2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Pose Estimation in Industrial Machine Vision Systems Under Sensing Dynamics: A Statistical Learning Approach

Chung-Yen Lin, Cong Wang, and Masayoshi Tomizuka[1]

***Abstract*— This paper deals with the problem of pose estimation (i.e., estimating position and orientation of an moving target) for real-time visual servoing, where the vision hardware is assumed to have severely limited measurement capability. In other words, we aim to compensate the slow sensor dynamics in industrial machine vision systems. The common approach is to predict the present target motion by propagating the delayed estimates with the target dynamics. Such method is sometimes problematic since the target motion characteristics (i.e., target dynamics) may change from one visual servoing task to another. Therefore, this paper presents a method which is able to estimate the target pose as well as learn the target dynamics. We apply the Expectation-Maximization algorithm to simultaneously solve the pose estimation problem and the target dynamics modeling problem. Several techniques including the extended Kalman filter/smoother, the block coordinate descent method, and the convex optimization method are utilized to address this problem. The effectiveness of the proposed algorithm is demonstrated experimentally on a 6-DOF industrial robot.**

I. INTRODUCTION

For vision hardware used in industrial applications, the response time and the computational power are often limited due to the cost issue. Therefore, the large sensing latency induced by image acquisition and processing will consequently make the sampling rate low. We refer these characteristics (i.e., large sensing latency and slow sampling rate) as the sensing dynamics associated with the vision system [1]. In this paper, we consider the problem of pose estimation (i.e., estimating position and orientation of an moving target [2]) for real-time visual servoing (in particular, the position based visual servoing [3]), where the vision hardware is assumed to have non-negligible sensing dynamics. A main challenge of this problem is that the unknown target poses may have complex dependencies on the target dynamics, whereas the target motion characteristics (i.e., target dynamics) may change from one visual servoing task to another. To overcome this problem, several techniques have been proposed [4]. These methods attempt to adapt different target motion characteristics by tuning the state estimator gain, in particular the noise covariance matrices in the extended Kalman filter. However, in the case when the vision system has a large feedback time delay (i.e., large sensing latency), these algorithms may no longer be useful. Namely, the present target motion needs to be predicted by propagating the delayed state estimates with the target dynamics, but such dynamic propagation cannot be affected by the estimator gain. Therefore, it is difficult for these methods to provide accurate predicted states without parameter adaptation for the target dynamics. Furthermore, most existing methods assume the states between two consecutive measurements to be constants. This assumption is not adequate for many control applications since the control input is often updated at a faster rate than the sensor sampling rate. For these reasons, we present a VSD compensator (or VSDC) to jointly estimate all the target pose, the estimator gain, and the target dynamics in a multi-rate framework.

We formulate the target dynamics, the sensing dynamics, and the image projection as an autoregressive (AR) model cascading with a static nonlinear function. The unknown target dynamics is parameterized by the AR coefficients and the noise statistics. Once the model is designed, we apply the Expectation-Maximization (EM) algorithm [5] to jointly estimate the model parameters and the system states. Several techniques including the extended Kalman filter (EKF), the extended Kalman smoother (EKS), the block coordinate descent (BCD) method [6], and the convex optimization [7] are applied to obtain the information we need in each EM iteration. The effectiveness of the proposed pose estimation algorithm is demonstrated by experimentation on the FANUC M-16*i*B robot setup.

II. MODEL

A. Target Dynamic Model

The motion of a rigid body can be generally described by a translation and a rotation [8]. Among various choices of the motion description, we use the Cartesian coordinates $\{X, Y, Z\}$ and the Z-Y-X Euler angles $\{\theta, \phi, \psi\}$ (i.e., yaw, pitch, roll) [8] for describing the translation and the rotation, respectively. In this paper, we denote the coordinates $\{X, Y, Z, \theta, \phi, \psi\}$ in the camera frame by $\{z_1, \ldots, z_6\}$ for simplicity.

For each coordinate z_i, the target dynamics is represented by an autoregressive (AR) model:

$$z_i(j) = -\alpha_{i1} z_i(j-N) - \alpha_{i2} z_i(j-2N) - \cdots - \alpha_{in} z_i(j-nN) + w_i(j) \quad (1)$$

where j is the time index of the controller, N is the sampling interval of the vision system, $\alpha_{i\bullet}$ are unknown model parameters, and n is the order of the AR model. The state vector z_i is considered to be random. We assume that the process noise w_i has independent and identically distributed (i.i.d.) Gaussian distribution $\mathcal{N}(0, \sigma_{wi}^2)$ at each time step.

*This work was supported by FANUC Corporation, Japan.

[1]Chung-Yen Lin, Wang Cong, and Masayoshi Tomizuka are with Department of Mechanical Engineering, University of California, Berkeley, CA 94720, USA. Email: {chung_yen, wangcong, tomizuka} at berkeley.edu

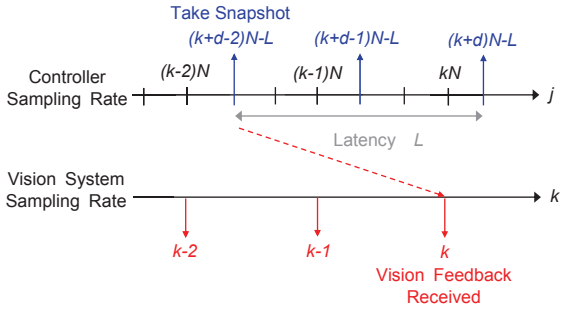


Fig. 1: Visualization of the multi-rate model for the vision feedback system

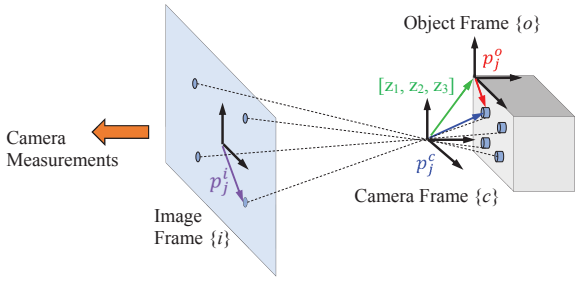


Fig. 2: Coordinate transformation and perspective projection

Let k denote the time index of the vision system. The latency of vision feedback is assumed to be L times of the controller sampling rate. The latency is also defined in the vision system sampling rate by dividing L by N and rounding up to the nearest integer, namely $d = \lceil \frac{L}{N} \rceil$, where $\lceil \bullet \rceil$ is the ceiling function. A visualization of this multi-rate model is shown in Fig. 1. Then, we define the lifted state vector $x_i(k)$ and the lifted process noise $n_{wi}(k)$ with respect to the vision system sampling rate as follows:

$$x_i(k) = \begin{bmatrix} z_i(j = (k+d-n)N - L) \\ z_i(j = (k+d-n+1)N - L) \\ \vdots \\ z_i(j = (k+d-1)N - L) \end{bmatrix} \tag{2}$$

$$n_{wi}(k) = \left[\mathbf{0}_{n-1}^{\mathrm{T}}, w_i(j = (k+d-1)N - L)\right]^{\mathrm{T}} \tag{3}$$

To further simplify the expression, we substitute (2)-(3) into (1) to yield:

$$x_i(k+1) = A_{(i)} x_i(k) + n_{wi}(k)$$

where $A_{(i)} \in \mathbb{R}^{n \times n}$ is the transition matrix given by:

$$A_{(i)} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -\alpha_{in} & -\alpha_{i,n-1} & -\alpha_{i,n-2} & \dots & -\alpha_{i1} \end{bmatrix}$$

Then, the overall target dynamic model combined with the dynamics of z_1, z_2, …, z_6 is obtained as:

$$x(k+1) = Ax(k) + n_w(k) \tag{4}$$

where $x := [x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, \dots, x_6^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6n}$ is the augmented state vector, $n_w := [n_{w1}^{\mathrm{T}}, n_{w2}^{\mathrm{T}}, \dots, n_{w6}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6n}$, while $A \in \mathbb{R}^{6n \times 6n}$ is a block diagonal matrix defined by:

$$A = A_{(1)} \oplus A_{(2)} \oplus \cdots \oplus A_{(6)}$$

The noise covariance matrix of the process noise vector n_w is denoted as $\Sigma_w \in \mathbb{R}^{6n \times 6n}$.

B. Multiple Markers with Single Camera

While the coordinates $\{z_1, \dots, z_6\}$ are not measured directly, these quantities can still be estimated by combining multiple measurements in a single image. To be precise, the target (i.e., the rigid body to be estimated) is considered as a collection of several feature points, where the relative positions of these points with each other are assumed to be known and invariant. This assumption is usually valid since the CAD drawings of the targets are available for most industrial applications. Based on the assumption, an one-to-one mapping between the coordinates $\{z_1, \dots, z_6\}$ and the feature points projection on image space can be obtained. We hence can estimate the target motion using the mapping and the image measurements.

To establish this mapping, we first consider the camera configuration shown in Fig. 2, where $\{c\}$, $\{o\}$, and $\{i\}$ represent the camera frame, the object frame (or target frame), and the image frame, respectively. The vectors $p_j^o = \begin{bmatrix} p_{j1}^o & p_{j2}^o & p_{j3}^o \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3$, $p_j^c = \begin{bmatrix} p_{j1}^c & p_{j2}^c & p_{j3}^c \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3$, and $p_j^i = \begin{bmatrix} p_{j1}^i & p_{j2}^i \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2$ are position vectors of the j-th feature point described in the frames $\{o\}$, $\{c\}$, and $\{i\}$, respectively.

Since the position of the j-th feature point on the frame $\{o\}$ is known (i.e., p_j^o is known), the vector p_j^c is expressed as:

$$p_j^c = R\left(z_4, z_5, z_6\right) p_j^o + \left[z_1, z_2, z_3\right]^{\mathrm{T}} \tag{5}$$

where $R\left(z_4, z_5, z_6\right) \in \mathbb{R}^{3 \times 3}$ is the rotation matrix in the camera frame $\{c\}$ for describing the transformation from $\{c\}$ to $\{o\}$. By using the perspective projection model [2], we can project the position vector p_j^c onto the image space as:

$$p_j = \begin{bmatrix} \frac{F p_{j1}^c}{p_{j3}^c} & \frac{F p_{j2}^c}{p_{j3}^c} \end{bmatrix}^{\mathrm{T}} \tag{6}$$

where F denotes the focal length of camera. Then, we collect the coordinate of the feature points (said, f features) from image as the output vector $y := [p_1^{i\mathrm{T}}, p_2^{i\mathrm{T}}, \cdots, p_f^{i\mathrm{T}}]^{\mathrm{T}}$ so that the measurement model is obtained by:

$$y(k) = g(z(k-d), p_1^o(k-d), \dots, p_f^o(k-d)) + n_v(k) \tag{7}$$

Here, the nonlinear function $g(\cdot)$ accounts for both the coordinate transformation (5) and the perspective projection

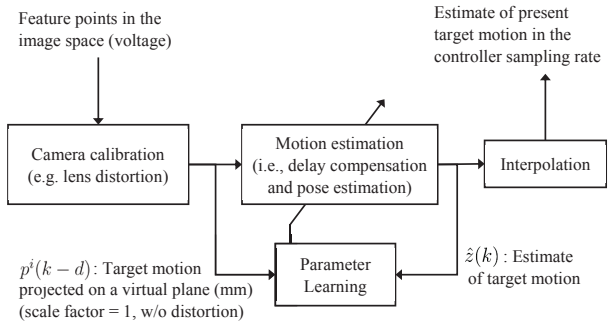


Fig. 3: Structure of the VSD compensator

(6). f is the number of feature points. $n_v \in \mathbb{R}^{2f}$ is the measurement noise vector of the sensor system that is assumed to have independent and identically distributed (i.i.d.) zero-mean Gaussian distribution with covariance Σ_v. As mentioned, the positions of the feature points are invariant relative to $\{o\}$ (i.e., $p_1^o, \ldots, p_f^o$ are known constants) in most applications. We thus can simplify (7) as:

$$y(k) = g(x(k)) + n_v(k) \tag{8}$$

where $x(k)$ is the state vector defined previously.

III. ALGORITHM

A. Basic Structure

Figure 3 shows the overall structure of the visual sensing dynamic compensator. The camera measurements are first passed through a calibration block for distortion correction. Once this is done, the measurement model in (8) can be used to describe the relationship between the object motion and the corresponding feature information projected on the virtual plane. The remaining VSD compensation problem is formulated into two parts: 1) we first try to learn the model parameters (i.e., the parameter learning block) that can describe the target motion characteristics under the sensing dynamics; 2) we then use the estimated model along with sampled data to estimate/predict the present target motion (i.e., the motion estimation block). The details of the design for each part of the VSD compensator will be discussed in the following sections.

B. Camera Calibration

The camera calibration block is used to correct the nonlinear effects in vision systems. In typical industrial machine vision systems, the nonlinear effects may include the lens distortion and the nonlinear response of the signal processing circuit and the A/D channels. In this paper, we apply the calibration process proposed in [9] for compensating these effects. A look-up table is built to map the camera measurements (contaminated by the lens distortion) from the sensing plane to a virtual plane where the nonlinear effects are fully corrected. The process of building the look-up table was detailed in [9].

C. Parameter Learning

Given observations of target features $Y(m) = \{y(1), \cdots, y(m)\}$ on image space, we aim to jointly estimate the model of the target dynamics $\Theta := \{A, \Sigma_v, \Sigma_w\}$ as well as the sequence of unknown target poses (i.e., $x(k)$, $k = 1, \ldots, m$). This problem can be formulated as the Expectation-Maximization (EM) algorithm framework by alternatively maximizing the likelihood of the model parameters for a given set of pose estimates (M-step); and computing the expected value for the pose estimates for a given target dynamics (E-step). Detailed derivation of the EM algorithm can be found in [5].

In each EM iteration, we first compute the conditional expectation of the complete log-likelihood using the estimated parameter in the previous iteration. The expected complete log likelihood is obtained as:

$$\mathcal{L}(\Theta; Y(m)) = E\left\{ \sum_{k=2}^{m} \log \mathcal{N}(x(k)|Ax(k-1), \Sigma_w) + \sum_{k=1}^{m} \log \mathcal{N}(y(k)|g(x(k)), \Sigma_v) \middle| Y(m) \right\} \tag{9}$$

where $E\{\bullet|Y(m)\}$ represents the conditional expectation of $\bullet$ computed with the current estimated parameter, while $\mathcal{N}(\bullet|\mu, \sigma^2)$ denotes the probability of $\bullet$ under a normal distribution with mean μ and variance σ^2.

We then maximize the function $\mathcal{L}(\Theta; Y(m))$ with respect to Θ to complete an EM iteration. Solving such optimization problem, however, is still a difficult task since the objective function (9) is not jointly concave over Θ. Although we cannot yield the optimum solution directly, we can use the block coordinate descent (BCD) technique to achieve a local optimum. That is, we alternatively chose a set of parameters (i.e., either A, Σ_w, or Σ_v) as the variable for parameter estimation, while the remaining parameters are assumed to have fixed values. For example, in the case of estimating the transition matrix A, we begin with writing down the explicit expression of the likelihood function as follows:

$$\mathcal{L}(A, \hat{\Sigma}_w, \hat{\Sigma}_v; Y(m)) = -\sum_{k=1}^{m} E\left\{ \tilde{y}^{\mathrm{T}}(k) \frac{\hat{\Sigma}_v^{-1}}{2} \tilde{y}(k) \middle| Y(m) \right\} - \sum_{k=2}^{m} E\left\{ \tilde{x}^{\mathrm{T}}(k, A) \frac{\hat{\Sigma}_w^{-1}}{2} \tilde{x}(k, A) \middle| Y(m) \right\} + \frac{m-1}{2} \log\det(\hat{\Sigma}_w^{-1}) + \frac{m}{2} \log\det(\hat{\Sigma}_v^{-1}) + c \tag{10}$$

where $\tilde{x}(k, A) := x(k) - Ax(k-1)$ and $\tilde{y}(k) := y(k) - g(x(k))$. c is a constant term accounts for the initial condition and some constant coefficients in the distribution. Here, we assume that the parameters $\hat{\Sigma}_w$ and $\hat{\Sigma}_v$ are fixed with the values equal to the current estimates obtained from the previous EM iteration. Since the transition matrix A is the only variable in this expression, the problem can be simplified as a quadratic programming (QP) with the matrix

variable A:

$$\min_{A} \quad \sum_{k=2}^{m} E\left\{\tilde{x}^{\mathrm{T}}(k,A)\hat{\Sigma}_w^{-1}\tilde{x}(k,A)\middle| Y(m)\right\}$$
$$\text{subject to} \quad A_{ij} = \begin{cases} 1 & \text{for } (i,j) \in \mathcal{S}_{A1} \\ 0 & \text{for } (i,j) \in \mathcal{S}_{A2} \end{cases} \tag{11}$$

where the sets $\mathcal{S}_{A1}$ and $\mathcal{S}_{A2}$ are introduced to enforce the identified matrix A has the desired structure as defined in Sec. II-A.

Similarly, to estimate the measurement noise covariance Σ_v, we set the transition matrix $\hat{A}$ and the precess noise covariance $\hat{\Sigma}_w$ equal to the current estimates; and then optimize the objective function $\mathcal{L}(\hat{A}, \hat{\Sigma}_w, \Sigma_v; Y(m))$ over the matrix variable Σ_v^{-1}:

$$\min_{\Sigma_v^{-1} \succ 0} \quad -m \log\det(\Sigma_v^{-1}) + \text{trace}\left(\Sigma_v^{-1} D_v\right) \tag{12}$$

where D_v is a positive definite data matrix given by:

$$D_v = \sum_{k=1}^{m} E\left\{\tilde{y}(k)(\tilde{y}(k))^{\mathrm{T}}\middle| Y(m)\right\} \tag{13}$$

where the property $\text{trace}(AB) = \text{trace}(BA)$ is utilized to simplify the expression.

Finally, the third minimization problem is obtained by optimize the objective function $\mathcal{L}(\hat{A}, \Sigma_w, \hat{\Sigma}_v; Y(m))$ over the matrix variable Σ_w^{-1}:

$$\min_{\bar{\Sigma}_w^{-1} \succ 0} \quad -(m-1)\log\det(\bar{\Sigma}_w^{-1}) + \text{trace}\left(\bar{\Sigma}_w^{-1} D_w\right)$$
$$\text{subject to} \quad \left(\bar{\Sigma}_w^{-1}\right)_{ij} = 0 \ \ \forall i,j \in \{1,2,\ldots,6\}, j \neq i \tag{14}$$

where $\bar{\Sigma}_w := \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_6^2)$ is a submatrix of Σ_w which corresponds to non-zero eigenvalues of Σ_w. The submatrix $\bar{\Sigma}_w$ is introduced to ensure the matrix variable is invertible. D_w is a positive definite data matrix defined as:

$$D_w = \sum_{k=2}^{m} E\left\{W\tilde{x}(k,\hat{A})\tilde{x}^{\mathrm{T}}(k,\hat{A})W^{\mathrm{T}}\middle| Y(m)\right\}$$

where the matrix $W \in \mathbb{R}^{6\times 6n}$ is structured as:

$$W_{ij} = \begin{cases} 1 & \text{for } (i,j) \in \mathcal{S}_w \\ 0 & \text{for } (i,j) \notin \mathcal{S}_w \end{cases}$$
$$\mathcal{S}_w = \{i,j | i \in \{1,2,\ldots,6\}, j = ni\}$$

Again, the constraint is utilized to preserve the desired structure of the noise covariance matrix. Most importantly, since each of the optimization problem in (11)-(14) has a convex objective function with equality constraints, the optimal solutions of A, Σ_v, and Σ_w can be found directly by introducing Lagrange multipliers and solving the Karush-Kuhn-Tucker (KKT) conditions [7].

The next step is to compute the expected values of the state vector and its corresponding covariances for the KKT conditions and the data matrices. This can be done by applying the extended Kalman filtering/smoothing (EKF/EKS) techniques to the model. The EKF formulation is given by:

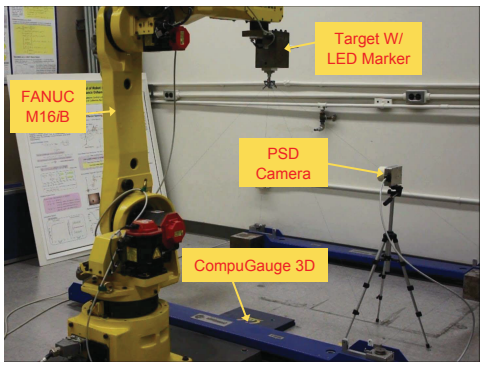


Fig. 4: Experimental setup

$$M(k) = \hat{A}Z(k-1)\hat{A}^{\mathrm{T}} + \hat{\Sigma}_w \tag{15}$$
$$C_i(k) = \left[\mathbf{0}_{2f\times(n-d-1)} \quad \left.\frac{\partial g(z)}{\partial z_i}\right|_{x=\hat{A}\hat{x}(k-1)} \quad \mathbf{0}_{2f\times d}\right] \tag{16}$$
$$C(k) = \left[C_1(k), C_2(k), \cdots, C_6(k)\right] \in \mathbb{R}^{2f\times 6n} \tag{17}$$
$$K(k) = M(k)C(k)\left(C(k)M(k)C^{\mathrm{T}}(k) + \hat{\Sigma}_v\right)^{-1} \tag{18}$$
$$\hat{x}(k) = \hat{A}\hat{x}(k-1) + K(k)\left(y(k) - g(\hat{A}\hat{x}(k-1))\right) \tag{19}$$
$$Z(k) = (I - K(k)C(k))M(k) \tag{20}$$

where $\hat{x}(k) = E\{x(k)|Y(k)\}$ is the *a-posteriori* estimates and $Z(k) = E\left\{(x(k)-\hat{x}(k))(x(k)-\hat{x}(k))^{\mathrm{T}}|Y(k)\right\}$ is the corresponding covariance matrix. $\frac{\partial g(z)}{\partial z_i}$ is the i-th column of the Jacobian matrix of $g(z)$, which can be computed efficiently by a closed form derivation.

Then, by applying the EKS, we obtain:

$$J(k) = Z(k)\hat{A}^{\mathrm{T}}M^{-1}(k+1) \tag{21}$$
$$S(k) = Z(k) + J(k)(S(k+1) - Z(k+1))J^{\mathrm{T}}(k) \tag{22}$$
$$\hat{x}^s(k) = \hat{x}(k) + J(k)\left(\hat{x}^s(k+1) - \hat{x}(k+1)\right) \tag{23}$$

where $\hat{x}^s(k) = E\{x(k)|Y(m)\}$ is the state estimate after smoothing (i.e., conditioning on $Y(m)$) and $S(k) = E\left\{(x(k)-\hat{x}^s(k))(x(k)-\hat{x}^s(k))^{\mathrm{T}}|Y(m)\right\}$ is the corresponding covariance.

Once the expected values in (22)-(23) are computed, we can continue to solve the optimization problems (11)-(14) using the corresponding KKT condition. The algorithm is running iteratively until the model parameters converge. While the estimated parameters obtained by the BCD algorithm may not necessarily achieve the global optimum for the original objective function (9), a local optimum can still be guaranteed if the algorithm converges to a limit point [6].

D. Motion Estimation

Note that once the model parameter learning is done, we can design a state observer for real-time target state estimation based on the estimated model obtained in Section III-C. One possible way to do this is directly implementing the

EKF as shown in (15)-(20). However, since the EKF in (15)-(20) is formulated at the vision system sampling rate, we need to approximate the states between measurement points (i.e., the state estimates at the controller sampling rate) by interpolation. To be precise, every time when a measurement is arrived at $j = kN$, we use the following formula to provide an estimate of the current target motion in the controller sampling rate:

$$\hat{z}(j) \approx \hat{z}(j_{\text{pre}}) + \frac{j - j_{\text{pre}}}{j_{\text{next}} - j_{\text{pre}}} \left(\hat{z}(j_{\text{next}}) - \hat{z}(j_{\text{pre}})\right)$$

where j_{next} and j_{pre} are time indices corresponding to the nearest state estimates around the time step j. For example, in the case of estimating $\hat{z}(j = kN+\ell)$ with $\ell \in [0, dN-L)$, we have the nearest future point $j_{\text{next}} = (k+d)N - L$ and the nearest past point $j_{\text{pre}} = (k+d-1)N - L$. Similarly, in another case when $\ell \in [dN - L, kN)$, we have $j_{\text{next}} = (k+d+1)N - L$ and $j_{\text{pre}} = (k+d)N - L$. Therefore, we obtain the linear interpolation formula in terms of the state vector x as follows:

$$\hat{z}(j = kN + \ell)$$

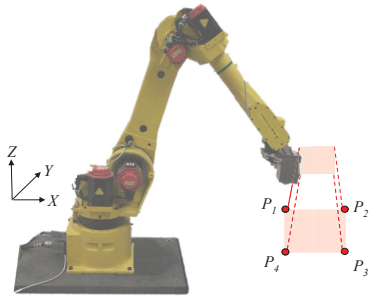
$$\approx \begin{cases} \text{if} \quad 0 \le \ell < dN - L: \\ \quad \gamma \ \text{diag}\left(\left[\mathbf{0}_{n-2}^{\mathrm{T}}, 1, 0\right], \ldots, \left[\mathbf{0}_{n-2}^{\mathrm{T}}, 1, 0\right]\right) \hat{x}(k) \\ \quad + (1-\gamma) \ \text{diag}\left(\left[\mathbf{0}_{n-1}^{\mathrm{T}}, 1\right], \ldots, \left[\mathbf{0}_{n-1}^{\mathrm{T}}, 1\right]\right) \hat{x}(k) \\ \text{if} \quad dN - L \le \ell < N: \\ \quad (1+\gamma) \ \text{diag}\left(\left[\mathbf{0}_{n-1}^{\mathrm{T}}, 1\right], \ldots, \left[\mathbf{0}_{n-1}^{\mathrm{T}}, 1\right]\right) \hat{x}(k) \\ \quad -\gamma \ \text{diag}\left(\left[\mathbf{0}_{n-1}^{\mathrm{T}}, 1\right], \ldots, \left[\mathbf{0}_{n-1}^{\mathrm{T}}, 1\right]\right) \hat{A}\hat{x}(k) \end{cases}$$

where $\gamma := d - \frac{\ell+L}{N}$.

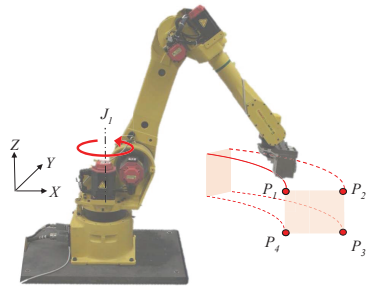
IV. EXPERIMENTAL STUDY

A. Test Setup

The proposed compensator is implemented on the 6-joint FANUC M-16*i*B robot setup as shown in Fig. 4. The system is equipped with built-in motor encoders and a three-dimensional position measurement system, the CompuGauge 3D (CG3D) system, for measuring the tool center point (TCP) position in Cartesian space. Additionally, a Position Sensitive Detector (PSD) based optical position sensor [9] is used to acquire the vision information. Unlike the regular camera system which requires image processing for obtaining feature information, the PSD camera senses the position of infrared beacons directly. Therefore, it can be sampled at high sampling rate without consuming much hardware resource. The sampling rates of all the sensor signals as well as the robot controller are set to be 1kHz, i.e. the sampling period of motion control is 1ms. The desired target position is generated by an infrared LED marker mounted on the robot end-effector. To mimic typical sensing capability of industrial machine vision systems, the output signal of the PSD camera is down-sampled to 30.3Hz with a fictitious latency 33ms (i.e., $d = 1$ and $L = 33$). The estimated target motion will be compared with the CG3D measurements for performance evaluation.



(a) Test setup 1: straight line (along Y-axis) without orientation change



(b) Test setup 2: curve with orientation change (only J_1 moves)

Fig. 5: Test setups for different types of target motion

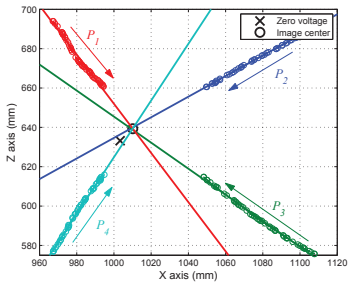
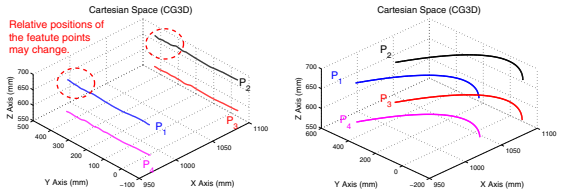


Fig. 6: Find the image center using least squares method

The algorithm is tested for two different motion patterns, namely, a straight line without rotation (test setup 1) and a curve with rotation (test setup 2). Due to the hardware limitations, the system is only equipped with one marker rather than multiple markers. We thus designed several TCP trajectories to mimic the motion of a virtual rigid body with vertecies P_1, P_2, P_3, and P_4 (as shown in Fig. 5). To be precise, we assume that the image data is generated by different markers (i.e., P_1, P_2, P_3, and P_4) mounted on a single rigid body, while these signals are actually measured separately. By using the PSD camera, we can obtain the 2D position measurement of each individual marker.

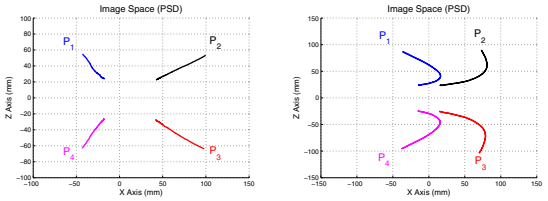
B. Finding the Image Center

It should be noted that the center of the image plane (i.e., the origin of the image plane $\{i\}$) of the PSD camera may not necessarily correspond to the point with zero voltage



(a) Test setup 1 (b) Test setup 2

Fig. 7: CG3D measurements



(a) Test setup 1 (b) Test setup 2

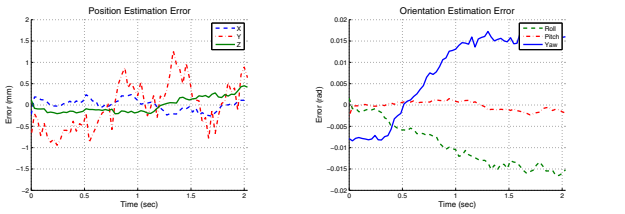
Fig. 8: PSD camera measurements (after camera calibration)

measurement. We thus need to estimate the image center before applying the VSD compensator to the measured signals. One straightforward way to identify the image center is to move markers in the Y-direction and find their corresponding principal point using the measurements on image plane. As shown in Fig. 6, we apply least squares method to fit a straight line for each sequence of measurements. The point that has the minimum distance to the line intersections will be considered as the image center.

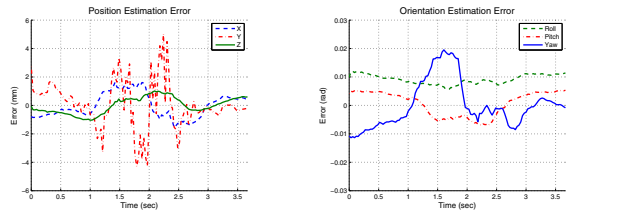
C. Experimental Results

The CG3D measurements and the PSD camera measurements for two different motion patterns are shown in Fig. 7 and Fig. 8. As emphasized by the red-dash circles, it is seen that relative positions of the feature points may change due to the robot tracking errors (e.g., the trajectory of each marker is not exactly a straight line as planned). This may break down the assumption that the feature point relative positions are invariant during the target motion, and hence may affect the performance of the VSD compensator. Figure 9 shows the corresponding estimation errors when we applied the VSD compensator for 10 iterations to the system. The model orders are selected to be $n = 4$ and $n = 6$ for the test setup 1 and 2, respectively. Here, the model order is determined by the Akaike information criterion (AIC) [10], which have been widely used for model selection. The position estimation errors are defined as the difference between the estimated translation vector (i.e., $\hat{z}_1$, $\hat{z}_2$, and $\hat{z}_3$) and the CG3D measurements, while the orientation estimation errors are defined as the difference between the estimated Euler angle (i.e., $\hat{z}_4$, $\hat{z}_5$, and $\hat{z}_6$) and the end-effector orientation computed by motor encoder measurements.

Figure 10 shows the convergence of the state estimation errors for the test setup 1 and 2, respectively. It shows that

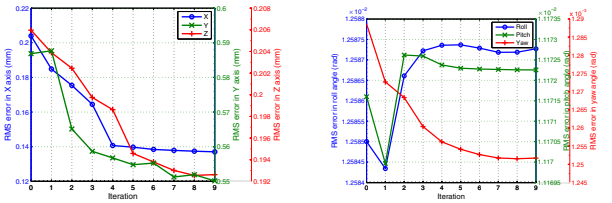


(a) Position error for test setup 1 (b) Orientation error for test setup 1

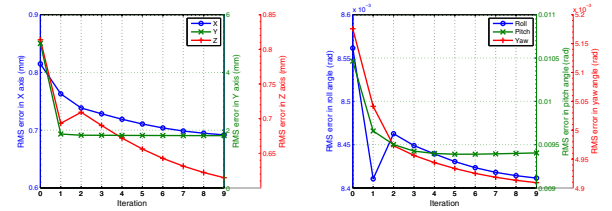


(c) Position error for test setup 2 (d) Orientation error for test setup 2

Fig. 9: State estimation error for different test setups



(a) Position error for test setup 1 (b) Orientation error for test setup 1



(c) Position error for test setup 2 (d) Orientation error for test setup 2

Fig. 10: Error convergence for different test setups

most of the estimation errors converge to smaller values after few iterations of the parameter learning algorithm, while the errors for both the roll angle and the pitch angle in the test setup 1 are actually even larger than the initial parameter setting. This result is not unreasonable since the coordinates of the target motion are coupled in the objective function (9) among the unknown parameters. That is, optimizing the overall objective function does not necessarily imply that the estimation error in every state will decrease. Although there is no guaranteed error reduction in every coordinate, the parameter learning algorithm is still beneficial since the decreasing of errors in z_1, z_2, z_3, and z_4 are much more significant than the increasing of errors in z_5 and z_6, which are quite small as seen in Fig. 10b.

Table I shows the root-mean-square errors of the estimation results when applying the proposed compensation algo-

TABLE I: Effectiveness of the VSD compensator with different settings

Test setup	Algorithm settings	Position errors (mm) e_1	e_2	e_3	$\|e_{1:3}\|_2$	Orientation errors (rad) e_4	e_5	e_6	$\|e_{4:6}\|_2$
1	VSD compensator	0.1942	0.5856	0.2020	0.6492	0.0126	0.0112	0.0012	0.0169
	VSDC w/o parameter learning	0.2042	0.5933	0.2040	0.6598	0.0126	0.0112	0.0013	0.0169
	VSDC w/o delay compensation	0.1899	3.3389	0.2066	3.3507	0.0127	0.0113	0.0013	0.0170
	EKF w/ linear extrapolation model	0.2002	3.3406	0.2086	3.3531	0.0127	0.0113	0.0013	0.0170
2	VSD compensator	0.6919	1.8113	0.6148	2.0341	0.0084	0.0094	0.0049	0.0135
	VSDC w/o parameter learning	0.8148	5.0068	0.8139	5.1376	0.0085	0.0104	0.0051	0.0144
	VSDC w/o delay compensation	2.0899	11.5738	0.6171	11.7772	0.0118	0.0093	0.0049	0.0158
	EKF w/ linear extrapolation model	2.1488	11.5738	0.6962	11.7922	0.0121	0.0096	0.0050	0.0162

rithm with different settings. More specifically, the delay step d is set as zero for the case of "without delay compensation", while the parameter learning block in Fig. 3 is disabled for the case of "without parameter learning". In the case when the parameter learning block is disabled, the linear extrapolation model (i.e., $\alpha_{i1} = -2$, $\alpha_{i2} = 1$, $\alpha_{ij} = 0$ for all $i \in 1, \ldots, 6$ and $j \in 3, \ldots, n$, similar to the model used in [11]) is utilized as the target dynamic model. The baseline estimator is designed as an EKF with the linear extrapolation model and manually tuned noise covariance matrices. The estimation error e_i (for $i = 1, \ldots, 6$) is defined as the root-mean square error in the direction of z_i. In both test setups, the estimation errors can be greatly reduced once the latency of the vision system is correctly compensated. In the test setup 1, it is seen that the parameter learning algorithm does not improve the performance significantly, which means that most of the estimation errors are induced by the sensor dynamics rather than the target dynamics. In the test setup 2, however, the results show that the parameter learning algorithm also play an important role (especially in z_2 direction) due to the complexity of the target motion pattern. In summary, the proposed VSD compensation scheme can provide superior pose estimation results in both test setups by 1) compensating the sensor dynamics and 2) correctly modeling the target motion characteristics.

V. CONCLUSIONS

This paper proposed a compensation method for alleviating the sensing dynamics issue in the position base visual servoing. By properly modeling the sensor dynamics and the target motion characteristics, we designed a state observer for real-time pose estimation. A learning procedure based on the Expectation-Maximization algorithm and the block coordinate descent method was developed for tuning the model parameters. The effectiveness of the algorithm was demonstrated experimentally on a 6-DOF industrial robot. It has been shown that the proposed method can provide superior state estimation performance under severely limited sensing capabilities.

REFERENCES

[1] P. I. Corke, "Dynamic issues in robot visual-servo systems," in *In Int. Symp. on Robotics Research ISRR95*. Springer, 1995, pp. 488–498.

[2] V. Lepetit and P. Fua, "Monocular model-based 3d tracking of rigid objects," *Found. Trends. Comput. Graph. Vis.*, vol. 1, no. 1, pp. 1–89, Jan 2005.

[3] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, 1996.

[4] F. Janabi-Sharifi and M. Marey, "A Kalman-filter-based method for pose estimation in visual servoing." *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 939–947, 2010.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubinr, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Statistical Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[6] P. Tseng and C. O. L. Mangasarian, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, pp. 475–494, 2001.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004.

[8] O. M. O'Reilly, *Intermediate dynamics for engineers: a unified treatment of Newton-Euler and Lagrangian mechanics*. New York, NY: Cambridge University Press, 2008.

[9] C. Wang, W. Chen, and M. Tomizuka, "Robot end-effector sensing with position sensitive detector and inertial sensors," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012.

[10] R. Shibata, "Selection of the order of an autoregressive model by akaike's information criterion," *Biometrika*, vol. 63, no. 1, pp. 117–126, 1976.

[11] A. Kosaka and G. Nakazawa, "Vision-based motion tracking of rigid objects using prediction of uncertanties." in *ICRA*. IEEE Computer Society, 1995, pp. 2637–2644.