motion for the system. Key to the method is the use of reaction wheels to control the orientation of the base and the elimination of the base linear motion from the equations of motion. It was shown that the effect of the reaction wheels on the dynamics of the system can be divided into two components. The first was the component related to the generalized momentum of the reaction wheels. The remaining component can be effectively included in the dynamics by modifying the inertial properties of the base. The linear motion of the base was removed from the equations of motion by using the law of conservation of linear momentum. With these two transformations, the resulting form of the kinetic energy was easily utilized in Lagrange's equation to obtain the dynamic equations of motion.

The passivity approach was adopted in the development of the adaptive controller. The method of stability analysis presented by Ortega and Spong was utilized [13]. The significant difference is in the utilization of the recursive form of the dynamic equations of motion in the statement of the stability theorem.

The form of the dynamic equations used in the stability proof directly leads to a computationally efficient recursive implementation of the control law and parameter update law. However, the computational load is still fairly high and increases quadratically with the number of links in the system. Due to the coupling of the dynamics, there does not seem to be any way of avoiding the quadratic complexity problem. Since it is known that $U_{jk} = U_{kj}^T$, some computational improvements could be made with a slight modification of the adaptation law such that only U_{jk} is estimated instead the current method of estimating both U_{jk} and U_{kj}. However, this does not appear likely to produce significant improvements. A more promising approach might be to avoid all of the coordinate transformations by referring the velocities, accelerations, and forces to their own link coordinates. For example, one would compute $A_k^{-1}\dot{A}_k$ instead of $\dot{A}_k$ and $A_j^{-1}F_{jk}(A_k^{-1})^T$ instead of F_{jk}.

A simulation was presented to demonstrate the effectiveness of the controller. A quadratic polynomial was used to generate the desired trajectory to illustrate the trajectory tracking capability of the controller.

A significant extension of these results would be the solution for manipulators containing closed kinematic loops, since these are the most common types of manipulators encountered in practice. This would also lead to methods for dual-arm coordinated motion control and compliant motion control.

References

[1] R. W. Longman, R. E. Lindberg, and M. F. Zedd, "Satellite-mounted robot manipulators—New kinematics and reaction moment compensation," *Int. J. Robotics Res.*, vol. 6, no. 3, pp. 87-103, 1987.

[2] Z. Vafa and S. Dubowsky, "On the dynamics of manipulators in space using the virtual manipulator approach," in *Proc. IEEE Int. Conf. Robotics Automat.* (Raleigh, NC), 1987.

[3] Y. Nakamura and R. Mukherjee, "Nonholonomic path planning of space robots," in *Proc. IEEE Int. Conf. Robotics Automat.* (Scottsdale, AZ), 1989, pp. 1050-1055.

[4] Y. Nakamura and R. Mukherjee, "Nonholonomic path planning of space robots via bidirectional approach," in *Proc. IEEE Int. Conf. Robotics Automat.* (Cincinnati, OH), 1990, pp. 1764-1769.

[5] W. W. Hooker and G. Margulies, "The dynamical attitude equations for an n-body satellite," *J. Astron. Sci.*, vol. XII, pp. 123-128, Winter 1965.

[6] W. W. Hooker, "A set of *r* dynamical attitude equations for an arbitrary *n*-body satellite having *r* rotational degrees of freedom," *AIAA J.*, vol. 8, no. 7, pp. 1205-1207, July 1970.

[7] J. R. Wertz, *Spacecraft Attitude Determination and Control.* Boston: D. Reidel, 1978.

[8] E. Papadopoulos and S. Dubowsky, "On the nature of control algorithms for space manipulators," in *Proc. IEEE Int. Conf. Robotics Automat.* (Cincinnati, OH), 1990, pp. 1102-1108.

[9] Y. Umetani and K. Yoshida, "Experimental study on two-dimensional free-flying robot satellite model," in *Proc. NASA Conf. Space Telerobot.*, vol. 5, 1989, pp. 215-224.

[10] R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control.* Cambridge, MA: MIT Press, 1981.

[11] K. Fu, R. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence.* New York: McGraw-Hill, 1987.

[12] R. Featherstone, *Robot Dynamics Algorithms.* Norwell, MA: Kluwer Academic, 1987.

[13] R. Ortega and M. W. Spong, "Adaptive motion control of rigid robots: A tutorial," in *Proc. 27th Conf. Decision Contr.* (Austin, TX), Dec. 1988, pp. 1575-1587.

[14] B. Brogliato, I. Landau, and R. Lozano-Leal, "Adaptive motion control of robot manipulators: A unified approach based on passivity," in *Proc. Amer. Control Conf.* (San Diego, CA), May 1990, pp. 2259-2264.

[15] J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *Int. J. Robotics Res.*, vol. 6, no. 3, 1987.

An Iterative Learning Control of Robot Manipulators

Tae-yong Kuc, Kwanghee Nam, and Jin S. Lee

Abstract—**An iterative learning scheme comprising a unique feedforward learning controller and a linear feedback controller is presented. In the feedback loop, the fixed-gain PD controller provides a stable open neighborhood along a desired trajectory. In the feedforward path, on the other hand, a learning control strategy is exploited to predict the desired actuator torques. It is shown that the predicted actuator torque converges to the desired one as the iteration number increases. The convergence is established based on the Lyapunov stability theory. The proposed learning scheme is structurally simple and computationaly efficient. Moreover, it posesses two major advantages: the ability to reject unknown deterministic disturbances and the ability to adapt itself to the unknown system parameters.**

I. Introduction

Although conventional fixed-gain linear feedback controllers are prevalent in current commercial robot systems, their performance is limited in controlling robot motion. Of numerous control schemes, the computed torque method is well known for its nonlinearity compensating scheme [6]. However, in practice, without a precise description of gear backlash, friction, etc., the computed torque method results in no better performance than that of a conventional fixed-gain controller. Also, the computed torque method does not seem to be flexible with respect to varying environmental conditions, such as payload changes, because it requires *a priori* knowledge of the dynamic model. A number of adaptive control techniques to develop controllers that can accommodate varying environments and are not sensitive to modeling errors have been reported [11], [16], [20]. A general drawback of the adaptive ap-

Manuscript received March 9, 1990; revised January 30, 1991. This work was supported in part by KOSEF under Grant 901-0803-013-2. Portions of this work were presented at the IEEE CDC, Tampa, FL, December 1989.

The authors are with the Department of Electrical Engineering, Pohang Institute of Science and Technology, Pohang, 790-600, South Korea.

IEEE Log Number 9103158.

proach may be the large computational load required for real-time parameter identification.

More recently, increasing attention has been paid to the application of iterative learning techniques [3], [8], [9] to robot controller. With the learning control techniques, it is possible to make a robot track an arbitrary desired trajectory within the work space. In addition, learning schemes are flexible and easy to implement because they do not require an exact dynamic model. Considerable work has been reported concerning the iterative learning scheme. Bondi *et al.* [7] used the high-gain feedback concept to limit the trajectory error within a uniform bound at each iteration of learning. Miller *et al.* [17] proposed a feasible control scheme utilizing the Cerebellar Model Articulation Controller (CMAC) which was developed by Albus [1]. Kawato *et al.* [14] and Miyamoto *et al.* [18] adopted a single-layer perceptron concept in computing the feedforward torque components for a robot manipulator. They chose system-dependent functions for predicates (called subsystems) that are partial cues to the linear or nonlinear system structure of the robot model. However, the selection of the predicates in this control scheme relies on knowledge of the robot model structure. This fact makes the controller dependent on the system structure, and thus this control scheme appears inconvenient when the system has an unknown or uncertain structure.

In this short paper, we propose a learning controller for robot manipulators that comprises a unique feedforward learning controller and a linear feedback controller. The overall control system structure may look similar to the other learning control schemes, since it utilizes both a feedforward compensator and a fixed-gain linear feedback controller. However, the difference lies in the learning scheme. Specifically, in the model given in [14] and [18], the update mechanism is essentially a kind of parameter adaptive scheme. In our learning scheme, the feedforward actuator torques are computed iterationwise for each step by adding up the feedback error terms.

Unlike the Bondi and Miller schemes [7], [17], our learning scheme does not requires acceleration terms. Also, our learning scheme has the capability of rejecting unknown disturbances and adapting itself to time-varying system parameters. The proof of convergence as well as the disturbance rejection is made based on the Lyapunov stability theory. The whole analysis utilizes a linearized model of a robot system along a desired trajectory.

II. The Learning Control Structure

Generally, the dynamics of robot manipulators with N degrees of freedom are described by

$$D(q(t))\ddot{q}(t) + B(q(t), \dot{q}(t)) + G(q(t)) + T_a(t) = T(t) \tag{1}$$

where $q(t) \in R^N$ denotes the generalized joint variables. Note that $D(q(t))$, $B(q(t), \dot{q}(t))$, and $G(q(t))$ are the inertia matrix, Coriolis plus centripetal forces, and the gravity force vector, respectively. $T_a(t) \in R^N$ describes disturbances due to unmodeled dynamics or unknown disturbances, and $T(t) \in R^N$ is the vector of input torques applied to each joint.

We assume that a given desired joint trajectory $q_d(t)$ belongs to $C^2[0, t_f]$, where $C^2[0, t_f]$ is the set of twice continuously differentiable functions on $[0, t_f]$. Our control objective here is to make the robot manipulator track a given desired trajectory, $q_d(t)$ for all $t \in [0, t_f]$. However, this objective is in general very difficult to achieve by conventional control methods due to the inherent nonlinearity and the unmodeled dynamics. The situation becomes even worse when the unknown system parameters have time-varying effects such as payload changes, mechanical wear, aging, etc. This type of system change, as well as the disturbance $T_a(t)$, is extremely difficult to manage. As a way to overcome this difficulty, we propose an iterative learning control technique that searches for a desired input torque through a sequence of repetitive operations. That is, by training the system through the repetitive operations, we are aiming at reducing trajectory error as the iteration number increases.

It is assumed that the disturbance $T_a(t)$ is repetitive for each iteration. $(q_d(t), \dot{q}_d(t), \ddot{q}_d(t))$ and $(q(t), \dot{q}(t), \ddot{q}(t))$ denote the joint position, velocity, and acceleration of desired and actual trajectories, respectively. The superscript i of each variable indicates its value at the ith iteration. Linearizing the system (1) along $q_d(t)$ for $t \in [0, t_f]$ at the ith iteration, we obtain the following linear time-varying system:

$$C_d(\ddot{q}^i(t) - \ddot{q}_d(t)) + E_d(\dot{q}^i(t) - \dot{q}_d(t)) + F_d(q^i(t) - q_d(t)) = T^i - S_d \tag{2}$$

where

$$C_d \equiv D(q_d(t))$$

$$E_d \equiv \left.\frac{\partial B}{\partial \dot{q}(t)}\right|_{(q_d(t), \dot{q}_d(t))}$$

$$F_d \equiv \left.\frac{\partial D}{\partial q(t)}\right|_{q_d(t)} \ddot{q}_d(t) + \left.\frac{\partial B}{\partial q(t)}\right|_{(q_d(t), \dot{q}_d(t))} + \left.\frac{\partial G}{\partial q(t)}\right|_{(q_d(t), \dot{q}_d(t))}$$

$$S_d \equiv D(q_d(t))\ddot{q}_d(t) + B(q_d(t), \dot{q}_d(t)) + G(q_d(t)) + T_a(t). \tag{3}$$

T^i is the input torque at the ith iteration.

To make the robot system track $q_d(t)$ for $t \in [0, t_f]$, we construct a controller as follows:

$$T^i = T_e^i + H^i \tag{4}$$

where $T_e^i = K(q_d - q^i) + L(\dot{q}_d - \dot{q}^i)$. In T_e^i, the matrices K and L are positive definite, which account for position and velocity feedback gains. Thus, T_e^i represents a conventional PD controller. H^i is the predicted feedforward control input to be computed at each iteration by a learning rule. Applying the input (4) to system (2), we obtain an error equation

$$C_d\ddot{e}^i(t) + (E_d + L)\dot{e}^i(t) + (F_d + K)e^i(t) = S_d - H^i \tag{5}$$

where $e^i(t) = q_d(t) - q^i(t)$. The error boundedness of this type of feedback controller has been established in several treatments [4], [12], [21]. As long as the term $S_d - H^i$ driving the error dynamics is finite, the size of the error bound can be made arbitrarily small with an increase in feedback gain. In practice, however, we cannot increase the feedback gain arbitrarily large because actuator torques are limited. This fact implies that, in general, the linear feedback control is not adequate for tracking especially when the system has modeling errors or nonlinearity. Hence, we intend to use the feedforward control input H^i along with the linear feedback controller with the objective of eliminating $S_d - H^i$ as $i \to \infty$. Then, even with reasonable feedback gains, the tracking error may converge to zero as the iteration proceeds. Based on this motivation, our control strategy is as follows: First, we choose positive definite matrices K and L appropriately large so that the error dynamics (5) along the desired trajectory are stable. Second, we update $H^i(t)$ with a learning rule so that $H^i(t)$ converges to $S_d(t)$ for all $t \in [0, t_f]$.

The control structure implementing the above control strategy is depicted in Fig. 1. The trajectory planner generates the desired trajectories, which are used as a reference input for the learning controller. The fixed-gain PD controller makes the overall system stable within uniform error bounds, whereas the feedforward controller updates the feedforward torque components H^i's. An associative memory structure can be used to implement the feedforward controller with a limited memory size. The structure of the associative memory is presented in [15], and more details will be reported later.

At the initial stage of learning, the H^i's are set to zero. But, the torque command $T_e^i(t)$ from the PD controller is significant because there are significant position and velocity errors. Hence, in the early stage of learning, the feedback torque component T_e^i is dominant over the feedforward torque component H^i. However, as the number of iterations increases, H^i becomes dominant over T_e^i.

III. The Learning Rule and Its Convergence

In this section, we derive a learning rule that updates H^i so that H^i converges to an unknown value S_d, and we prove its convergence. In deriving a learning rule we consider the following index for $t \in [0, t_f]$:

$$J_t = \frac{1}{2} \sum_{r=1}^{\infty} \| S_d(t) - H^r(t) \|^2. \tag{6}$$

Applying the gradient descent rule, we obtain

$$H^{i+1} = H^i - \beta \frac{\partial J_t}{\partial H^i} = H^i + \beta (S_d - H^i) \tag{7}$$

where β is a positive constant often called a *training factor*. β should be such that $0 < \beta < 2$ to guarantee the convergence of H^i (see Appendix A). Since S_d is not known, (7) cannot be used directly as a learning rule. Instead, replacing the unknown term $S_d - H^i$ by the available quantity T_e^i in (4), we obtain the following learning rule:

$$H^{i+1} = H^i + \beta T_e^i. \tag{8}$$

Note from (5) that if K and L are sufficiently large, then T_e^i is about the same as $S_d - H^i$. Roughly speaking, the learning rule may be considered as a searching algorithm for the unknown desired input torque S_d in which the error T_e^i from the PD controller is used to update H^i. We assume that time t is reset to zero at the starting point of each iteration. We also assume that $q^i(0) = q_d(0)$ and $\dot{q}^i(0) = \dot{q}_d(0)$ for all $i \geq 1$. In practice, we usually choose the training factor β to be less than unity due to sensitivity considerations and choose $H^1(t) = 0$ for all $t \in [0, t_f]$.

To simplify the proof of stability, we let $K = aL$ for a positive constant a. For $x \in R^n$ and $A \in R^{n \times n}$, we denote by $\|x\|$ the Euclidean norm and by $\|A\|$ its induced norm. In the case when each element of A is a function of t, we let $\|A\|_m = \max_{0 \leq t \leq t_f} \{\|A(t)\|\}$. For a symmetric matrix $M \in R^{n \times n}$, we denote by $\lambda_{\min}(M)$ the minimum eigenvalue of M. In the proof of convergence, we select a performance index $V^i(t)$ as follows:

$$V^i(t) \equiv \int_0^t (z^i(\tau))^T \Lambda (z^i(\tau))\, d\tau \quad \text{for} \quad t \in [0, t_f]$$

where $z^i(t) = \dot{e}^i() + ae^i(t)$ and $\Lambda = \beta L = \beta L^T$. Then, since $z^i(t)$ may be considered as an input to a strictly proper stable filter, convergence of $z^i(t)$ will guarantee that of $e^i(t)$.

Theorem: With the control input (4) and learning rule (8), the following hold for all $t \in [0, t_f]$:

i) $V^{i+1}(t) \leq V^i(t)$

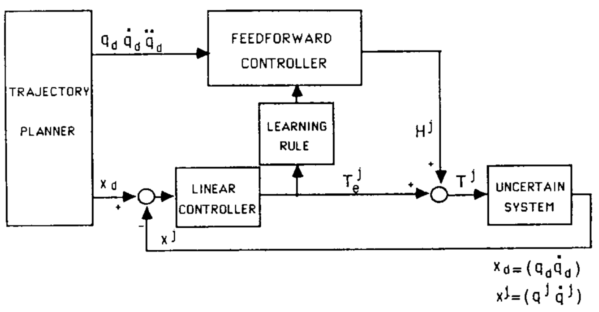


Fig. 1. Schematic diagram of the learning control scheme.

ii) $\lim_{i \to \infty} e^i(t) = 0$

iii) $\lim_{i \to \infty} H^i = S_d$

if the controller gains K and L are selected such that

$$l_a \equiv \lambda_{\min}((2 - \beta)L) - (2a\|C_d\|_m + \|\dot{C}_d\|_m) > 0$$

$$l_b \equiv \lambda_{\min}((2 - \beta)L) - \frac{2}{a}\|F_d\|_m > 0$$

$$\sqrt{l_a l_b} > \frac{1}{a}\|F_d\|_m.$$

Proof: See Appendix B.

The conditions imposed on the theorem imply that the magnitude of feedback gains depends on the desired configuration parameters C_d, $\dot{C}_d$, and F_d. In general, if we want to make a nonlinear system track a desired path without tracking error and with only a linear feedback controller, the feedback gains should be infinite. However, if there is a nonlinear compensating controller in the feedforward path, feedback gains need not be arbitrarily large. In our control scheme, the exactly same methodology is implemented. If the nonlinear function $S_d(t)$ in (5) is not canceled out by $H^i(t)$, the trajectory error $e^i(t)$ does not vanish. The learning rule in our feedforward controller estimates the unknown nonlinear function $S_d(t)$ and compensates it by generating a compensating torque $H^i(t)$. Since the feedforward controller is taking care of nonlinearity compensation by the learning rule of (8), there is no need to make the feedback gains K, L arbitrarily large in order to neglect the effects of $S_d(t)$. However, to guarantee the stability of the homogeneous part of the error dynamics (5), K and L should have large values.

IV. A Simulation Example and Results

The learning control scheme described in this work is applied to control the joint angles of a robot manipulator. As a simulation example, we consider the motion control of the Unimation PUMA 560 arm. The coordinate frames for the PUMA 560 arm are attatched on each joint using a modified Denavit–Hartenberg method [10] in the following schematic diagram. Their parameters are shown in Table I [5].

Dynamic simulations are performed for the first three links of the PUMA 560 arm with link masses $m_2 = 17.40$ kg, $m_3 = 4.8$ kg and the payload $m_p = 5.0$ kg. Here, we limit the maximum torques to 97.6, 186.4, and 89.4 Nm for links 1, 2, and 3 [5].

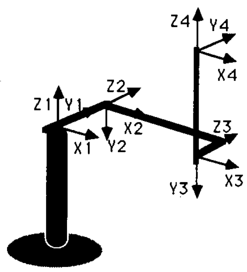


Fig. 2. Kinematic structure of the PUMA 560 arm.

TABLE I

Modified Denavit-Hartenberg Parameters

i	θ_i	$d_i(m)$	$a_{i-1}(m)$	$\alpha_{i-1}(^\circ)$
1	q_1	0	0	0
2	q_2	0.2435	0	−90
3	q_3	−0.0934	0.4318	0
4	q_4	0.4331	−0.0203	90
5	q_5	0	0	−90
6	q_6	0	0	90

For the desired joint trajectories, we choose for $0 \le t \le 6$

$$\begin{bmatrix} q_{d1}(t) \\ q_{d2}(t) \\ q_{d3}(t) \end{bmatrix} = \begin{bmatrix} (\pi/3)\cos(4\pi/3)t \\ -(\pi/3)\cos(4\pi/3)t - \pi/4 \\ (\pi/2)\cos(4\pi/3)t \end{bmatrix}$$

$$\begin{bmatrix} \dot{q}_{d1}(t) \\ \dot{q}_{d2}(t) \\ \dot{q}_{d3}(t) \end{bmatrix} = \begin{bmatrix} -(4\pi^2/9)\sin(4\pi/3)t \\ (4\pi^2/9)\sin(4\pi/3)t \\ -(2\pi^2/3)\sin(4\pi/3)t \end{bmatrix}.$$

Further, we choose $\beta_i = 0.6 (i = 1, 2, 3)$ for each joint and 0.005 s for the sampling time interval, and the following for the gain matrices:

$$K = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

$$L = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 70 & 0 \\ 0 & 0 & 30 \end{bmatrix}.$$

The actual trajectories were obtained by solving differential equations of the robot motion [5] using the Runge-Kutta fourth method. We considered the cases where the payload m_p of link 3 is 0 and where $m_p = 5$ kg. Fig. 3(a) shows the actual and desired trajectories of the system with no load and without the feedforward learning controller. Fig. 3(b) shows that the trajectories of the same system with the learning controller converge to the desired ones as the learning proceeds. After 30 iterations, the rms position and velocity errors reduce to less than 1% of the initial errors, i.e., errors of the system without learning. In Fig. 4, $\sum_{k=1}^{n}\{H^i(k)\}^2$ and $\sum_{k=1}^{n}\{T_e^i(k)\}^2$ for joints 1, 2, and 3 are plotted versus the iteration number. The figure shows the role exchange in the controller: at the initial stage of learning, the torques T_e^i are much larger than the feedforward torque component H^i, whereas T_e^i decreases and H^i increases as learning proceeds. Further, the torques H^i converge to the torques required for tracking the desired trajectory. We have applied a payload of 5 kg after the fiftieth iteration without load. Fig. 5(a) shows the actual trajectories of the system without the learning controller. One can see that the tracking errors increase when the payload is inserted. However, with the learning controller, the trajectory errors becomes trivial even under the payload changes in Fig. 5(b). It is shown in Fig. 6 that the trajectory errors caused by the payload variation become negligible after 10 or more iterations.

V. Discussion

In our learning control scheme, a linear controller is fully exploited in attracting the state of the system to the stable region, while the learning rule makes corrections at each iteration to follow a desired trajectory. Our scheme utilizes a classical linear PD controller to keep the tracking error within a domain of attraction and a feedforward learning controller to compensate for the nonlinear characteristics of the robot manipulator. Therefore, two types of controllers, a linear PD controller and a feedforward learning control strategy, cooperate since the linear controller provides stability while the feedforward controller produces compensating torques that are updated over a sequence of iterations. In the early stages of iteration, the linear feedback controller assumes the predominant role for achieving stability. But as learning proceeds, the torque computed from the feedforward controller converges to the actuator torques required to track the desired trajectory. Hence, at the final stage of learning, the robot manipulator is operated predominantly by the feedforward learning controller in computing the desired torque values.

Due to the simplicity of the learning rule given here, the computational burden does not present a problem in real-time applications. When the robot motion speed increases, the gains need to be increased to maintain system stability. The simulation results of the PUMA 560 arm, however, show that the transient effects caused by actuator saturation are not so severe and disappear quickly as learning proceeds.

VI. Conclusion

A learning control technique is presented in this paper for the control of robot manipulators. The learning controller is structurally simple, computationaly efficient, and posesses excellent disturbance rejection properties. The convergence proof is given based on the Lyapunov stability theory. Neither a teaching signal nor a manipulator acceleration measurement/estimation is used in the learning controller, which makes the controller flexible and robust under varying and noisy environments. The feasibility of the control scheme is demonstrated through a computer simulation. In our opinion, this is one of the simplest learning controllers reported in the literature.

Appendix A

Subtracting both sides of (7) by S_d, we obtain

$$S_d - H^{i+1} = (S_d - H^i) - \beta(S_d - H^i) = (1 - \beta)(S_d - H^i).$$

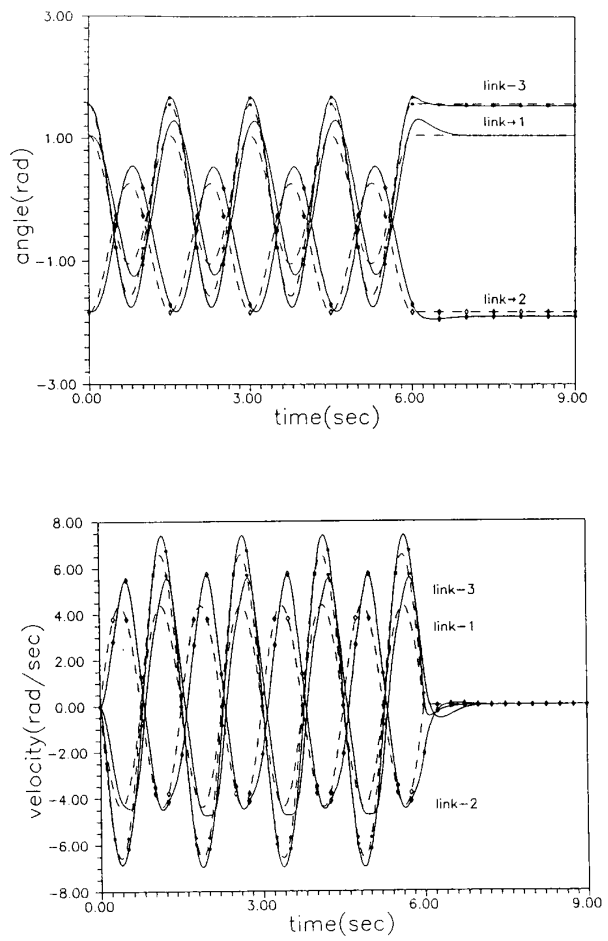
Therefore, to guarantee the convergence of the bias input error with respect to iteration, β should be in the range of $0 < \beta < 2$.

Appendix B

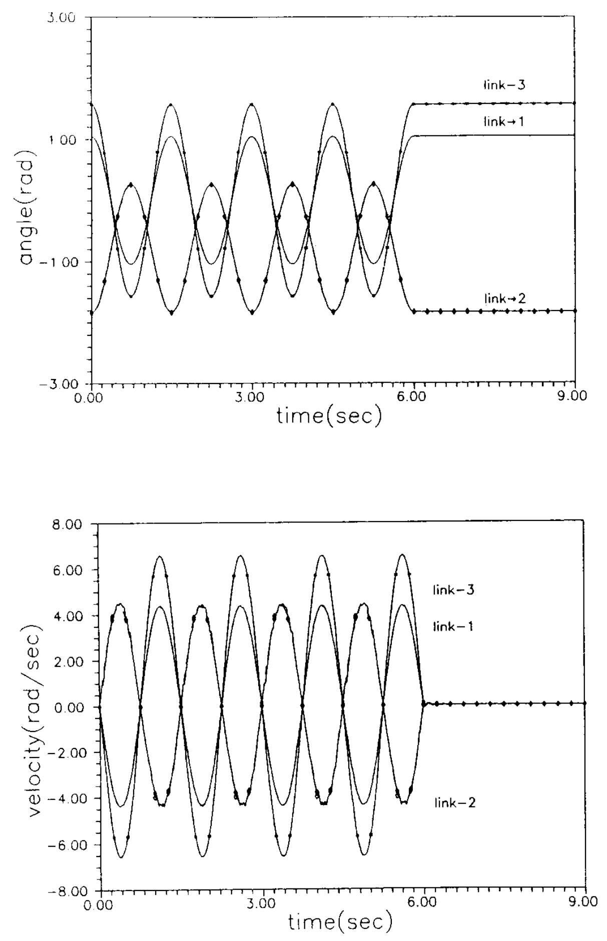
Proof of the Theorem

Let $\delta z^i \equiv z^{i+1} - z^i$ and $\delta e^i \equiv e^{i+1} - e^i$. Then from (5) and (8) we obtain

$$C_d \delta \dot{z}^i(t) + (E_d + L - aC_d)\delta z^i(t) + (a^2 C_d - aE_d + F_d)\delta e^i(t) = -\beta L z^i(t). \quad \text{(B1)}$$



(a)



(b)

Fig. 3. Joint position and velocity trajectories of the system (a) without learning and (b) after 30 iterations. Dotted line: desired trajectory; solid line: actual trajectory.

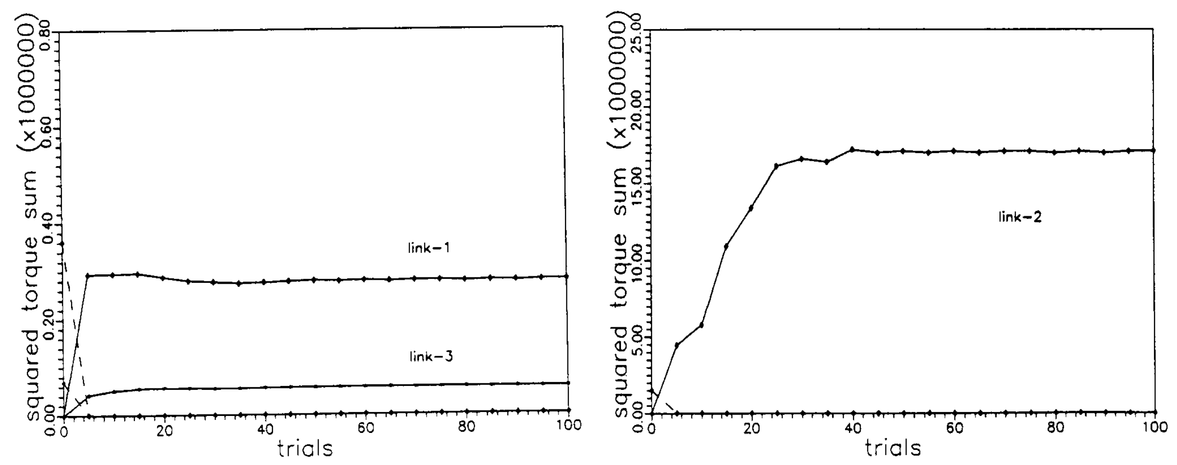


Fig. 4. Plots of squared torque sums versus iteration number. Solid line: $\sum_k \{H^i(k)\}^2$; dashed line: $\sum_k \{T_e^i(k)\}^2$.

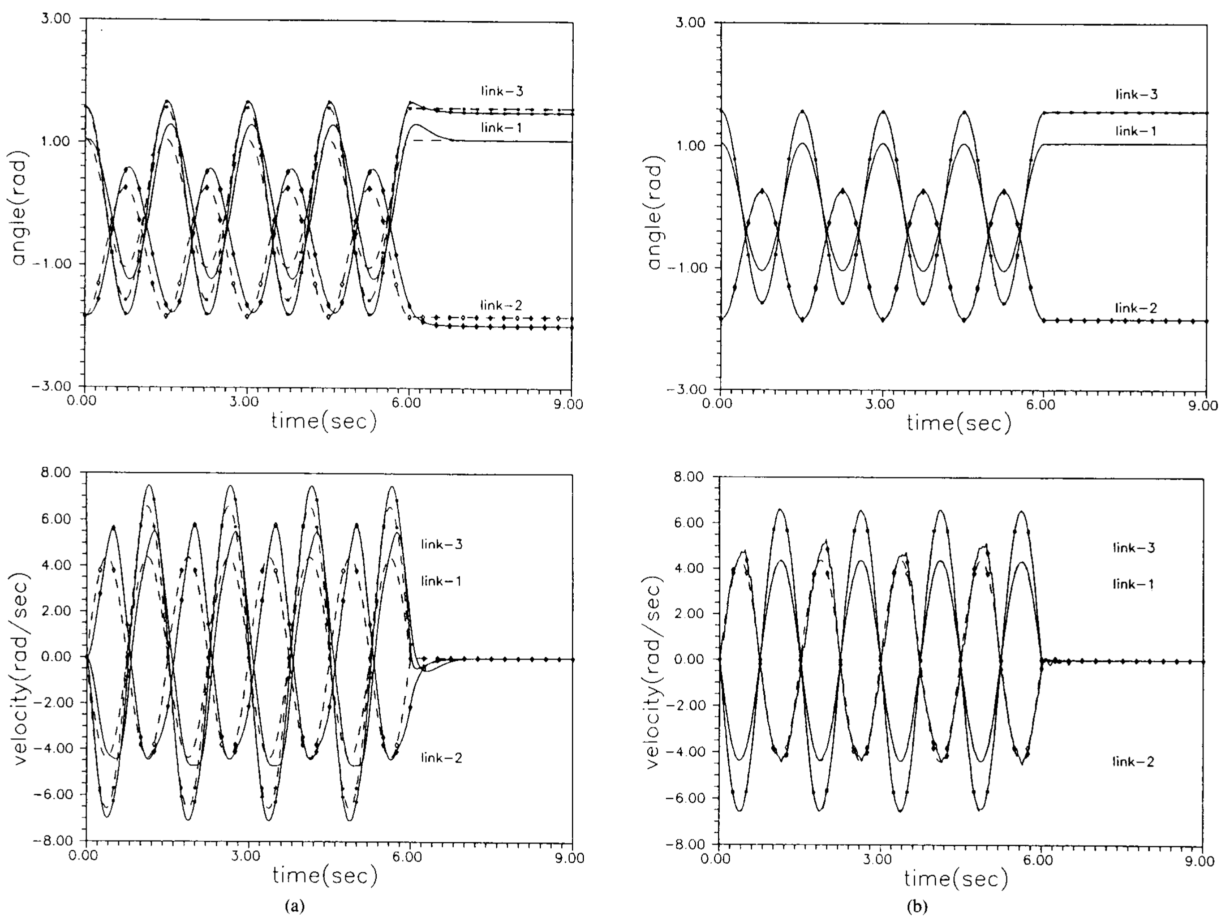


Fig. 5. Joint position and velocity trajectories of the system with $m_p = 5$ kg (a) without learning and (b) attached after the fiftieth iteration. Dotted line: desired trajectory; solid line: actual trajectory.

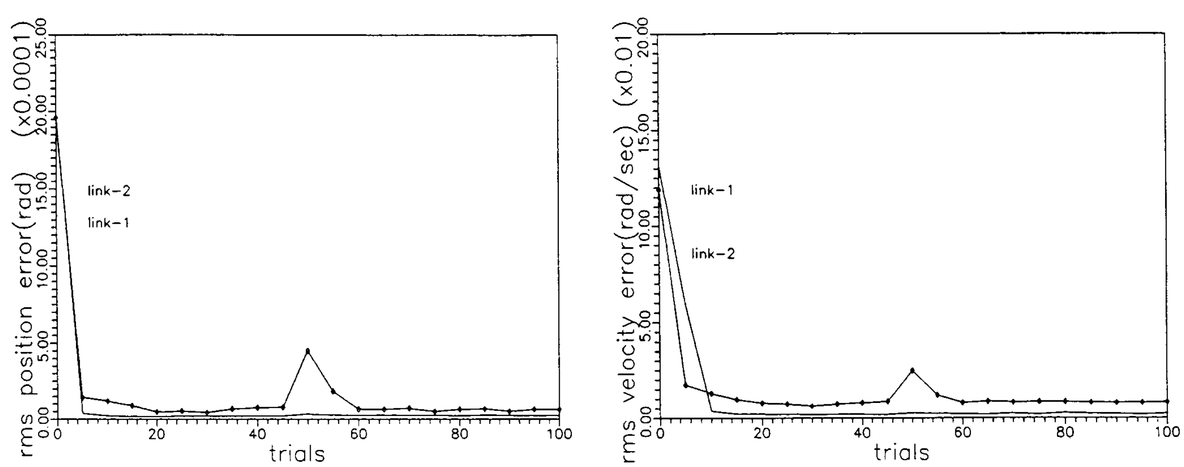


Fig. 6. Profiles of rms position and velocity errors with payload variation after the fiftieth iteration.

From the definition V^i, we obtain

$$V^{i+1}(t) = \int_0^t (z^{i+1}(\tau))^T \Lambda (z^{i+1}(\tau))\, d\tau = V^i(t)$$
$$+ \int_0^t (\delta z^{i^T}(\tau)\Lambda \delta z^i(\tau) + 2\delta z^{i^T}(\tau)\Lambda z^i(\tau))\, d\tau.$$

Let $\Delta V^i(t) \equiv V^{i+1}(t) - V^i(t)$.

Substituting z^i and applying integration by parts yields

$$\Delta V^i = \int_0^t \delta z^{i^T}\Lambda \delta z^i \, d\tau + 2\int_0^t \delta z^{i^T}\Lambda z^i\, d\tau = \int_0^t \delta z^{i^T}\Lambda\delta z^i\, d\tau$$
$$-2\int_0^t \delta z^{i^T}(C_d \delta \dot{z}^i + (E_d + L - aC_d)\delta z^i$$
$$+ (a^2 C_d - aE_d + F_d)\delta e^i)\, d\tau = -\delta z^{i^T}(t) C_d \delta z^i(t)$$
$$-\int_0^t \delta z^{i^T}((2-\beta)L + E_d - 2aC_d)\delta z^i\, d\tau$$
$$-2\int_0^T \delta z^{i^T}(a^2 C_d - aE_d + F_d)\delta e^i\, d\tau$$

where we have utilized the initial condition $\delta z^i(0) = 0$ and the fact that $\dot{C}_d - E_d$ is a skew-symmetric matrix. Substituting $\delta\dot{e}^i + a\delta e^i$ for δz^i and applying integration by parts again yields

$$\Delta V^i = -\delta z^{i^T}(t) C_d \delta z^i(t) - a\delta e^{i^T}(t) L_c \delta e^i(t)$$
$$-\int_0^t \delta \dot{e}^{i^T}((2-\beta)L - 2aC_d + \dot{C}_d)\delta\dot{e}^i\, d\tau$$
$$-2a\int_0^t \delta\dot{e}^{i^T}\frac{1}{a}F_d \delta e^i\, d\tau$$
$$-a^2\int_0^t \delta e^{i^T}\left((2-\beta)L + \frac{2}{a}F_d\right)\delta e^i\, d\tau$$
$$= -\delta z^{i^T}(t)C_d\delta z^i(t) - a\delta e^{i^T}(t)L_c\delta e^i(t) - \int_0^t W(\tau)\, d\tau$$

where $L_c \equiv (2-\beta)L - aC_d$ and

$$W(t) = \delta\dot{e}^{i^T}((2-\beta)L - 2aC_d + \dot{C}_d)\delta\dot{e}^i + 2a\delta\dot{e}^{i^T}\frac{1}{a}F_d\delta e^i$$
$$+ a^2\delta e^{i^T}\left((2-\beta)L + \frac{2}{a}F_d\right)\delta e^i.$$

From the assumptions, we obtain

$$W \geq l_a\|\delta\dot{e}^i\|^2 + 2a\delta\dot{e}^{i^T}\frac{1}{a}F_d\delta e^i + a^2 l_b\|\delta e^i\|^2.$$

Applying the Schwartz inequality, we obtain

$$W \geq \frac{1}{2}l_a\left(\|\delta\dot{e}^i\| - \frac{a}{l_a}\left\|\frac{1}{a}F_d\right\|\|\delta e^i\|\right)^2$$
$$+\frac{1}{2}a^2 l_b\left(\|\delta e^i\| - \frac{1}{al_b}\left\|\frac{1}{a}F_d\right\|\|\delta\dot{e}^i\|\right)^2$$
$$+\frac{1}{2}a^2\left(l_b - \frac{1}{l_a}\left\|\frac{1}{a}F_d\right\|^2\right)\|\delta e^i\|^2$$
$$+\frac{1}{2}\left(l_a - \frac{1}{l_b}\left\|\frac{1}{a}F_d\right\|^2\right)\|\delta\dot{e}^i\|^2$$
$$\geq 0.$$

This implies that for positive definite matrices C_d and L_c

$$\Delta V^i(t) = -\delta z^{i^T}(t)C_d\delta z^i(t) - a\delta e^{i^T}(t)L_c\delta e^i(t)$$
$$-\int_0^t W(\tau)\, d\tau \leq 0 \quad \text{for all} \quad t\in[0, t_f]$$

where $L_c \equiv (2 - \beta L - aC_d (> 0)$. Hence, statement i) of the theorem follows. The equality holds only when $\delta z^i = \delta e^i = \delta\dot{e}^i = 0$ for all $t\in[0, t_f]$. $\Delta V^i(t) \leq 0$ implies that V^i converges to a constant as $i\to\infty$. This, in turn, implies that δz^i and δe^i vanish as $i\to\infty$. Since $\delta z^i(t)$ vanishes for all $t\in[0, t_f]$, it also follows that $\delta\dot{z}^i(t) = 0$ for all $t\in[0, t_f]$ as $i\to\infty$. Therefore, the left-hand side of (B1) converges to zero as $i\to\infty$. Thus, $\lim_{i\to\infty} z^i(t) = 0$ for all $t\in[0, t_f]$. On the other hand, since $e^i(t)$ and $\dot{e}^i(t)$ are independent of each other, $\lim_{i\to\infty} e^i(t) = \lim_{i\to\infty}\dot{e}^i(t) = 0$ for all $t\in[0, t_f]$ implying ii). From (5), iii) follows, completing the proof. ■

Remark: We have used $2\|(\cdot)\|$ instead of $\|(\cdot) + (\cdot)^T\|$, where $(\cdot)$ represents a square matrix.

References

[1] J. S. Albus, *Brains, Behavior, and Robotics*. New York: Byte Books, 1981.

[2] ——, "Theoretical and experimental aspects of a cerebellar model," Ph.D. dissertation, Univ. of Maryland, 1972.

[3] S. Arimoto, S. Kawamura, and F. Miyasaki, "Bettering operation of robots by learning," *J. Robotic Syst.*, pp. 123-140, 1984.

[4] S. Arimoto and F. Miyasaki, "Stability of robustness of PID feedback control for robot manipulators of sensory capability," in *Proc. 1st Int. Symp. Robotic Res.*, 1983.

[5] B. Armstrong, Q. Khatib, and J. Burdick, "The explicit dynamic model and inertial parameters of the PUMA 560 arm," in *Proc. IEEE Robotics Automat. Conf.*, 1986.

[6] A. K. Bejezy, "Robot arm dynamics and control," Tech. Memo. 33-669, Jet Propulsion Lab., Pasadena, CA, 1974.

[7] P. Bondi, G. Casaline, and L. Gambardella, "On the iterative learning control theory for robotic manipulators," *IEEE J. Robotics Automat.*, vol. 4, no. 1, Feb. 1988.

[8] G. Casalino and G. Bartolini, "A learning procedure for the control of movements of robotic manipulators," presented at IASTED Symp. Robotics Automat., Amsterdam, The Netherlands, June 1984.

[9] J. J. Craig, "Adaptive control of manipulators through repeated trials," in *Proc. Amer. Control Conf.* (San Diego, CA), June 1984.

[10] ——, *Introduction to Robotics: Mechanics and Control*. Reading, MA: Addison-Wesley, 1986.

[11] J. J. Craig, P. Hsu, and S. Sastry, "Adaptive control of mechanical manipulators," presented at IEEE Int. Conf. Robotics and Automat., San Francisco, CA, 1986.

[12] S. V. Gusev, "Linear stabilization of nonlinear systems: Program motion," *Syst. Contr. Lett.*, vol. 11, pp. 409-412, 1988.

[13] S. Kawamura, F. Miyazaki, and S. Arimoto, "Realization of robot motion based on a learning method," *IEEE Trans. Syst. Man. Cybern.*, vol. 18, no. 1, Jan./Feb. 1988.

[14] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE Control Syst. Mag.*, Apr. 1988.

[15] T. Kuc and K. Nam, "CMAC based iterative learning control of robot manipulators," presented at IEEE CDC, Tampa, FL, Dec. 1989.

[16] C. S. G. Lee and M. J. Chung, "An adaptive control strategy for mechanical manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 837-840, 1984.

[17] W. T. Miller, III, F. H. Glanz, and L. G. Kraft, III, "Application of a general learning algorithm to the control of robotic manipulators," *Int. J. Robotics Res.*, vol. 6, no. 2, Summer 1987.

[18] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robot manipulator," *IEEE Trans. Neural Networks*, vol. 1, pp. 251-265, 1988.

[19] S. Oh, Z. Bien, and I. H. Suh, "An iterative learning control method with application for the robot manipulator," *IEEE J. Robotics Automat.*, vol. 4, no. 5, Oct. 1988.

[20] J.-J. E. Slotine and W. Lee, "Composite adaptive control of robot manipulators," *Automatica*, vol. 25, no. 4, pp. 509-519, 1989.

[21] X. Wang and L. Chen, "Proving the uniform boundedness of some commonly used control scheme for robot," presented at IEEE Int. Conf. Robotics Automat., Scottsdale, AZ, May 1980.

Computing Location and Orientation of Polyhedral Surfaces Using a Laser-Based Vision System

Din-Chang Tseng and Zen Chen

Abstract—**A laser-based vision system for computing the location and orientation of 3-D polyhedral surfaces is proposed. In this system, an expanded laser beam passes through a code plate marked with equally spaced vertical and horizontal lines and impinges on a polyhedral object to create a spatial-encoded image for analysis. Then, based on the vanishing points or the directly available line directions of the perceived grid lines on the polyhedral surface, the polyhedral surface orientation can be inferred. In the meantime, the given dimensions of the grid pattern on the plate are used to estimate the depth information of the polyhedral surfaces. More importantly, we shall solve the noise problem that occurs in the real image by a least squares estimation method and an iterative refinement method based on a geometric constraint criterion. Experiments are conducted to provide practical insight into the method. The experimental results indicate that the method is remarkably accurate and stable.**

I. Introduction

Computer vision endows machines with a visual capability with applications in robot navigation, surface measurement, object modeling, camera position determination, automatic target recognition, etc. In the literature, a number of methods have been presented for measuring the 3-D orientation, location, and structure of objects using structured light. In [1] and [2], a vertical slit projector and a TV camera were employed to construct a range finder based on a triangulation technique to acquire range data for measuring polyhedral surfaces. Hall *et al.* [3] used a mask of a known form to project easily detected features on the surface of the object. The known form mask and the recorded image were then used as a stereo image pair. Then by using a least squares method, the three-dimensional coordinates of points on the object were calculated from the transformation matrices of the image pair. These two methods require the point correspondences to compute the 3-D coordinates of points. However, the correspondence information is generally not easy to get. Wang *et al.* [4] used grid coding to infer the surface orientation and the structure of visible object surfaces. They used the direction of the projected stripes to infer local surface orientation and did not require any correspondence relationship either the grid lines or the grid junctions. To simplify the mathematical derivation, they assumed a parallel projection model. However, based on this model, the absolute depth can not be determined. Hu and Stockman [5] also presented a method for 3-D surface measurement using a projected grid of light. Based on triangulation computation and some geometric and topological constraints on the grid pattern, the 3-D surface points were computed. Again, the point correspondence information was used in this method.

Manuscript received September 26, 1988; revised February 20, 1991.

D.C. Tseng is with the Department of Electronic Engineering, National Central University, Chung-Li, Taiwan, Republic of China.

Z. Chen is with the Institute of Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

IEEE Log Number 9102767.

In this study, a new system for determining the 3-D location and orientation of polyhedral surfaces will be proposed. Compared to previous approaches, this system can find not only the orientation but also the depth parameter of polyhedral surfaces without using the point correspondences as required in the triangulation technique. Moreover, our method's computation model is nontrigonometric and rather simple. In this system a laser beam passes through a grid plate marked with equally spaced vertical and horizontal lines and impinges on a polyhedral object to create a spatial-encoded image for analysis. Based on the vanishing points or the directly available line directions of the perceived grid lines on the polyhedral surface, the polyhedral surface orientations can be inferred. In the meantime, the given dimensions of the grid pattern on the plate are used to estimate the depth information of the polyhedral surfaces. In order to deal with the noise in the real image, the extracted perceived grid pattern is first rectified by a least squares estimation method. Then an iterative method based on a geometric constraint criterion is employed to refine the polyhedral surface estimation. Experimental results indicate that the method is remarkably accurate and stable.

II. Geometric Configurations

A. Camera Geometry

Let the x_c, y_c, and z_c axes be the three principal axes of a camera geometry, and let the origin be at the camera lens center. Assume that the image plane is parallel to the $x_c y_c$ plane and its position is at $z_c = q$. The q value is roughly equal to the focal length for an object at a great distance. Assume that (x_i, y_i, z_i) is a 3-D point and (a_i, b_i) is its corresponding 2-D image point; then they are related by $a_i = qx_i/z_i$ and $b_i = qy_i/z_i$.

B. Laser Projector Geometry

A laser projector consisting of a laser source and a grid plate is used to generate two orthogonal sets of parallel sheets of light planes. The coordinate system of the laser projector is defined as follows. The two orthogonal grid lines on the plate are defined as the x_r and y_r axes, and the normal vector to the grid plate is defined as the z_r axis. The dimensions of the grid pattern on the plate are known. The relative orientations of the x_r, y_r, and z_r axes with respect to the camera coordinate system can be determined based on the same method, to be given later, that is used to determine the orientation of an unknown polyhedral surface. For this so-called laser calibration process, a planar surface with a known orientation (say, parallel to the grid plate) is used.

III. Grid Coding and Segmentation of Arbitrary Polyhedral Surfaces

A. Grid Coding of Polyhedral Surfaces

The perceived grid lines and grid points in the grid-coded polyhedron image need to be extracted in order to infer the 3-D location and orientation of the visible polyhedral surfaces. Since the perceived grid lines are the darkest pixels in the image, these lines can be extracted by a regular thresholding method. Also, a regular four-connected thinning algorithm can be applied to the thresholded grid lines to produce the lines. The following steps are used to obtain the 2-D coordinates of perceived grid points and a list of grid features to be used for polyhedral surface segmentation.

1) Grid Point Extraction: The thin grid lines obtained in the image are scanned row by row. The nearby branching pixels are detected at the place where two horizontal runs of grid pixels begin