Similar conclusion applies when a noise perturbation from n_x is added on F. Besides the batch learning method discussed above, when data are collected continuously and learning is performed in a sequential manner, [54] suggests that both the learning accuracy and the converging rate of the sequential learning algorithm depend on the condition number of the data matrix.

Note that not all methods that reduce the condition number of the data matrix can reduce the learning error. [58] presents a method to reduce the condition number of a matrix by scaling its rows (i.e., multiplying a diagonal matrix). This method is helpful on mitigating the influence of the rounding error in digital computing. Equation (5.7), however, indicates this method will not help reduce the learning error against unmodeled response because it also scales up U.

Planning Well Conditioned Trajectories

Usually, when the trajectory is designed manually by intuition, the data matrix might even be not of full column rank. Try-and-error adjustment takes much effort and does not guarantee improvement. A desired method should depend little on or do not require an initial design of the trajectory. Numerical optimization based on gradient estimation is not preferred because quite often it only leads to local optima with limited improvement. In regard to these considerations, a method different from conventional practices is introduced as follows.

1) Uniform excitation using low-discrepancy sequences

When identifying transfer functions, white noise is a good candidate of excitation because it has a uniform spectrum over all frequencies. Similarly, for the model learning problem described in Eq. (5.2), if the columns of the data matrix can be excited with high dimensional white noise, the eigenvalue spectrum of the model can be fully explored. This ideal preference is not practical for most applications, especially when dynamic models of mechanical systems are learned. An industrial robot simply cannot follow a white noise trajectory. The idea, however, is inspiring. Define the p-dimensional vector space associated with the row vectors in the regressor $f \in \mathbb{R}^{o \times p}$ as the feature space of the model, which is mapped from the state space $\{x | x \in \mathbb{R}^s\}$ by f. It is desirable to have the trajectory explore the feature space thoroughly and uniformly. One way is to make the trajectory go through a certain number of key points that are spread over the feature space thoroughly and uniformly. The resulting data matrix will have full column rank and at least contain a well conditioned subset. Extraction of this subset will be discussed in the next section. Because it is often

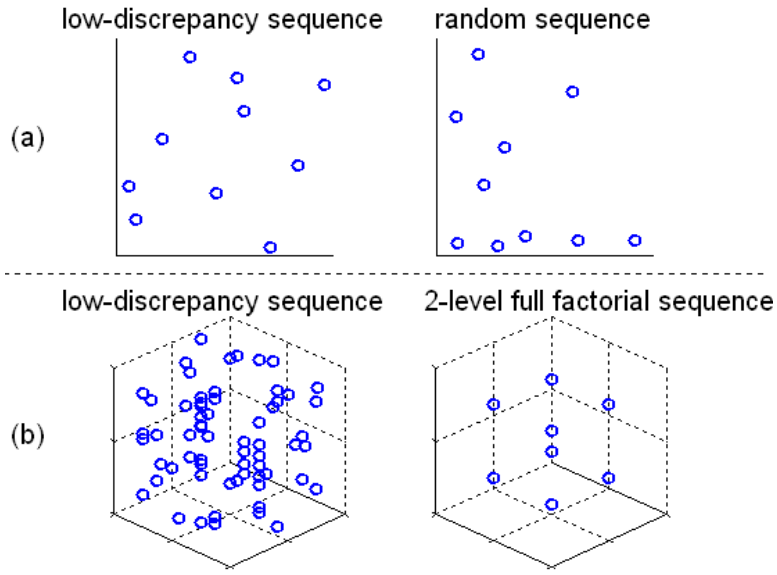


Figure 5.1: Superior uniformity of low-discrepancy sequences - (a) A low-discrepancy sequence compared with a uniformly distributed random sequence - uniformity with limited number of points (10 points). (b) A low-discrepancy sequence compared with a full factorial sequence ($2^6 = 64$ points are distributed in a 6D space and projected to a 3D subspace).

difficult to evaluate the inverse of f, it is easier to spread the points in the state space rather than in the feature space. Due to the nonlinearity of f, the corresponding distribution in the feature space will lose uniformity to some extent. This can be adjusted using the subset selection method as well.

The dimension of the state space is often high. Even for a simple two-axis robot manipulator, the state space is six-dimensional (Section 5.2). In order to spread a limited number (e.g., less than 100) of points uniformly in a high-dimensional space, a low-discrepancy sequence is probably the best choice. Low-discrepancy sequences are deterministic sequences constructed to occupy a hypercube with uniform spacing (i.e., low discrepancy). Because they are often used to replace uniformly distributed random sequences, they are also called pseudorandom sequences. Compared to random sequences, low-discrepancy sequences provide good uniformity even when only a very limited number of points are spread (Fig. 5.1 (a)). This property is helpful in terms that the resulting trajectory does not need to be very long. A low-discrepancy sequence is also more efficient than a full factorial sequence in terms that its projection to a lower-dimensional subspace is also highly uniform. As shown in Fig. 5.1 (b), the 2^6 points of the full factorial sequence overlap at 8 locations when projected to a three-dimensional subspace. Other advantages of a low-discrepancy sequence include the property that a continuous subset from the sequence preserves good uniformity over the whole hypercube.

Among various types of low-discrepancy sequences, the Sobol sequence is used here due to its superior uniformity in high-dimensional applications [59]. The sequence is usually generated from the so called direction numbers, which are in the form of $v_{i,j} = m_{i,j}/2^i$, where $m_{i,j}$ is an odd integer coming from a specific recurrence sequence[60]. The component of the k^{th} point in the Sobol sequence on the j^{th} dimension is given as

$$x_k^{(j)} = b_1 v_{1,j} \oplus b_2 v_{2,j} \oplus b_2 v_{2,j} \oplus \cdots \tag{5.10}$$

where b_i is the i^{th} bit of k's binary code, and $\oplus$ is the bit-by-bit exclusive-or operator. More details of the implementation can be found in [61].

2) Uniformity adjustment via matrix subset selection

As mentioned, it is easier to spread the key points over the state space whose boundaries are often as simple as a hypercube. Because of the nonlinearity of the mapping f, the corresponding distribution in the feature space is distorted and is less uniform. Recovery of the uniformity can be done through matrix subset selection. The problem is often called matrix column subset selection due to its usual formulation, but here it is actually the rows that are selected. The goal is to select a specified number of rows from the data matrix to form a well conditioned subset.

The exact optimal solution of the problem can probably be obtained only through exhaustive search, which is computationally impractical. Results that are good enough, however, can be obtained via certain factorizations of the data matrix [62]. In particular, rank-revealing QR (RRQR) factorizations are helpful. Given a desired subset size k, a RRQR factorization of a matrix $A \in \mathbb{R}^{m\times n}$ has the form of

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ \mathbf{0} & R_{22} \end{pmatrix} \tag{5.11}$$

where Π is a permutation matrix, Q is orthogonal, and $R_{11} \in \mathbb{R}^{k\times k}$ is well conditioned. The columns in A identified by the first k columns of Π form a well conditioned subset. Various algorithms have been developed to compute an RRQR factorization, providing different lower bounds of R_{11}'s smallest singular value, with a general form of

$$\sigma_k(R_{11}) \geqslant \frac{\sigma_k(A)}{l(k,n)} \tag{5.12}$$

where $l(k,n)$ is bounded by a polynomial of k and n [63].

After recovering uniformity by subset selection, a smooth trajectory is designed to connect the selected points. Besides the selected key points, the additional points along the trajectory might not have maximum uniformity in the feature space and the corresponding data matrix could have a slightly worse condition number. It is always doable to use only the data recorded at the key points for model learning. More data, however, is desirable to mitigate the influence of sensing noise. If the condition number deteriorates too much, subset selection can

be performed again. An RRQR factorization, however, will not be efficient as the dimension of the data matrix can be very high. Instead, a randomized algorithm that consists of two steps is preferred. First, a raw subset is selected based on a particular random sequence with appropriate cardinality. Then, factorization is performed over the raw subset to remove the redundant points that reduce the uniformity. The algorithm gives improvement with a certain rate of success [64].

3) Trajectory fitting and scaling

When learning the dynamic model of a mechanical system, a smooth trajectory is required. The trajectory shall contain all the key points determined in the previous sections and is physically feasible for the system to follow. Take learning the multibody dynamic model of a robot as an example. Each point in the state space contains the rotation angles, angular velocities, and angular accelerations of all joints. Consecutive key points are connected with splines to form a smooth trajectory. Specifically, the rotation of the i^{th} joint between the k^{th} and $(k+1)^{\text{th}}$ key points is described as

$$q_i(t) = \Phi(t)\, a_{i,k} \quad t \in [0, T_k] \tag{5.13}$$

where $\Phi(t) = (1\ t \cdots t^5)$ and $a_{i,k} = \left(a_{i,k}^{(0)}\ a_{i,k}^{(1)} \cdots a_{i,k}^{(5)}\right)^{\mathrm{T}}$. First, $a_{i,k}$ is determined using the continuity conditions with T_k set to 1 second:

$$\begin{pmatrix} \Phi(0) \\ \Phi'(0) \\ \Phi''(0) \\ \Phi(1) \\ \Phi'(1) \\ \Phi''(1) \end{pmatrix} a_{i,k} = \begin{pmatrix} q_{i,k} \\ \dot{q}_{i,k} \\ \ddot{q}_{i,k} \\ q_{i,k+1} \\ \dot{q}_{i,k+1} \\ \ddot{q}_{i,k+1} \end{pmatrix} \tag{5.14}$$

Then, the maximum velocity and acceleration within the time interval $[0, T_k]$ are examined. If they are beyond the nominal limits of the robot, T_k is scaled up by a factor of $r > 1$ so that velocity and acceleration will be scaled by $1/r$ and $1/r^2$ respectively. If the total time duration of the trajectory matters, speeding up can be done similarly by scaling T_k by $r < 1$ when the maximum velocity and acceleration are below the limits. Sorting the key points in advance according to position or velocity can also help.

Applications

1) Dynamic model identification of a two-axis robot manipulator

The advantages of the proposed method is more appreciable when dealing with problems with a high-dimensional feature space. This example, however, has a relatively simple regressor

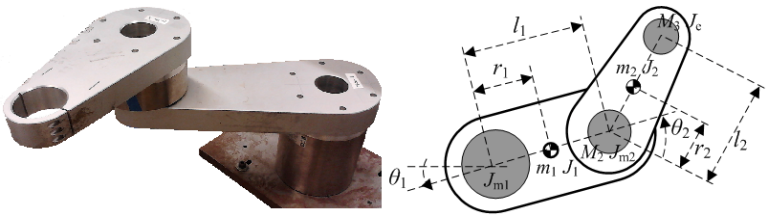


Figure 5.2: A two-axis robot manipulator

and a low-dimensional feature space, and is presented for the purpose of giving an intuitive illustration of the proposed method. Because of the low-dimensionality of the feature space, the distribution uniformity of the key points in the feature space can be easily examined by visual inspection.

Consider the two-axis planar robot manipulator shown in Fig. 5.2. The regressor of its multibody dynamic model [65] in the form of Eq. (5.1) can be expressed as

$$\begin{pmatrix} \ddot{\theta}_1 & \left(2\ddot{\theta}_1+\ddot{\theta}_2\right)\cos\theta_2-\dot{\theta}_2\left(2\dot{\theta}_1+\dot{\theta}_2\right)\sin\theta_2 & \ddot{\theta}_2 \\ 0 & \ddot{\theta}_1\cos\theta_2+\dot{\theta}_1^2\sin\theta_2 & \ddot{\theta}_1+\ddot{\theta}_2 \end{pmatrix} \tag{5.15}$$

where θ_1 and θ_2 are the rotation angles of the two joints respectively. There are three inertia parameters to be learned, corresponding to a three-dimensional feature space. The rotation angles, angular velocities, and angular accelerations of the two joints span a six-dimensional state space. First, a Sobol sequence with 100 points is distributed over a hypercube in the state space bounded by the nominal upper and lower limits of the variables. Each point in the sequence contains six components and is represented by a line in Fig. 5.3 (a). The corresponding distribution of end-effector positions in the robot workspace is examined to exclude the points violating spatial boundaries (Fig. 5.3 (b)). As shown in Fig. 5.3 (c), because of the nonlinear mapping from the state space to the feature space, the distribution of the feasible points in the feature space is not quite uniform. Matrix subset selection is performed to pick ten points (the red points in Fig. 5.3 (c)) out of all the spatially feasible points to form a distribution with good uniformity. The condition number of the data matrix is reduced from 4.34 (all feasible points) to 3.40 (selected points). Finally, spline fitting and scaling are performed to connect the selected key points, giving the trajectory shown in Fig. 5.3 (d). The condition number of the data matrix corresponding to the whole trajectory is 3.52. Note that for a feature space with a dimension as low as three, a data matrix of small condition number can be easily obtained. The purpose of this example is mainly on giving an intuitive illustration, whereas the advantages of the proposed method will be appreciated more in the following two examples with high-dimensional feature spaces.

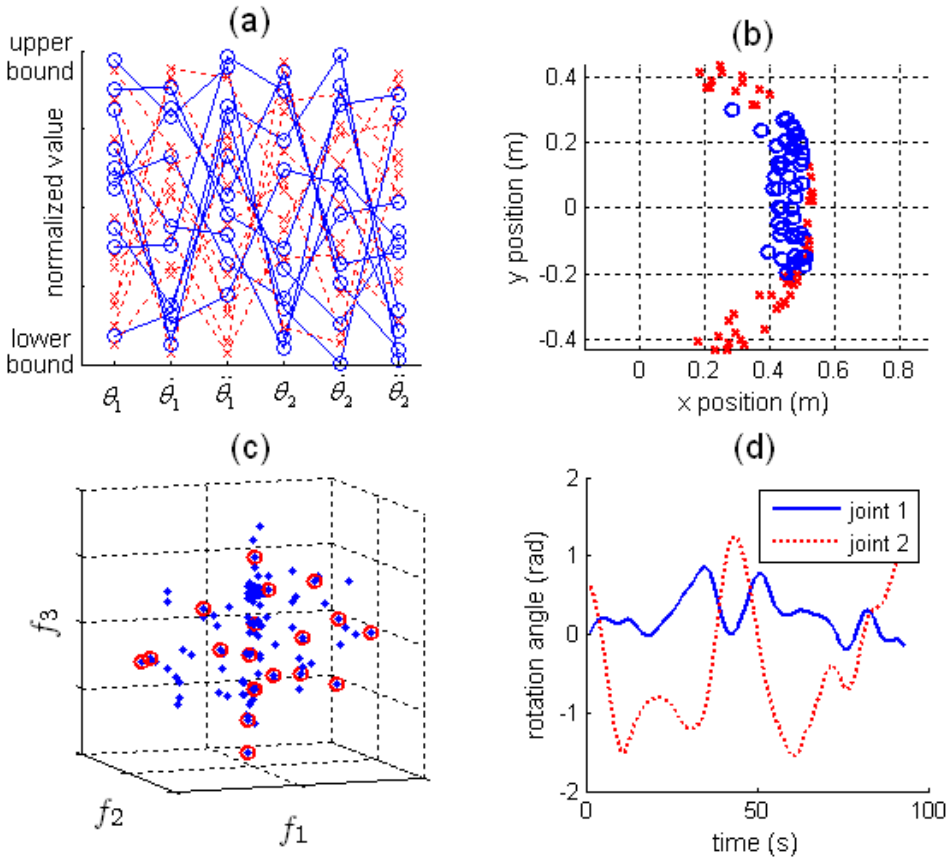


Figure 5.3: Planning a well conditioned trajectory for a two-axis robot manipulator - (a) A Sobol sequence in the state space. Only the first 20 points are drawn. Each line represents a point in the 6D state space. Red lines correspond to the red points in (b). (b) Distribution of the end-effector positions corresponding to the Sobol sequence in the state space. Red points are excluded by spatial constraints. (c) Distribution of the key points in the feature space. Red points are selected as a subset with better uniformity. (d) The final trajectory.

2) camera Calibration

Accurate camera calibration is indispensable for vision guided robot systems. The goal is to identify both the extrinsic parameters (i.e., camera location and orientation with respect to the robot) and intrinsic parameters (i.e., focal length, principal point, etc.). In an eye-to-hand setup, the robot carries a sample target such as a cross-shaped marker to different locations. Each location's coordinates $M = (x\ y\ z)$ in the robot workspace and its identified position $m = (h\ v)$ in the image are recorded and used to learn the camera model. In this application, the state space is simply the three-dimensional Cartesian space. The nonlinear imaging model of a camera can be transformed to the form of Eq. (5.1). Consider the widely

adopted model formulation in [66]:

$$\begin{pmatrix} M & 1 & \mathbf{0} & 0 & -hM & -h \\ \mathbf{0} & 0 & M & 1 & -vM & -v \end{pmatrix} \widehat{Z} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{5.16}$$

where the parameter vector $\widehat{Z}$ has twelve components. As explained in [66], the maximum rank of the data matrix is 11, which enables $\widehat{Z}$ to be determined up to a scaling factor. An additional constraint $||\widehat{Z}(9:11)|| = 1$ is used to obtain a unique result. Modify the formulation as below so it fits better with the framework introduced earlier:

$$\begin{pmatrix} M & 1 & \mathbf{0} & 0 & -hM \\ \mathbf{0} & 0 & M & 1 & -vM \end{pmatrix} Z = \begin{pmatrix} h \\ v \end{pmatrix} \tag{5.17}$$

where the modified parameter vector $Z = \widehat{Z}(1:11)/\widehat{Z}(12)$ has eleven components and can be determined uniquely when the data matrix has full column rank. The feature space is eleven-dimensional. At least six points are needed.

A widely used industrial practice is sampling the target at the vertices of an octahedron (Fig. 5.4 (a)). This method sometimes gives ill-conditioned data though it seems to explore the field-of-view thoroughly. The layout produced by the proposed method gives a much improved condition number (Fig. 5.4 (d)). For comparison purpose, the octahedron layout is adjusted using numerical optimization (a trust-region-reflective algorithm, Fig. 5.4 (b)), which gives a local optima not very different from the initial layout with limited improvement. Because it is not a dynamic model to be learned and the robot can stop at the sampling locations, a smooth trajectory is not required to connect all sampling locations. When the key points are mapped from the state space to the feature space by the model in Eq. (5.17), a rough learning result (e.g., obtained using the octahedron layout) is needed to estimate h and v in the regressor.

3) Dynamic model identification of a wafer handling robot with very limited workspace

The proposed method is applied to generate a trajectory to identify the multibody dynamic model of an eight-link two-DOF parallel robot. The robot is used inside the vacuum environment of an IC fabrication tool to transfer silicon wafers among the processing chambers and the loading dock. Description of the robot and derivation of a symbolic expression of its dynamic model can be found in [67]. There are seventeen inertia parameters to be learned, and the regressor has a size of 2×17, corresponding to a seventeen-dimensional feature space. Despite the high-dimensional feature space to be explored, the robot is allowed to move within a very limited area inside the tool (Fig. 5.5). Manually planning a well conditioned trajectory is difficult unless the robot is removed from the tool and tested without spatial constraints. In the case of after-installation system identification and health monitoring, removing the robot from and restoring it to an installed tool is extremely time-consuming. In

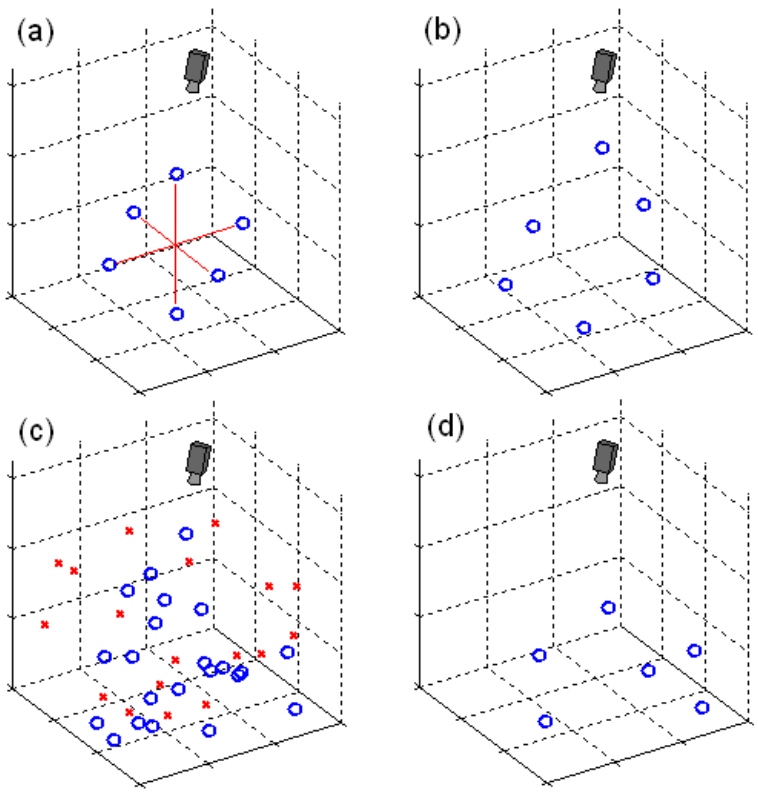


Figure 5.4: Layout of sampling points for camera calibration - (a) A conventional octahedron layout ($\kappa = 1.4 \times 10^7$). (b) Adjusted octahedron layout ($\kappa = 7.4 \times 10^6$). (c) A Sobol sequence layout. Red points are outside the field-of-view or interfere with workspace environment. ($\kappa = 6.6 \times 10^5$) (d) Final layout after subset selection ($\kappa = 5.6 \times 10^5$).

regard to this challenge, the proposed method produces a trajectory that efficiently explores the feature space within the constrained workspace.

The eight-link parallel robot is actuated by two motors. The rotation angles, angular velocities, and angular accelerations of the motors span a six-dimensional state space. First, a Sobol sequence with 100 points is distributed over a hypercube in the state space bounded by the nominal upper and lower limits of the variables. The corresponding positions of the end-effector tip in the robot workspace are examined. The points that fall out of the allowed area or sit too close to the walls are dropped (Fig. 5.6 (a)). A minimum number of nine points are chosen through subset selection as the key points. The final trajectory as shown in Fig. 5.7 (a) gives a data matrix with a condition number of 7.2×10^3.

The trajectory is tested for model learning using a robot simulator, in which true values of the model parameters are known. Simulated sensing noise is added to the measurements of the motion variables and the torques. A baseline trajectory constructed based on rotation and extension/retraction motions is also tested for comparison (Fig. 5.7 (b)). Such a tra-

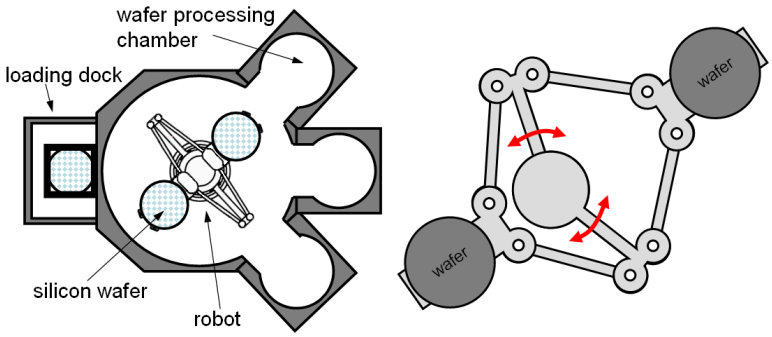


Figure 5.5: A wafer handling robot inside an IC fabrication tool

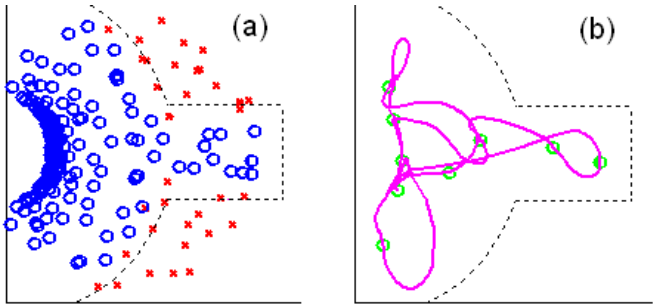


Figure 5.6: Trajectory planning for a wafer handling robot - (a) Distribution of the end-effector tip positions corresponding to a Sobol sequence in the state space. Red points are excluded by spatial constraints (dashed lines). (b) Final trajectory in the robot workspace after subset selection, sorting, spline fitting, and scaling.

jectory is widely used by industrial users for model learning of a wafer handling robot inside a tool. The robot is controlled by a cascade PID controller. Because the dynamic model is yet to be learned, no torque feedforward control is available, resulting in large tracking errors (Fig. 5.7). Due to these factors, the condition number of the data matrix constructed from actual system response is worse than the designed value to some extent - 1.1×10^4 when the proposed trajectory is used, and 2.1×10^6 when the baseline trajectory is used. A comparison of the learning errors is shown in Fig. 5.8. The advantage of the proposed method is obvious.

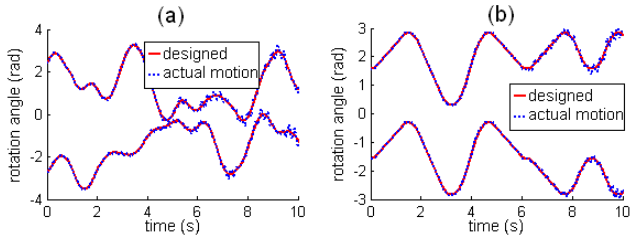


Figure 5.7: Trajectories for model learning tests - (a) Planned by the proposed method. (b) A baseline trajectory.

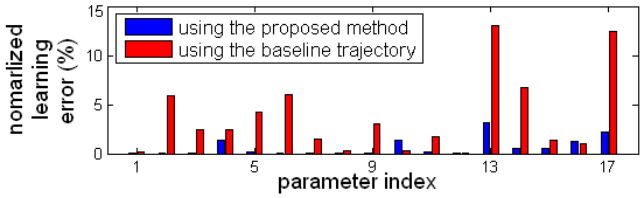


Figure 5.8: Comparison of learning errors

5.3 Fast Modeling and Identification of Robot Dynamics using the Lasso

A dynamic model of a robot manipulator is a function in the form of $\boldsymbol{\tau} = g(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, Z)$, where τ is the vector of torques applied by the motors, $\mathbf{q}$ is the vector of rotation angles of the motors, and Z consists of the mechanical parameters.

The conventional method to model and identify robot dynamics involves the following steps [68, 69]:

1) Apply analytical mechanics, derive the dynamic model in the form of Lagrange's equation of motion:

$$M\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + f = \tau \tag{5.18}$$

where $M = M(\mathbf{q}, Z)$ is the mass matrix, $C = C(\mathbf{q}, \dot{\mathbf{q}}, Z)$ represents the centrifugal and Coriolis effect, and f is the friction. Gravity is not considered as the method is applied to planar robots that operate horizontally.

2) Analytically transform Eq. (5.18) to the form of Eq. (5.1)

$$YZ = \tau \tag{5.19}$$

where $Y = Y(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is a matrix determined solely by robot motion. It is called a *regressor* or *regression matrix*. Except for certain types of robots that have closed-chain mechanisms

[67], the linear decoupling in the form of Eq. (5.19) can always be done. Instead of including every individual mechanical parameter of the robot, such as the length, mass, and rotational inertia of each link, the components in Z can be functions of those parameters. Thus, Z might have a dimension much lower than the total number of individual mechanical parameters.

3) Drive the actual robot through a trajectory that gives a variety of motion patterns. Record the signals of $\mathbf{q}$ and τ from sensors. Compute $Y(t_k)$ at selected time t_k's. Stack data to form the data matrices

$$\mathbf{Y} = \begin{pmatrix} Y(t_1) \\ Y(t_2) \\ Y(t_3) \\ \vdots \end{pmatrix} , \mathbf{T} = \begin{pmatrix} \tau(t_1) \\ \tau(t_2) \\ \tau(t_3) \\ \vdots \end{pmatrix} \tag{5.20}$$

4) As described in Eq. (5.2) in the previous section, solve a least squares regression problem to determine Z:

$$\min_Z \|\mathbf{Y}Z - \mathbf{T}\|_2 \tag{5.21}$$

A major difficulty of going through this procedure comes from the analytical derivation in step 1) and 2). Quite often, the complexity of the equations is challenging for by-hand derivation. Usually symbolic computation using commercial software is necessary. Inputting the problem to the software, however, still requires much effort and attention not to introduce any typo. In addition, even with symbolic computation, the operating person is required to have a comprehensive knowledge on multi-body dynamics.

One approach that may simplify the modeling and identification procedure considerably is to skip the analytical derivation and obtain an expression of Y using a machine learning method. With some basic information of the robot kinematics, one is able to build the candidate terms that form the components in Y. Further using a *regressor selection* technique, the actual components of Y can be identified from the candidates. The next section explains the proposed method with a simple example of a two-axis direct-drive robot. Then the method is applied to a three-link belt-driven robot for further demonstration. Discussion and conclusion are given at the end.

A Simple Example

Consider the two-axis robot manipulator shown in Fig. 5.2. Two direct-drive motors actuate the two joints independently, i.e., motor rotation $\mathbf{q}$ equals the joint rotation θ. Applying analytical mechanics, a dynamic model in the form of Eq. (5.18) can be obtained, with

$$M = \begin{pmatrix} \alpha + 2\beta\cos q_2 & \delta + \beta\cos q_2 \\ \delta + \beta\cos q_2 & \delta \end{pmatrix} \tag{5.22}$$

$$C = \beta\sin q_2 \begin{pmatrix} -2\dot{q}_2 & -\dot{q}_2 \\ \dot{q}_1 & 0 \end{pmatrix} \tag{5.23}$$

where α, β, δ are the (condensed) parameters to be identified [70]. They are functions of the mechanical parameters specified in Fig. 5.2. Friction is not included in the model. It is considered as unmodeled disturbance in the experiment data, and challenges the robustness of the approach[1].

The model can be transformed into the form of Eq. (5.19), with the regression matrix Y as

$$\begin{pmatrix} \ddot{q}_1 & (2\ddot{q}_1 + \ddot{q}_2)\cos q_2 - \dot{q}_2(2\dot{q}_1 + \dot{q}_2)\sin q_2 & \ddot{q}_2 \\ 0 & \ddot{q}_1 \cos q_2 + \dot{q}_1^2 \sin q_2 & \ddot{q}_1 + \ddot{q}_2 \end{pmatrix} \tag{5.24}$$

and the parameter vector as $Z = (\alpha, \beta, \delta)^T$.

An alternative method to obtain the expression of Y without going through analytical derivation is by using a machine learning approach. Considering the direct-drive configuration of the robot, the components in Y should be linear combinations of some candidate terms. Each of these terms is in the form of AB, where A is $\ddot{q}_i$ or $\dot{q}_i\dot{q}_j$, and B is 1, $\sin q_i$, or $\cos q_i$, with $i, j = 1, 2$. In total, there are twenty-five[2] candidates. The actual components of Y are selected from those candidates using a *regressor selection* method - the l_1-norm penalized least squares regression (a.k.a, the *Lasso* [71, 72]). The problem is formulated as

$$\min_{\tilde{Z}_i \in \mathbb{R}^{25}} \left\| \tilde{\mathbf{Y}} \tilde{Z}_i - \mathbf{T}_i \right\|_2^2 + \lambda \left\| \tilde{Z}_i \right\|_1 \quad (i = 1, 2) \tag{5.25}$$

where

$$\tilde{\mathbf{Y}} = \begin{pmatrix} \tilde{Y}(t_1) \\ \tilde{Y}(t_2) \\ \tilde{Y}(t_3) \\ \vdots \end{pmatrix}, \ \mathbf{T}_i = \begin{pmatrix} \tau_i(t_1) \\ \tau_i(t_2) \\ \tau_i(t_3) \\ \vdots \end{pmatrix} \tag{5.26}$$

$\lambda > 0$ is an adjustable penalty weight. In $\tilde{\mathbf{Y}}$,

$$\tilde{Y} = \begin{pmatrix} \ddot{q}_1, & \ddot{q}_2, & \dot{q}_1\dot{q}_2, & \cdots \end{pmatrix}_{1 \times 25} \tag{5.27}$$

includes all the candidate terms. Note the models for two motors are identified separately. Instead of a single 3×1 vector Z, two 25×1 vectors $\tilde{Z}_1$ and $\tilde{Z}_2$ will be obtained. Both of them have many (near) zero components, corresponding to the unselected candidate terms. The sparsity is encouraged by the l_1-norm penalty. In order to select the correct terms and obtain good regression accuracy, the penalty weight λ should be tuned properly. The simplest tuning method is cross-validation, whereas advanced intelligent methods provide additional benefits. [73] gives a good review on related techniques.

[1] An additional part for friction can be included in the model and transformed into the form of Eq.(5.19) for least squares regression. The regression result of the friction coefficients, however, is highly vulnerable to sensor noise. This is especially true for the data with velocity crossing zero.

[2] In fact, considering that the motion of the robot is centrosymmetric with respect to joint 1, q_1 should not appear in the model, and can actually be excluded from B. This can reduce the total number of candidates to fifteen.

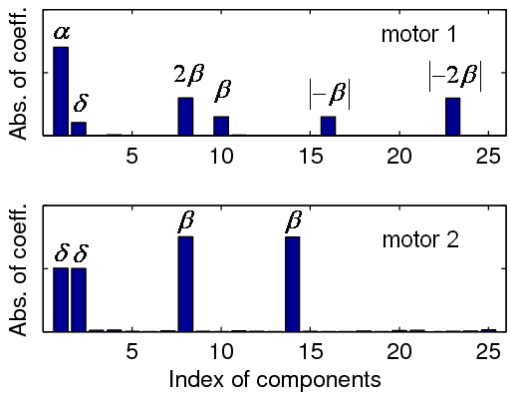


Figure 5.9: Regression selection for a two-axis direct-drive robot

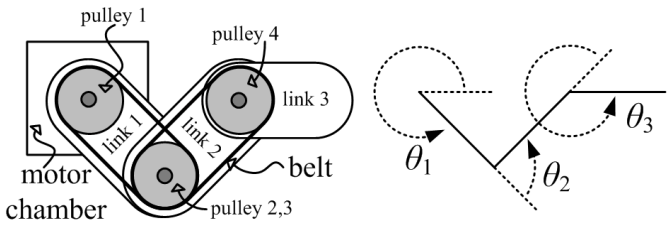


Figure 5.10: A three-link two-DOF belt-driven robot (Actual picture of the robot is confidential.)

Apply Eq.(5.25) to experiment data. Plot the absolute values of the components in the optimal $\tilde{Z}_i$'s on bar charts (Fig.5.9). The effect of selection is obvious. The six selected terms for the model of motor 1 are $\ddot{q}_1$, $\ddot{q}_2$, $\ddot{q}_1 \cos q_2$, $\ddot{q}_2 \cos q_2$, $\dot{q}_2^2 \sin q_2$, and $\dot{q}_1 \dot{q}_2 \sin q_2$, which are consistent with the first row of Y in Eq. (5.24). The dynamic model for motor 1 is

$$\hat{Y}_1 \hat{Z}_1 = \tau_1 \tag{5.28}$$

where the length of $\hat{Y}_1$ and $\hat{Z}_1$ is 6. $\hat{Y}_1$ and $\hat{Z}_1$ shrink from $\tilde{Y}$ and the optimal $\tilde{Z}_1$ respectively after eliminating the unselected components. Specifically, $\hat{Z}_1 = (\alpha,\ \delta,\ 2\beta,\ \beta,\ -\beta,\ -2\beta)^T$. Situation is similar for motor 2, where the length of $\hat{Y}_2$ and $\hat{Z}_2$ is 4. Note the learning algorithm is not aware that only three parameters (α, β, and δ) in $\hat{Z}_i$'s are independent. This issue is discussed later.

Application to a Belt-Driven Robot

Figure 5.10 shows a three-link two-DOF belt-driven robot. It is widely used in semiconductor industry for wafer handling [74, 75]. Two motors sit coaxially beneath joint 1. Motor 1

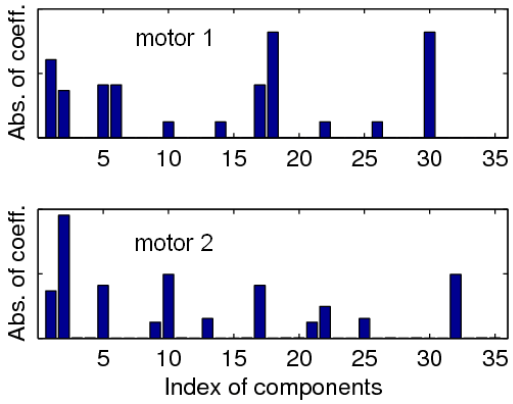


Figure 5.11: Regressor selection for a three-link two-DOF belt-driven robot

drives link 1 directly. Motor 2 drives a belt mechanism via pulley 1 (not connected with link 1). When the two motors rotate synchronously, the robot arm rotates without radial motion. When motor 1 rotates while motor 2 being held, the robot arm extends or retracts radially. Due to the belt mechanism, link 3 stays along the radial direction at all time. The relationship between the joint angles θ_i and motor rotation q_i is

$$\begin{aligned} \theta_1 &= q_1 \\ \theta_2 &= 2q_2 - 2q_1 \\ \theta_3 &= q_1 - q_2 \end{aligned} \tag{5.29}$$

The candidate terms for the regression matrix are again in the form of AB, where A is $\ddot{q}_i$ or $\dot{q}_i\dot{q}_j$ with $i, j = 1, 2$, and B is 1, $\sin(\bullet)$ or $\cos(\bullet)$ with $\bullet$ being θ_2, θ_3, or $\theta_2 + \theta_3$ expressed as in Eq.(5.29). Because the motion of the robot is centrosymmetric with respect to joint 1, θ_1 is not considered for B. Apply the Lasso to the experiment data, eleven components are selected from thirty-five candidates for each motor (Fig.5.11). The result agrees with an analytical derivation in [76].

5.4 Chapter Summary

This chapter discussed planning and learning algorithms for intelligent system modeling and identification. Section 5.2 presented a method of planning well conditioned trajectories for the purpose of model learning [77]. Low-discrepancy sequences are used to provide thorough and uniform exploration of the state space. The uniformity of the corresponding excitation in the feature space is obtained through matrix subset selection. A spline fitting and scaling procedure ensures the smoothness and physical feasibility of the final trajectory. The resulting data matrix has a good condition number. Compared to conventional methods that use analytical or numerical optimization, the proposed method can be easily applied to various

problems with little ad hoc formulation and does not require an initial design. The proposed approach was applied to camera calibration and robot dynamic model identification. Section 5.3 presented an approach for fast modeling and identification of robot dynamics is presented [78]. By using a regressor selection technique of machine learning, analytical dynamic models can be obtained without going through the derivation of analytical mechanics. The obtained model is useful for many control strategies.

Chapter 6

Conclusions and Future Work

This dissertation presented a series of data-driven approaches to improve the sensing and control of robot manipulators from several aspects, including sensor fusion for motion sensing, statistical learning for compensating slow feedback, nonparametric learning control, as well as intelligent modeling and identification.

Chapter 2 presented a method of using auxiliary sensors and a sensor fusion algorithm to directly measure the motion of a robot end-effector. The non-contact direct measurement of end-effector position and good estimation of its velocity are highly desirable to overcome the limitations of conventional indirect drive servo system of industrial robots. Chapter 2 first proposed the development of a PSD camera using position sensitive detector. While being cost-effective, the PSD camera provides much faster response than typical vision systems. With proper work on noise mitigation and calibration, the PSD camera has demonstrated promising precision and accuracy. These advantages make it favorable to be utilized in real-time feedback systems. Furthermore, to acquire good velocity estimation, a kinematic Kalman filter based on a three-dimensional rigid body motion model was applied to perform sensor fusion, where the measurements from inertial sensors mounted on the robot end-effector were fused with that from the PSD camera. Performance tests of the PSD camera and validation experiments of the sensor fusion scheme were conducted on a FANUC M-16iB robot. Promising results was demonstrated on both position sensing and velocity estimation of the robot end-effector. Future work can be conducted on increasing the measurement dimensions of the PSD camera system by using more markers and/or more than one camera. In particular, with more markers, advanced multiplexing technique may be required. In addition, self-calibration techniques of inertial sensors to incorporate with the KKF scheme may be a key aspect if the velocity estimation is to be further improved.

Chapter 3 discussed a statistical learning algorithm to compensate slow feedback for real-time motion control. The approach is intended to help industrial robots utilize visual feedback in real-time motion control. Due to limit of hardware cost, common vision systems equipped on industrial robots have large latency and low sampling rate, i.e., visual sensing dynamics. In order to utilize the slow visual feedback to guide real-time motion control, Section 3.2 presented a visual sensing dynamics compensation (VSDC) algorithm.

The VSDC algorithm compensates sensing latency by using stochastic models to predict the current move of the target from the delayed measurements. The low sampling rate is compensated by interpolation using the model. Statistical methods have been developed for adaptive model learning. Tests were conducted to provide evaluation from different aspects, including estimation accuracy, bandwidth, and the effect in closed-loop visual servoing.

Using the compensation algorithm, Section 3.3 discussed the control strategies for robot manipulators to track moving targets based on the compensated vision guidance. In order to implement real-time vision guidance on industrial robot manipulators, both the sensing limits of typical industrial machine vision systems and the dynamics characteristics of industrial robots need to be fully considered. A cascade control structure was proposed. It includes a kinematic visual servoing (KVS) module and a robot dynamics compensation module (RDC). A basic Jacobian-based KVS control law was formulated using sliding control. Then, the speed and torque limits of actuators were addressed using a constrained optimal control approach. The proposed schemes were tested experimentally. A major limit of the introduced method is that the field-of-view (FOV) limit of the vision sensor is not included in the constraints. Due the rotation of the wrist, a 6-DOF robot manipulator loses view of the target easily when approaching it. Unexpected move might happen in such a situation. However, adding the FOV limit to the constraints often makes the optimal control problem infeasible. Future exploration can focus on dealing with the FOV limit as well as the preferred relative approaching direction.

Chapter 4 presented a nonparametric learning control method and its application on precision tracking control of industrial robot manipulators. The method features data-driven iterative compensation that benefits from the repetitive nature of industrial robots' motion. A two-part compensation structure was proposed, in which motor side tracking and transmission error are handled by separate learning modules. Depending on the specific setup of end-effector sensing, either timed trajectory measurement or untimed two-dimensional contour inspection can be used for the compensation. Nonparametric statistical learning was applied using Gaussian process regression. Considerations on incorporating analytical models and selecting data subsets for more efficient learning were discussed. Experimental results showed that the proposed method for torque compensation can significantly improve motor side tracking performance. Meanwhile, the motor reference compensation scheme can handle transmission error effectively. For a benchmark hole cutting trajectory with a one-inch diameter and one-second duration, the tracking error can be reduced from above two millimeters to sub-millimeter levels within a few iterations.

Chapter 5 discussed planning and learning algorithms for intelligent system modeling and identification. Section 5.2 presented a method of planning well conditioned trajectories for the purpose of model learning. Low-discrepancy sequences are used to provide thorough and uniform exploration of the state space. The uniformity of the corresponding excitation in the feature space is obtained through matrix subset selection. A spline fitting and scaling procedure ensures the smoothness and physical feasibility of the final trajectory. The resulting data matrix has a good condition number. Compared to conventional methods that use analytical or numerical optimization, the proposed method can be easily applied to various

problems with little ad hoc formulation and does not require an initial design. Examples of applications were given using camera calibration and robot dynamic model identification. In Section 5.3, an approach for fast modeling and identification of robot dynamics was presented. By using a regressor selection technique of machine learning, analytical expression of dynamic models can be obtained without going through the derivation of analytical mechanics. The obtained models are useful for many control strategies. A major limit of the proposed modeling method is that as the degrees of freedom of the robot increase, building the candidate terms in the regression matrix becomes more difficult. This is especially true when gravity is involved. Future study may focus on developing a multi-level feature selection strategy to obtain a more favorable computation complexity.

Bibliography

[1] C. Wang, W. Chen, and M. Tomizuka. "Robot end-effector sensing with position sensitive detector and inertial sensors". In: *International Conference on Robotics and Automation*. May 2012, pp. 5252–5257.

[2] S. Jeon, M. Tomizuka, and T. Katou. "Kinematic Kalman filter (KKF) for robot end-effector sensing". In: *Journal of Dynamic Systems, Measurement, and Control* 131.2 (2009), pp. 21010–21018.

[3] G. Brooker. *Introduction to Sensors for Ranging and Imaging*. Scitech, 2009.

[4] K. Peiponen, R. Myllyla, and A.V. Priezzhev. *Optical Measurement Techniques*. Springer, 2009.

[5] F. Blais. "Review of 20 Years of Range Sensor Development". In: *Journal of Electronic Imaging* 13.1 (Jan. 2004), pp. 231–240.

[6] B. Shirinzadeh et al. "Laser interferometry-based guidance methodology for high precision positioning of mechanisms and robots". In: *Robotics and Computer-Integrated Manufacturing* 26 (1 Feb. 2010), pp. 74–82.

[7] H. C. Shim, M. Kochem, and M. Tomizuka. "Use of Accelerometer for Precision Motion Control of Linear Motor Driven Positioning System". In: *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 4. Aachen, Germany, 1998, pp. 2409–2414.

[8] D.J. Lee and M. Tomizuka. "State/ Parameter/ Disturbance Estimation With an Accelerometer in Precision Motion Control of a Linear Motor". In: *Proceedings of 2001 ASME International Mechanical Engineering Congress and Exposition*. Nov. 2001, pp. 11–16.

[9] S. Jeon, M. Tomizuka, and T. Katou. "A New Kinematic Kalman Filter (KKF) for End-Effector Sensing of Robotic Manipulators". In: *Proceedings of 2007 ASME International Mechanical Engineering Congress and Exposition*. 2007, pp. 957–962.

[10] B. Cyganek and J.P. Siebert. *An Introduction to 3D Computer Vision Techniques and Algorithms*. WILEY, 2009.

[11] J. Pers and S. Kovacic. "Nonparametric, Model-Based Radial Lens Distortion Correction Using Tilted Camera Assumption". In: *Proceedings of the Computer Vision Winter Workshop*. Citeseer, 2002, pp. 286–295.

[12] I. Amidror. "Scattered data interpolation methods for electronic imaging systems: a survey". In: *Journal of electronic imaging* 11.2 (2002), pp. 157–176.

[13] P.I. Corke. "Dynamic Issues in Robot Visual-Servo Systems". In: *Proceedings of 1995 International Symposium on Robotics Research*. Springer, 1995, pp. 488–498.

[14] P.I. Corke and M.C. Good. "Dynamic effects in visual closed-loop systems". In: *IEEE Transactions on Robotics and Automation* 12.5 (Oct. 1996), pp. 671–683.

[15] C. Wang, C.-Y. Lin, and M. Tomizuka. "Statistical Learning Algorithms to Compensate Slow Visual Feedback for Industrial Robots". In: *Journal of Dynamic Systems, Measurement, and Control* 137 (3 2014), pp. 031001–1.

[16] A. Namiki and M. Ishikawa. "Robotic catching using a direct mapping from visual information to motor command". In: *Proceedings of 2003 IEEE International Conference on Robotics and Automation*. Vol. 2. Sept. 2003, pp. 2400–2405.

[17] N. Furukawa et al. "Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system". In: *Proceedings of 2006 IEEE International Conference on Robotics and Automation*. May 2006, pp. 181–187.

[18] X. Rong Li and V.P. Jilkov. "Survey of maneuvering target tracking, Part I - Dynamic models". In: *IEEE Transactions on Aerospace and Electronic Systems* 39.4 (Oct. 2003), pp. 1333–1364.

[19] R.A. Singer and K.W. Behnke. "Real-Time Tracking Filter Evaluation and Selection for Tactical Applications". In: *IEEE Transactions on Aerospace and Electronic Systems* 7.1 (Jan. 1971), pp. 100–110.

[20] S. Jeon. "State estimation based on kinematic models considering characteristics of sensors". In: *Proceedings of 2010 American Control Conference*. July 2010, pp. 640–645.

[21] C. Wang, C.-Y. Lin, and M. Tomizuka. "Visual Servoing Considering Sensing Dynamics and Robot Dynamics". In: *Proceedings of the 6th IFAC Symposium on Mechatronic Systems*. Apr. 2013, pp. 45–52.

[22] D. Swaroop et al. "Dynamic surface control for a class of nonlinear systems". In: *IEEE Transactions on Automatic Control* 45 (10 2000), pp. 1893–1899.

[23] J.-J. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Englewood Cliffs, 2004.

[24] M. Keshmiri, M. Keshmiri, and A. Mohebbi. "Augmented online point to point trajectory planning, a new approach in catching a moving object by a manipulator". In: *Proceedings of 2010 IEEE International Conference on Control and Automation*. June 2010, pp. 1349–1354.

[25] T. Kroger and J. Padial. "Simple and robust visual servo control of robot arms using an on-line trajectory generator". In: *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*. May 2012, pp. 4862–4869.

[26] T. Kroger and F.M. Wahl. "Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events". In: *IEEE Transactions on Robotics* 26.1 (Feb. 2010), pp. 94–111.

[27] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*. Springer-Verlag, Berlin, 2003.

[28] P. Reynoso-Mora, W. Chen, and M. Tomizuka. "On the time-optimal trajectory planning and control of robotic manipulators along predefined paths". In: *American Control Conference (ACC), 2013*. 2013, pp. 371–377.

[29] W. Chen and M. Tomizuka. "Direct Joint Space State Estimation in Robots With Multiple Elastic Joints". In: *IEEE/ASME Transactions on Mechatronics* PP.99 (2013), pp. 1–10.

[30] L. Sciavicco and L. Villani. *Robotics: modelling, planning and control*. Springer, 2009.

[31] W. Wang and C. Liu. "Controller design and implementation for industrial robots with flexible joints". In: *IEEE Transactions on Industrial Electronics* 39.5 (Oct. 1992), pp. 379–391.

[32] M. Becquet. "Analysis of flexibility sources in robot structure". In: *Proceeding of IMACS/IFAC Symppsium - Modeling and Simulation of Distributed Parameters, Hiroshima, Japan*. 1987, pp. 419–424.

[33] S. Arimoto, S. Kawamura, and F. Miyazaki. "Bettering operation of robots by learning". In: *Journal of Robotic systems* 1.2 (1984), pp. 123–140.

[34] M. Norrlöf and S. Gunnarsson. "Time and frequency domain convergence properties in iterative learning control". In: *International Journal of Control* 75.14 (2002), pp. 1114–1126.

[35] D.A. Bristow, M. Tharayil, and A.G. Alleyne. "A survey of iterative learning control". In: *Control Systems, IEEE* 26.3 (2006), pp. 96–114.

[36] C.G. Atkeson, C.H. An, and J.M. Hollerbach. "Estimation of inertial parameters of manipulator loads and links". In: *The International Journal of Robotics Research* 5.3 (1986), pp. 101–119.

[37] J.-J. Slotine and W. Li. "On the adaptive control of robot manipulators". In: *The International Journal of Robotics Research* 6.3 (1987), pp. 49–59.

[38] W.S. Cleveland and S.J. Devlin. "Locally weighted regression: an approach to regression analysis by local fitting". In: *Journal of the American Statistical Association* 83.403 (1988), pp. 596–610.

[39] S. Schaal, C.G. Atkeson, and S. Vijayakumar. "Scalable techniques from nonparametric statistics for real time robot learning". In: *Applied Intelligence* 17.1 (2002), pp. 49–60.

[40] D. Nguyen-Tuong, M. Seeger, and J. Peters. "Computed torque control with nonparametric regression models". In: *American Control Conference, 2008*. IEEE. 2008, pp. 212–217.

[41] C.E. Rasmussen. *Gaussian processes for machine learning*. Cambridge, MA: The MIT Press, 2006.

[42] P. Pastor et al. "Learning task error models for manipulation". In: *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2013, pp. 2612–2618.

[43] J. Peters and S. Schaal. "Learning to control in operational space". In: *The International Journal of Robotics Research* 27.2 (2008), pp. 197–212.

[44] D.-S. Kwon and W.J. Book. "A time-domain inverse dynamic tracking control of a single-link flexible manipulator". In: *Journal of dynamic systems, measurement, and control* 116.2 (1994), pp. 193–200.

[45] W. Singhose. "Command shaping for flexible systems: A review of the first 50 years". In: *International Journal of Precision Engineering and Manufacturing* 10.4 (2009), pp. 153–168. ISSN: 1229-8557.

[46] W. Chen and M. Tomizuka. "Dual-Stage Iterative Learning Control for MIMO Mismatched System With Application to Robots With Joint Elasticity". In: *Control Systems Technology, IEEE Transactions on* (2013).

[47] P.N. Yianilos. "Data structures and algorithms for nearest neighbor search in general metric spaces". In: *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*. 1993, pp. 311–321.

[48] R.M. Neal. "Bayesian Learning for Neural Networks". In: *Lecture Notes in Statistics 118*. New York, NY: Springer-Verlag, 1996.

[49] A. Schirru et al. "Efficient Marginal Likelihood Computation for Gaussian Process Regression". In: *arXiv preprint arXiv:1110.6546* (2011).

[50] C.-C. Lo and C.-Y. Hsiao. "A method of tool path compensation for repeated machining process". In: *International Journal of Machine Tools and Manufacture* 38.3 (1998), pp. 205–213.

[51] R. Ramesh, M.A. Mannan, and A.N. Poo. "Tracking and contour error control in CNC servo systems". In: *International Journal of Machine Tools and Manufacture* 45.3 (2005), pp. 301–326.

[52] R.W. Zarkos and G.F. Rogers. "A complete algorithm for computing area and center of gravity for polygons". In: *Computers & Geosciences* 13.5 (1987), p. 561.

[53] C. Ricolfe-Viala and A.-J. Sanchez-Salmeron. "Optimal conditions for camera calibration using a planar template". In: *Proceedings of the 18th IEEE International Conference on Image Processing*. 2011, pp. 853–856.

[54] B. Armstrong. "On finding 'exciting' trajectories for identification experiments involving systems with non-linear dynamics". In: *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. Mar. 1987, pp. 1131–1139.

[55] M. Gautier and W. Khalil. "Exciting Trajectories for the Identification of Base Inertial Parameters of Robots". In: *International Journal of Robotics Research* 11.4 (Aug. 1992), pp. 362–375.

[56] J. Swevers et al. "Optimal robot excitation and identification". In: *IEEE Transactions on Robotics and Automation* 13.5 (1997), pp. 730–740.

[57] W. Rackl, R. Lampariello, and G. Hirzinger. "Robot excitation trajectories for dynamic parameter estimation using optimized B-splines". In: *2012 IEEE International Conference on Robotics and Automation*. May 2012, pp. 2042–2047.

[58] R. Braatz and M. Morari. "Minimizing the Euclidean Condition Number". In: *SIAM Journal on Control and Optimization* 32.6 (1994), pp. 1763–1768.

[59] H. Niederreiter. "Low-discrepancy and low-dispersion sequences". In: *Journal of Number Theory* 30.1 (1988), pp. 51–70.

[60] P. Bratley and B.L. Fox. "Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator". In: *ACM Transactions on Mathematical Software* 14.1 (Mar. 1988), pp. 88–100.

[61] S. Joe and F.Y. Kuo. "Remark on Algorithm 659: Implementing Sobol's quasirandom sequence generator". In: *ACM Transactions on Mathematical Software* 29.1 (Mar. 2003), pp. 49–57.

[62] T. Chan and P. Hansen. "Some Applications of the Rank Revealing QR Factorization". In: *SIAM Journal on Scientific and Statistical Computing* 13.3 (1992), pp. 727–741.

[63] M. Gu and S. Eisenstat. "Efficient Algorithms for Computing a Strong Rank-Revealing QR Factorization". In: *SIAM Journal on Scientific Computing* 17.4 (1996), pp. 848–869.

[64] J.A. Tropp. "Column Subset Selection, Matrix Factorization, and Eigenvalue Optimization". In: *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. New York, New York, Jan. 2009, pp. 978–986.

[65] J. Craig, P. Hsu, and S.S. Sastry. "Adaptive control of mechanical manipulators". In: *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*. Vol. 3. Apr. 1986, pp. 190–195.

[66] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. Cambridge, MA: MIT Press, 1993.

[67] X. Yu et al. "Dynamics Modeling and Identification of a dual-blade wafer handling robot". In: *Proceedings of the 2013 ASME Dynamic Systems and Control Conference*. Oct. 2013.

[68] H. Mayeda, K. Osuka, and A. Kanagawa. "A New Identification Method for Serial Manipulator Arms". In: *Proceedings of the IFAC 9th World Congress*. July 1984, pp. 74–79.

[69] B. Siciliano et al. *Robotics: Modelling, Planning and Control*. Springer-Verlag, Berlin, 2009.

[70] T. Kroger and F.M. Wahl. "Stability and Robustness Analysis of a Class of Adaptive Controllers for Robotic Manipulators". In: *The International Journal of Robotics Research* 9.3 (June 1990), pp. 74–92.

[71] R. Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 58 (1 1996), pp. 267–288.

[72] R. Tibshirani. "Regression shrinkage and selection via the lasso: a retrospective". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73 (3 2011), pp. 273–282.

[73] H. Wang, B. Li, and C. Leng. "Shrinkage tuning parameter selection with a diverging number of parameters". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71.3 (2009), pp. 671–683.

[74] R. Abbe and D. Baker. *Rotary to Linear Motion Robot Arm*. US Patent 4897015. Jan. 1990.

[75] A. Todorov. *Robot with Belt-Drive System*. US Patent 2008/0121064 A1. May 2008.

[76] B. Till. "Dynamical Model of a Three-Link Remotely-Driven Vacuum Robot". In: *Proceedings of the 2010 Applied Materials ET Conference*. 2010.

[77] C. Wang, X. Yu, and M. Tomizuka. "Fast Modeling and Identification of Robot Dynamics using the Lasso". In: *Proceedings of the 2013 ASME Dynamic Systems and Control Conference*. Oct. 2013.

[78] C. Wang et al. "Fast Planning of Well Conditioned Trajectories for Model Learning". In: *Proceedings of the International Conference on Intelligent Robots and Systems*. Oct. 2014.