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

Design of Kinematic Controller for Real-time Vision Guided Robot Manipulators*

Cong Wang, Chung-Yen Lin, and Masayoshi Tomizuka

Abstract— **This paper discusses the control strategies for robot manipulators to track moving targets based on real-time vision guidance. The work is motivated by some new demands of vision guided robot manipulators in which the workpieces being manipulated are in complex motion and the widely adopted look-then-move control strategy cannot give satisfactory performance. In order to serve industrial applications, the limited sensing and actuation capabilities have to be considered properly. In our work, 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. validation is conducted through simulation and experiment.**

I. INTRODUCTION

Over the past two decades, vision guided robot manipulators have been applied increasingly in various industries. In most scenarios, the workpieces being manipulated are either stationary (e.g., piled in a bin) or in simple and very predictable motion (e.g., carried by a belt conveyor that moves at known constant speed). In such cases, a simple look-then-move control strategy gives good performance. Recently, many new demands for vision guided industrial robots have raised. In some potential applications, the motion of the workpiece is less predictable, and the look-then-move strategy can hardly accomplish the tasks. Fig. 1 shows an example in which the robot needs to pick up a swinging workpiece from a hanging conveyor. In this task, the vision sensor has to provide real-time measurement of the workpiece's motion. Based on the real-time vision feedback, robot end-effector first approaches the workpiece and moves synchronously with it. Then, the grasping action is performed. There are two key aspects in realizing real-time vision guidance for industrial robot manipulators:

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 sensing dynamics* [1]. It makes real-time tracking of a moving target difficult. For this reason, *visual sensing dynamics compensation (VSDC)* is needed.

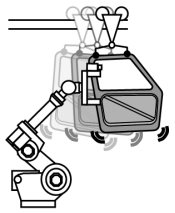


Fig. 1. An application of real-time vision guidance

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. We call this add-on *kinematic visual servoing (KVS)*.

Our previous papers [2] and [3] elaborate on the first aspect mentioned above, whereas this paper focuses on the second aspect, i.e., design of the KVS algorithms. In particular, actuator limits are considered. In most publications about visual servoing, the robot is assumed to have simple low order linear response to velocity or position command, and actuator limits have often been ignored [4]–[7]. Through out this paper, it is assumed that an image processing module can identify the position of the target in the image with a reasonable sampling rate, latency, and noise.

II. KINEMATIC VISUAL SERVOING

Assume one of the VSDC algorithms presented in [2] and [3] is used to compensate the low sampling rate and large latency introduced in image acquisition and processing, Fig. 2 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

*This work was supported by FANUC Corporation. Some control hardware and software were donated by National Instruments Corporation.

The authors are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720-1740, USA {wangcong, chung_yen, tomizuka}@berkeley.edu

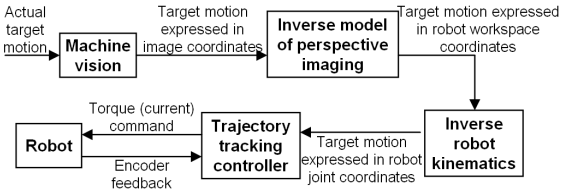


Fig. 2. A simple method for visual servoing

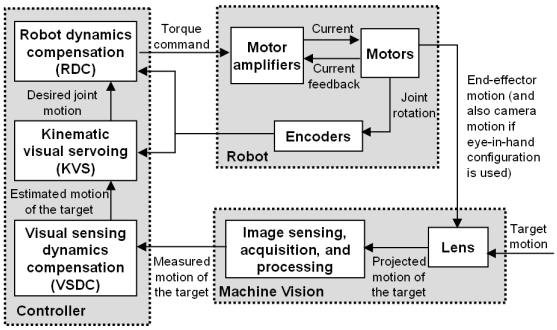


Fig. 3. A cascade control structure

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. A Cascade Structure via Multi-sliding Surface Control

In our previous work [2], a cascade control structure (Fig. 3) was proposed via multi-surface sliding control (MSC). As the basis of further discussion, the notations are briefly introduced in the following. A vision guided robot manipulator of eye-in-hand configuration is described by

$$\begin{cases} \boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \\ \mathbf{X}_{ee} = f_{rbt}(\mathbf{q}) \\ \mathbf{s}_{tgt} = f_{img}(\mathbf{X}_{ee}, \mathbf{X}_{tgt}) \end{cases} \tag{1}$$

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. $\mathbf{X}_{ee} = \left(\mathbf{p}_{ee}^T, \mathbf{r}_{ee}^T\right)^T$ 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).

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

$$\dot{\mathbf{S}}_1 = \dot{\mathbf{s}}_{tgt} = \mathbf{J}_{img}\mathbf{J}_{rbt}\dot{\mathbf{q}} + \frac{\partial \mathbf{s}_{tgt}}{\partial t} \tag{2}$$

where $\mathbf{J}_{rbt}$ is the robot Jacobian. $\mathbf{J}_{img} = \nabla_{\mathbf{x}_{ee}} f_{img}$ is called the 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 (2). 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.

$$\dot{\mathbf{q}}_d = -\mathbf{J}_{rbt}^{\dagger}\mathbf{J}_{img}^{\dagger}\left(\frac{\partial \mathbf{s}_{tgt}}{\partial t} + \mathbf{K}_{kvs}\,\mathrm{sign}(\mathbf{S}_1)\right) \tag{3}$$

where $\bullet_d$ means a desired value. $\bullet^{\dagger}$ means a generalized or augmented inverse [2]. 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

$$\begin{aligned} \dot{\mathbf{S}}_2 &= \ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d \\ &= \mathbf{M}^{-1}(\mathbf{q})\left(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{F}(\dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q})\right) - \ddot{\mathbf{q}}_d \end{aligned} \tag{4}$$

This time, 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.

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{invdy} + \boldsymbol{\tau}_{rbst} \tag{5}$$

where

$$\boldsymbol{\tau}_{rbst} = -\mathbf{M}(\mathbf{q})\,\mathbf{K}_{rdc}\,\mathrm{sign}(\mathbf{S}_2) \tag{6}$$

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

$$\boldsymbol{\tau}_{invdy} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{7}$$

gives a computed torque using the inverse dynamics of the robot. Equations (3) and (5) form a cascade controller. Equation (3) 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, and we call its task *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 [2], [8]. 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, (5) determines the torque to actuate the robot to follow the synthetic control given by (3). It can be considered as doing *robot dynamics compensation* (RDC). [9] mentioned about replacing the robustness term in (5) with a PID controller. Then, the RDC control law becomes almost the same to a conventional trajectory tracking controller. The only difference is that the computed torque part uses feedback instead of the desired values. This will satisfy many end users' preference of utilizing the trajectory tracking controller currently available in their robots.

B. Tailored Formulation for a 6-DOF Robot Manipulator

In [2], our experiments using a 2-DOF planar robot showed good result. Here, we extend the Jacobian-based KVS control law to a FANUC M-16*i*B 6-DOF robot manipulator (Fig. 4), and validate using a high-fidelity simulator. One concern in the formulation is about computing the

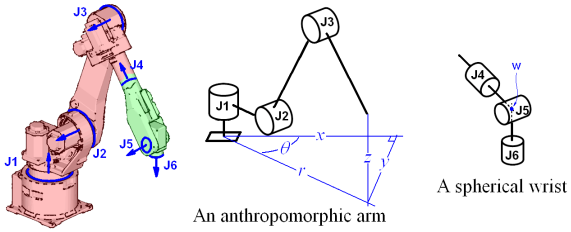


Fig. 4. A FANUC M-16*i*B robot and its kinematic feature

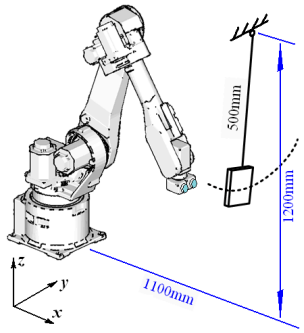


Fig. 5. Simulation setup of a 6-DOF manipulator and a swinging target

robot Jacobian in real-time. Usually the robot Jacobian can be computed through differential kinematics. It relates the rotation speed of n joints ($\dot{\mathbf{q}} \in \mathbb{R}^{\mathbf{n}}$) 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

$$\begin{bmatrix} \dot{\mathbf{p}}_{ee} \\ \dot{\mathbf{r}}_{ee} \end{bmatrix} = \mathbf{J}_{rbt}\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J}_{rbt,p} \\ \mathbf{J}_{rbt,r} \end{bmatrix} \dot{\mathbf{q}} \tag{8}$$

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 FANUC M-16*i*B robot, $n = 6$.

In the Jacobian-based KVS control law, the robot Jacobian needs to be evaluated in every control cycle. This brings much additional computation load. In addition, the singularity of 6-axis motion is complicated. When singularity occurs, special augmentation is needed to find a generalized inverse of the robot Jacobian. In [2], we proposed a generalized inverse using singular value decomposition. The method was shown to be effective on a 2-DOF planar robot, which has a very simple singularity scenario. The general stability of the method, however, can hardly be proved. Considering the complexity of the 6-axis motion, applying that method on a FANUC M-16*i*B robot is not preferable.

To solve the concern, we take the advantage of the specific kinematics structure of FANUC M-16*i*B robots. A FANUC M-16*i*B robot can be viewed as the combination of an *anthropomorphic arm* and a *spherical wrist* (Fig. 4). 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-16*i*B 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 ideas introduced above, we propose a revised version of the Jacobian-based KVS control law:

$$\dot{\mathbf{q}}_{d,123} = -\left[\mathbf{J}_{rbt,p}^{-1} \;\; \mathbf{0}_{3\times 3}\right] \mathbf{J}_{img}^{\dagger} \left(\frac{\partial \mathbf{s}_{tgt}}{\partial t} + \mathbf{K}_{kvs}\mathbf{s}_{tgt} \right) \tag{9}$$

where $\dot{\mathbf{q}}_{d,123}$ is the desired velocity of J1, J2, and J3. Meanwhile, for J4, J5, and J6, we do not use the Jacobian-based KVS control law. 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.

The control scheme is tested using a high-fidelity simulator of FANUC M-16*i*B robots. 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 [10], [11]. 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 (5).

The control task is to approach and track a damped swinging target in front of the robot. This is to mimic a task of picking a swinging workpiece. Fig. 5 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.

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

[2]Hereafter, we use $\mathbf{J}_{rbt,p}$ to represent a 3×3 matrix without the zeros.

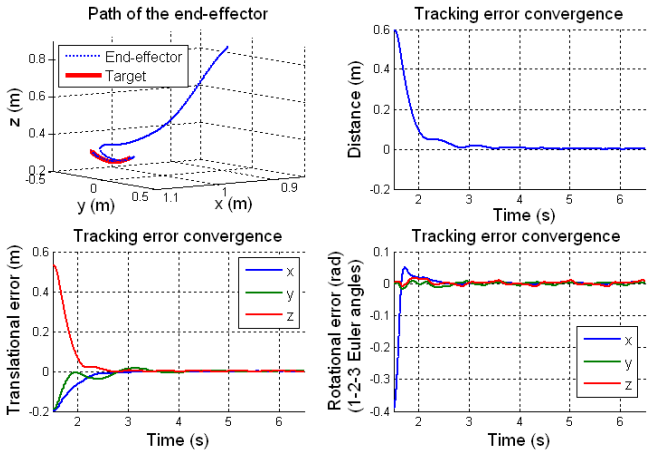


Fig. 6. Simulation result - applying a tailored Jacobian-based KVS control law on a 6-DOF robot manipulator

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 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 [2]. Fig. 6 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. In our case, the target is swinging in a y-z plane in the workspace of the robot (Fig. 5). A preferred relative approaching direction might be along x direction. This can be realized by adjusting the gain matrix $\mathbf{K}_{kvs}$ in the KVS control law. In the simulation shown in Fig. 6, $\mathbf{K}_{kvs}$ 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. 7(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. 7(b,c)). The risk of collision is attenuated.

III. ADDRESSING ACTUATOR LIMITS IN KVS

The Jacobian-based KVS control law does not handle the torque and speed limits of the actuators well. In (9), 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

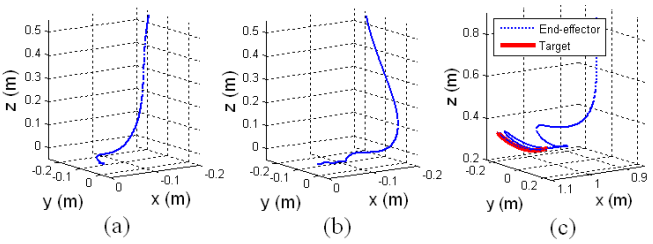


Fig. 7. Controlling the relative approaching direction - (a) The original relative path, (b) The improved relative path, (c) The improved path in the workspace

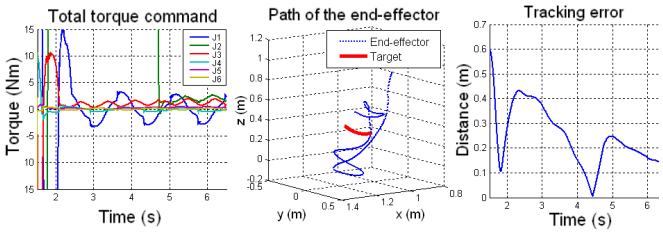


Fig. 8. Undesirable response caused by torque saturation

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. Fig. 8 shows an example when the KVS gain matrix is set blindly without careful tuning. The torque limits of the motors are $4Nm$, $4Nm$, $2Nm$, $2Nm$, $1Nm$, and $1Nm$ for J1 to J6 respectively.

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 [12], 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 [13] and [14], 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.

A. A Constrained Optimal Control Approach

In order to avoid the difficulty of choosing a final point for trajectory planning, we propose an alternative method by formulating the task into a constrained optimal control problem. As mentioned, each time when a new visual measurement becomes available, the VSDC algorithm introduced in [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

$$\min_{\ddot{\mathbf{q}}_d(i)} \lambda \mathbf{e}^T(kN+N)\,\mathbf{S}\,\mathbf{e}\,(kN+N) + \sum_{i=kN+1}^{kN+N} \mathbf{e}^T(i)\,\mathbf{S}\,\mathbf{e}\,(i) \qquad (10)$$

subject to

$$\begin{bmatrix}\mathbf{q}_d\,(i)\\ \dot{\mathbf{q}}_d\,(i)\end{bmatrix} = \begin{bmatrix}\mathbf{I} & T\mathbf{I}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{q}_d\,(i-1)\\ \dot{\mathbf{q}}_d\,(i-1)\end{bmatrix} + \begin{bmatrix}\frac{T^2}{2}\mathbf{I}\\ T\mathbf{I}\end{bmatrix}\ddot{\mathbf{q}}_d\,(i-1)$$
$$-\bar{\mathbf{v}} \le \dot{\mathbf{q}}_d\,(i) \le \bar{\mathbf{v}} \qquad (11)$$
$$-\bar{\boldsymbol{\tau}} \le \hat{\boldsymbol{\tau}}\,(i) \le \bar{\boldsymbol{\tau}}$$
$$(i = kN+1 \sim kN+N)$$

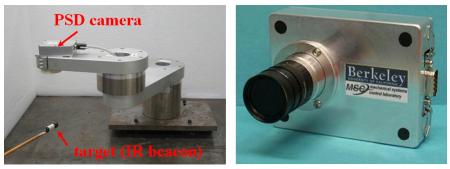
where the index i counts the basic control cycles. One vision sampling period is assumed to be N times of the basic control cycle. $\mathbf{I}$ is a 3×3 identity matrix. $\ddot{\mathbf{q}}_\mathbf{d}$ is the synthetic control (i.e., the reference motion) to be followed by the RDC control law. $\mathbf{e} = \begin{pmatrix}\mathbf{q}_d^T & \dot{\mathbf{q}}_d^T\end{pmatrix}^T - \begin{pmatrix}\mathbf{q}_{tgt}^T & \dot{\mathbf{q}}_{tgt}^T\end{pmatrix}^T$ 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. This enables us to simplify the problem by replacing $\hat{\boldsymbol{\tau}}$ in the constraints with a rough estimate

$$\tilde{\boldsymbol{\tau}}\,(i) = \mathbf{M}\,(kN)\,\ddot{\mathbf{q}}_d\,(i) + \mathbf{C}\,(kN)\,\dot{\mathbf{q}}_d\,(i) + \mathbf{G}\,(kN) \qquad (12)$$

This is an affine expression. It changes (10) 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 [15]. 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 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. The large analysis load cannot be implemented in our case, because the task needs to be done at the beginning of every vision sampling period.



(a) A 2-DOF planar robot (b) A PSD camera

Fig. 9. Experiment setup of a 2-DOF planar robot, a PSD camera, and an infrared point target

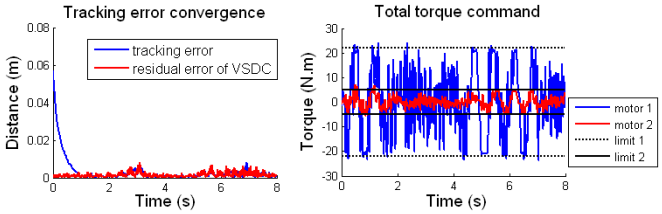


Fig. 10. Avoiding actuator saturation - experiment result using a 2-DOF planar robot

In order to obtain the robustness brought by feedback as well as keep a light computation load, we propose a revised batch solution. 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.,

$$\ddot{\mathbf{q}}_d\,(i) = \ddot{\mathbf{q}}_{d,b}\,(i) + K\begin{bmatrix}\mathbf{q}_{d,b}\,(i) - \mathbf{q}\,(i)\\ \dot{\mathbf{q}}_{d,b}\,(i) - \dot{\mathbf{q}}\,(i)\end{bmatrix} \qquad (13)$$

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

The proposed method is first tested on a 2-DOF planar robot (Fig. 9). The simple kinematics and dynamics, as well as the direct-drive actuators make the robot ideal for preliminary validation. A PSD camera is used to provide vision feedback [2], [16]. A manually moved infrared beacon is used as a point target. Fig. 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 (11) with an approximation. The excess, however, is minor. In general, the torque stays within the limits.

B. Tailored Formulation for a 6-DOF Robot Manipulator

Upon the promising result on a 2-DOF planar robot, we extend the method to a FANUC M-16*i*B 6-DOF robot manipulator and test using our high-fidelity simulator. Compared to a 2-DOF planar robot, a 6-DOF robot manipulator has much

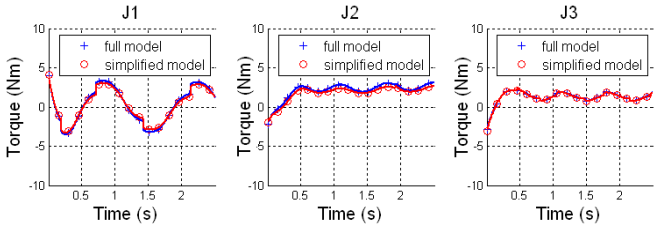


Fig. 11. Comparison between the full and simplified dynamic models

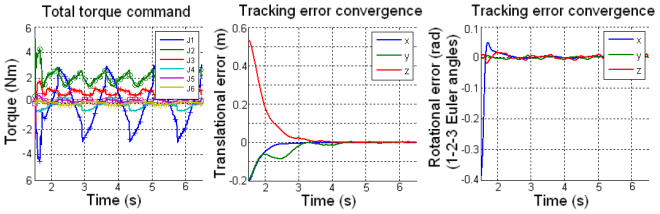


Fig. 12. Avoiding actuator saturation - simulation result of a 6-DOF robot manipulator

more complex dynamics. 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 (12) requires considerable computation load and can hardly be done in real-time at the desired $1kHz$ rate. In order to reduce computation load, we use an approximation of the full Lagrangian dynamic model. The approximation assumes that the motion of J4, J5, and J6 contributes little to the torque of J1, J2, and J3, 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. To justify this assumption, we study the torque profile along the trajectory generated by the Jacobian-based KVS control law. Fig.11 shows the comparison between the torques computed using the full and simplified Lagrangian dynamic models.

Using the simplified Lagrangian dynamic model, the constrained optimal KVS approach is tested using our high-fidelity simulator. Again, the torque limits of the motors are $4Nm$, $4Nm$, $2Nm$, $2Nm$, $1Nm$, and $1Nm$ for J1 to J6. Fig. 12 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.

IV. CONCLUSIONS

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. Continued from our previous work on visual sensing dynamics compensation, this paper discussed the consideration of robot dynamics in controller design. 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 first tested experimentally using a simple 2-DOF planar robot. After special tailoring, extended validation was conducted using a high-fidelity simulator of a 6-DOF industrial manipulator.

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. This makes experiments on an actual 6-DOF industrial manipulator less reliable. Unlike a 2-DOF planar robot, 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. Our next work will focus on dealing with the FOV limit as well as the preferred relative approaching direction.

REFERENCES

[1] P. Corke and M. Good, "Dynamic effects in visual closed-loop systems," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 671–683, October 1996.

[2] 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*, pp. 45–52, April 2013.

[3] C.-Y. Lin, C. Wang, and M. Tomizuka, "Visual tracking with sensing dynamics compensation using the Expectation-Maximization algorithm," in *Proceedings of the 2013 American Control Conference*, pp. 6281–6286, June 2013.

[4] F. Chaumette and S. Hutchinson, "Visual servo control I - basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, pp. 82–90, December 2006.

[5] F. Chaumette and S. Hutchinson, "Visual servo control II - advanced approaches," *IEEE Robotics Automation Magazine*, vol. 14, pp. 109–118, March 2007.

[6] P. Corke, *Robotics, vision and control*. Springer-Verlag, Berlin, 2011.

[7] F. Janabi-Sharifi, L. Deng, and W. Wilson, "Comparison of basic visual servoing methods," *IEEE/ASME Transactions on Mechatronics*, vol. 16, pp. 967–983, October 2011.

[8] D. Swaroop, J. Hedrick, P. Yip, and J. Gerdes, "Dynamic surface control for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 45, pp. 1893–1899, 2000.

[9] C. Wang, C.-Y. Lin, and M. Tomizuka, "Visual servoing for robot manipulators considering sensing and dynamics limitations," in *Proceedings of the 2013 ASME Dynamic Systems and Control Conference*, October 2013.

[10] 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*, pp. 371–377, 2013.

[11] W. Chen and M. Tomizuka, "Direct joint space state estimation in robots with multiple elastic joints," *IEEE/ASME Transactions on Mechatronics*, vol. PP, no. 99, pp. 1–10, 2013.

[12] 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*, pp. 1349–1354, June 2010.

[13] 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*, pp. 4862–4869, May 2012.

[14] T. Kroger and F. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, pp. 94–111, February 2010.

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

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