[15] J. Angeles, G. Yang, and I. M. Chen, "Singularity analysis of three-legged, six-DOF platform manipulators with RRRS legs," in *Proc. 2001 IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics*, July 2001, pp. 32–36.
[16] F. C. Park and J. W. Kim, "Singularity analysis of closed kinematic chains," *ASME J. Mechan. Des.*, vol. 121, no. 2, pp. 32–38, Mar. 1999.
[17] G. Yang, I. M. Chen, W. Lin, and J. Angeles, "Singularity analysis of three-legged parallel robots based on passive joint velocities," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 413–422, Aug. 2001.
[18] C. M. Gosselin, "Determination of the workspace of 6-DOF parallel manipulators," *ASME J. Mechan. Des.*, vol. 112, pp. 331–336, Sept. 1990.
[19] I. A. Bonev and J. Ryu, "A new approach to orientation workspace analysis of 6-DOF parallel manipulators," *Mechanism and Machine Theory*, vol. 36, no. 1, pp. 15–28, 2001.
[20] E. F. Fichter, "A Stewart platform-based manipulator: General theory and practical construction," *Int. J. Robot. Res.*, vol. 5, no. 2, pp. 157–182, 1986.
[21] O. Masory and J. Wang, "Workspace evaluation of Stewart platforms," in *Proc. ASME 22nd Biennial Mechanisms Conf.*, vol. 45, 1992, pp. 337–346.
[22] T. Arai, T. Tanikawa, J.-P. Merlet, and T. Sendai, "Development of a new parallel manipulator with fixed linear actuator," in *ASME Japan/USA Symp. Flexible Automation*, 1996, pp. 145–149.
[23] E. Ottaviano and M. Ceccarelli, "Optimal design of CaPaMan (Cassino Parallel Manipulator) with a specified orientation workspace," *Robotica*, vol. 20, pp. 159–166, 2002.
[24] J.-P. Merlet, "Détermination de l'espace de travail d'un robot parallèle pour une orientation constante," *Mechanism and Machine Theory*, vol. 29, no. 8, pp. 1099–1113, 1994.
[25] R. P. Podhorodeski and K. H. Pittens, "A class of parallel manipulators based on kinematically simple branches," *ASME J. Mechan. Des.*, vol. 116, pp. 908–914, 1994.
[26] R. Brockett, "Robotic manipulators and the product of exponential formula," in *Int. Symp. Math. Theory of Network and Systems*, Israel, 1983, pp. 120–129.
[27] F. C. Park, "Computational aspect of manipulators via product of exponential formula for robot kinematics," *IEEE Trans. Automat. Contr.*, vol. 39, no. 9, pp. 643–647, 1994.
[28] R. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.
[29] C. M. Gosselin and J. Sefrioui, "Polynomial solutions for the direct kinematic problem of planar three-degree-of-freedom parallel manipulators," in *Proc. 5th Int. Conf. Advanced Robotics*, Pisa, Italy, June 19–22, 1991, pp. 1124–1129.
[30] L.-W. Tsai, *Robot Analysis*. New York: Wiley, 1999.
[31] G. Yang, W. H. Chen, and I. M. Chen, "A geometrical method for the singularity analysis of 3-RRR planar parallel robots with different actuation schemes," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2002, pp. 2055–2060.
[32] J. S. Rao and R. V. Dukkipati, *Mechanism and Machine Theory*. New Delhi, India: Wiley Eastern Ltd., 1989.

Smooth Motion Generation for Unicycle Mobile Robots Via Dynamic Path Inversion

Corrado Guarino Lo Bianco, Aurelio Piazzi, and Massimo Romano

***Abstract*—A new motion-generation approach is proposed for wheeled mobile robots described by the unicycle kinematic model. This approach permits the generation of smooth continuous-acceleration controls using a dynamic path-inversion procedure that exploits the concept of G^3-paths, i.e., Cartesian paths with third-order geometric continuity (both the curvature function and its derivative, with respect to the arc length, are continuous). The exposed steering method is well suited to be adopted for the robot's iterative steering within a supervisory control architecture for sensor-based autonomous navigation. A worked example illustrates the approach.**

***Index Terms*—Dynamic path inversion, geometric continuity, smooth motion generation, unicycle mobile robots.**

I. Introduction

The work of Dubins [1] and Reeds and Shepp [2] devoted to the planning with minimal length paths are well-known fundamental results in the field of nonholonomic motion generation of wheeled mobile robots and car-like vehicles (see the survey [3] by Laumond *et al.*). A known limit of the Dubins–Reeds–Shepp approach is that the resulting paths are composed by line segments and circular arcs and do not have overall continuous curvature. Therefore, swift high-performance maneuvers cannot be achieved. At the end of the 1980s, many authors [4]–[6] proposed several curve primitives to overtake this limit: in all of the cases, continuous-curvature paths, usually denoted as G^2-paths, were generated. Subsequently, many authors have worked on planning schemes ensuring continuous-curvature paths [7], [8]. A recent approach to continuous-curvature motion generation was proposed in [9] where continuous-curvature paths were obtained by using a new primitive, the $\boldsymbol{\eta}$-spline or quintic G^2-spline. This primitive was adopted to achieve a straightforward approach for the iterative steering of vision-based autonomous vehicles.

In this paper, the discussion is focused on the generation of a smooth motion control for unicycle mobile robots (UMRs), i.e., wheeled mobile robots whose kinematics is described by the unicycle model. In more detail, it will be demonstrated that, if such robots are driven with smooth linear and angular velocity command signals, i.e., signals which are continuous with their derivatives, they generate G^3-paths, i.e., paths for which both the curvature function and its derivative with respect to the arc length are continuous. It will be further demonstrated that, conversely, the UMRs can be driven along any G^3-path by means of appropriately devised C^1 command signals. A possible control strategy is then proposed. Its main features are the following.

1) A dynamic path inversion algorithm is designed to synthesize the UMR inputs. This algorithm requires planning with G^3-paths [10] using interpolating conditions at the path end-points.

Manuscript received July 9, 2003; revised December 19, 2003. This paper was recommended for publication by Associate Editor K. Lynch and Editor H. Arai upon evaluation of the reviewers' comments. This work was supported in part by MIUR Scientific Research Funds under the framework of a COFIN 2002 project. This paper was presented in part at the 2002 IEEE Intelligent Vehicles Symposium, Versailles, France, June 18–20, 2002.

The authors are with the Dipartimento di Ingegneria dell'Informazione, Università di Parma, I-43100 Parma, Italy (e-mail: guarino@ce.unipr.it; piazzi@ce.unipr.it; romano@ce.unipr.it).

Digital Object Identifier 10.1109/TRO.2004.832827

2) The UMR inputs, i.e., the linear and angular robot velocities, are globally generated as C^1 functions (velocities and accelerations are continuous time functions).

The proposed inversion-based procedure, which does not require any integration because it relies on the differential flatness of the unicycle model [11], will permit the real-time generation of continuous-acceleration feedforward inputs.

Due to the inevitable mismatch between the ideal unicycle model and the real one, the UMR tends to drift from the planned paths. These path errors can be reduced with a timely updating of the path replanning in accordance with an iterative steering technique [12]. As is known, this technique can be interpreted as a discrete-time feedback which is very useful in a real robot navigation scenario [13].

This paper is organized as follows. Section II introduces notation and preliminaries on first-, second-, and third-order geometric continuity of curves and paths. Section III poses a reachability problem in an extended state space of the unicycle model and derives a first result (*Proposition 1*). The dynamic path-inversion-based procedure is detailed in Section IV. A worked example illustrates the approach in Section V. The concluding remarks of Section VI end the paper.

II. Notation and Preliminaries on G^3-Paths

The Euclidean norm of a vector $\mathbf{p}$ is denoted by $\|\mathbf{p}\|$. Let C^i indicate the set of functions that are continuous until the ith derivative and let C_p indicate the class of piecewise continuous functions. A curve on the $\{x, y\}$ plane can be described by means of the following parametrization $\mathbf{p}(u)$:

$$\begin{aligned} \mathbf{p} : [u_0, u_1] &\to \mathbb{R}^2 \\ u &\to [\alpha(u)\beta(u)]^T \end{aligned} \tag{1}$$

where $[u_0, u_1]$ is a real closed interval. The "path" associated with the curve $\mathbf{p}(u)$ is $\mathbf{p}([u_0, u_1])$, i.e., the image of $[u_0, u_1]$ under the vectorial function $\mathbf{p}(u)$.

Definition 1: A curve $\mathbf{p}(u)$ is regular if $\dot{\mathbf{p}}(u) \in C_p([u_0, u_1])$ and $\dot{\mathbf{p}}(u) \neq \mathbf{0}\ \forall u \in [u_0, u_1]$.

The curve length measured along $\mathbf{p}(u)$ is denoted by s; it can be expressed as a function f of u as

$$\begin{aligned} f : [u_0, u_1] &\to [0, f(u_1)] \\ u \to s &= \int_{u_0}^{u} \|\dot{\mathbf{p}}(\xi)\|\, d\xi. \end{aligned} \tag{2}$$

Evidently, given a regular curve $\mathbf{p}(u)$, the length function $f(\cdot)$ is continuous over $[u_0, u_1]$ and bijective; hence, its inverse is continuous too and it will be denoted as

$$\begin{aligned} f^{-1} : [0, f(u_1)] &\to [u_0, u_1] \\ s &\to u = f^{-1}(s). \end{aligned} \tag{3}$$

Associated with every point of a regular curve $\mathbf{p}(u)$, there is the orthonormal moving reference frame $\{\boldsymbol{\tau}(u), \boldsymbol{\nu}(u)\}$ which is congruent with the axes of the $\{x, y\}$ plane and where $\boldsymbol{\tau}(u) = \dot{\mathbf{p}}(u)/\|\dot{\mathbf{p}}(u)\|$ denotes the unit tangent vector to the curve $\mathbf{p}(u)$. Define $\arg\{\boldsymbol{\tau}(\cdot)\}$ as the angle θ between $\boldsymbol{\tau}$ and the x axis. The angle θ is counterclockwise positive.

Definition 2 (G^1-curves): A parametric curve $\mathbf{p}(u)$ has first-order geometric continuity, and we say $\mathbf{p}(u)$ is a G^1-curve, if $\mathbf{p}(u)$ is regular and its unit tangent vector is a continuous function along the curve, i.e., $\boldsymbol{\tau}(\cdot) \in C^0([u_0, u_1])$.

For any regular curve such that $\ddot{\mathbf{p}}(u) \in C_p([u_0, u_1])$, the scalar curvature can be defined according to the Frenet formula $\dot{\boldsymbol{\tau}}(u) = \kappa_c(u)\boldsymbol{\nu}(u)$ (see, for example, [14, p. 109]). This defines the curvature function with respect to the parameter u as follows:

$$\begin{aligned} \kappa_c : [u_0, u_1] &\to \mathbb{R} \\ u &\to \kappa_c(u). \end{aligned} \tag{4}$$

According to the theory of planar curves, an explicit expression of $\kappa_c(u)$ is $\kappa_c(u) = [\dot{\alpha}(u)\ddot{\beta}(u) - \ddot{\alpha}(u)\dot{\beta}(u)]/(\dot{\alpha}^2(u) + \dot{\beta}^2(u))^{3/2}$. The scalar curvature can also be expressed as a function of the curve length s. In the following, such a function will be indicated as

$$\begin{aligned} \kappa : [0, f(u_1)] &\to \mathbb{R} \\ s &\to \kappa(s) \end{aligned} \tag{5}$$

and the bijectivity of function $f(u)$ makes it possible to write

$$\kappa(s) = \kappa_c\left(f^{-1}(s)\right). \tag{6}$$

Evidently, by virtue of relation (6), the curvature function $\kappa_c(u)$ is continuous if and only if function $\kappa(s)$ is continuous, i.e., $\kappa_c(\cdot) \in C^0([u_0, u_1]) \Leftrightarrow \kappa(\cdot) \in C^0([0, f(u_1)])$.

Definition 3 (G^2-curves): A parametric curve $\mathbf{p}(u)$ has second-order geometric continuity, and we say $\mathbf{p}(u)$ is a G^2-curve, if $\mathbf{p}(u)$ is a G^1-curve, $\ddot{\mathbf{p}}(\cdot) \in C_p([u_0, u_1])$ and its scalar curvature is continuous along the curve, i.e., $\kappa_c(\cdot) \in C^0([u_0, u_1])$ or $\kappa(\cdot) \in C^0([0, f(u_1)])$.

G^1- and G^2-curves were originally introduced by Barsky and Beatty [15] in a computer graphics context. The main results of this paper require the introduction of curves with third-order geometric continuity. This can be done according to the following definition.

Definition 4 (G^3-curves): A parametric curve $\mathbf{p}(u)$ has third-order geometric continuity, and we say that $\mathbf{p}(u)$ is a G^3-curve, if $\mathbf{p}(u)$ is a G^2-curve, $\dddot{\mathbf{p}}(\cdot) \in C_p([u_0, u_1])$, and the derivative of the scalar curvature, with respect to the arc length s, is continuous along the curve, i.e., $\dot{\kappa}(\cdot) \in C^0([0, f(u_1)])$.

Definition 5 (G^1-, G^2-, and G^3-paths): A path of a Cartesian space, i.e., a set of points of this space, is a G^i-path ($i = 1, 2, 3$) if there exists a parametric G^i-curve whose image is the given path.

Remark 1: The definitions provided herein for the geometric continuity are not the most general ones and are restricted to planar curves. A recent survey devoted to the kth-order geometric continuity of curves and surfaces in a general context can be found in [16].

III. The Problem and a First Result

Consider a wheeled mobile robot governed by the nonholonomic unicycle model

$$\dot{x}(t) = v(t)\cos\theta(t) \tag{7}$$

$$\dot{y}(t) = v(t)\sin\theta(t) \tag{8}$$

$$\dot{\theta}(t) = \omega(t) \tag{9}$$

where x and y indicate the robot position with respect to a stationary frame, θ is its heading angle, and v and ω are its linear and angular velocities to be considered as the control inputs of the robot. In order to achieve high-motion performances, these inputs $v(t)$ and $\omega(t)$ will be synthesized as C^1-functions, i.e., linear and angular accelerations will be continuous signals.

From a mathematical standpoint, the state of model (7)–(9), at time t, is given by $\{x(t), y(t), \theta(t)\}$. In the following, it is convenient to use an *extended state*, comprising the inputs and their derivatives, defined as

$$\{x(t), y(t), \theta(t), v(t), \dot{v}(t), \omega(t), \dot{\omega}(t)\}.$$

Then, the considered motion generation problem can be stated as a reachability problem in the extended state space.

The Problem: Given any assigned traveling time $t_f > 0$, find control inputs $v(\cdot), \omega(\cdot) \in C^1([0, t_f])$ such that the mobile robot starting from an arbitrary initial extended state

$$\begin{aligned}\mathbf{p}_A &= [x_A \; y_A]^T = [x(0) \; y(0)]^T \\ \theta_A &= \theta(0) \\ v_A &= v(0) \\ \dot{v}_A &= \dot{v}(0) \\ \omega_A &= \omega(0) \\ \dot{\omega}_A &= \dot{\omega}(0)\end{aligned}$$

reaches the arbitrary final extended state

$$\begin{aligned}\mathbf{p}_B &= [x_B \; y_B]^T = [x(t_f) \; y(t_f)]^T \\ \theta_B &= \theta(t_f) \\ v_B &= v(t_f) \\ \dot{v}_B &= \dot{v}(t_f) \\ \omega_B &= \omega(t_f) \\ \dot{\omega}_B &= \dot{\omega}(t_f).\end{aligned}$$

A solution to the introduced problem will be provided by the path dynamic inversion procedure described in Section IV and the overall motion strategy can be then based on an iterative steering [12] issued by a supervisory control system [17]. The real-time knowledge of the robot position is used by the supervisory system to steer the UMR from the current extended state to a future extended state in an iterative fashion. In such a way, for the UMR, swift high-performance motion is possible while intelligent or elaborate behaviors are performed.

The following proposition that is the first contribution of the paper is essential to understand how to plan a desired path connecting $\mathbf{p}_A$ with $\mathbf{p}_B$.

Proposition 1: Assign any $t_f > 0$. If a Cartesian path is generated by model (7)–(9) with inputs $v(t), \omega(t) \in C^1([0, t_f])$, and $v(t) \neq 0 \; \forall t \in [0, t_f]$, then it is a G^3-path. Conversely, given any G^3-path, there exist inputs $v(t), \omega(t) \in C^1([0, t_f])$ with $v(t) \neq 0 \; \forall t \in [0, t_f]$ and initial conditions such that the path generated by model (7)–(9) coincides with the given G^3-path.

Proof: Let us demonstrate the first part of the proposition. Consider a Cartesian planar $\{x, y\}$-path generated by the model (7)–(9) by means of two command signals $v(t)$ and $\omega(t)$ continuous with their derivatives. The generated path can be found by explicitly solving (7)–(9) for any t belonging to a given time interval $[0, t_f]$. Let us indicate by $[x(t), y(t)]^T$ the generated trajectory. A known result of the planar curve theory makes it possible to express the curve unit tangent vector as

$$\boldsymbol{\tau}(t) = \frac{[\dot{x}(t) \; \dot{y}(t)]^T}{\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}} = [\cos\theta(t) \; \sin\theta(t)]^T, \quad \text{if } v(t) > 0$$
$$\boldsymbol{\tau}(t) = \frac{[\dot{x}(t) \; \dot{y}(t)]^T}{\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}} = -[\cos\theta(t) \; \sin\theta(t)]^T, \quad \text{if } v(t) < 0.$$

Taking into account model (7)–(9) and the continuity of ω, it is possible to deduce that θ and, consequently, the unit tangent vector $\boldsymbol{\tau}$, are continuous.

The scalar curvature for a planar path is defined as

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{\frac{3}{2}}}. \tag{10}$$

Explicit expressions for $\ddot{x}$ and $\ddot{y}$ can be derived from (7)–(9) as

$$\ddot{x}(t) = \dot{v}(t)\cos\theta(t) - v(t)\omega(t)\sin\theta(t) \tag{11}$$
$$\ddot{y}(t) = \dot{v}(t)\sin\theta(t) + v(t)\omega(t)\cos\theta(t). \tag{12}$$

By using (11)–(12) and the model (7)–(9), (10) becomes

$$\kappa(t) = \frac{\omega(t)}{v(t)}, \quad \text{if } v(t) > 0 \tag{13}$$
$$\kappa(t) = -\frac{\omega(t)}{v(t)}, \quad \text{if } v(t) < 0. \tag{14}$$

Hence, the continuity on v and ω infers the continuity of the curvature κ.

Finally, the curvature derivative with respect to curve length s is expressed as

$$\frac{d\kappa(t)}{ds} = \frac{\frac{d\kappa(t)}{dt}}{\frac{ds(t)}{dt}} = \frac{\dot{\omega}(t)v(t) - \omega(t)\dot{v}(t)}{v^2(t)} \frac{1}{v(t)}.$$

As a consequence, it is possible to assert that, according to the continuity of v, $\dot{v}$, ω, and $\dot{\omega}$, the curvature derivative $d\kappa/ds$ is continuous also. This demonstrates that the path generated by model (7)–(9) is a G^3-path.

The second part of the proposition will be demonstrated constructively. Suppose that we want to assign a path in the $\{x, y\}$ plane by means of a G^3-curve parametrized by u, i.e., $\mathbf{p}(u) := [\alpha(u) \; \beta(u)]^T$ with $u \in [u_0, u_1]$.

Let us assume as initial conditions $x(0) = \alpha(u_0)$, $y(0) = \beta(u_0)$, and $\theta(0) = \arg\{\boldsymbol{\tau}(u_0)\}$. Freely assign $v(t) \in C^1([0, t_f])$ with $t \in [0, t_f]$ such that $v(t) > 0$, and the equation

$$\int_0^{t_f} v(\xi)d\xi = f(u_1) \tag{15}$$

is satisfied.

Finally, assign $\omega(t)$ according to the following equation:

$$\omega(t) := v(t)\kappa(s)\big|_{s=\int_0^t v(\xi)d\xi}. \tag{16}$$

Owing to the G^3 continuity of the planned curve, we can claim the continuity of $\kappa(s)$ so that, since $v(t) \in C^1([0, t_f])$, it is possible to conclude that $\omega(t) \in C^0([0, t_f])$. Moreover, its first derivative can be expressed as

$$\dot{\omega}(t) = v^2(t) \left.\frac{d\kappa}{ds}\right|_{s=\int_0^t v(\xi)d\xi} + \dot{v}(t)\kappa(s)\big|_{s=\int_0^t v(\xi)d\xi}. \tag{17}$$

All of the functions appearing in (17) are continuous. This proves that, as required, $\omega(t) \in C^1([0, t_f])$.

To conclude the demonstration of the second part of the proposition, it is necessary to prove that the following time functions:

$$\alpha(u)\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} \tag{18}$$
$$\beta(u)\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} \tag{19}$$
$$\arg\{\tau(u)\}\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} \tag{20}$$

are a solution of model (7)–(9). First of all, it is immediately possible to verify that (18)–(20) satisfy the given initial conditions. By differentiating with respect to the time both (18) and (19), we can write

$$\begin{aligned}&\frac{d}{dt}\alpha(u)\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} \\ &= \left.\frac{\dot{\alpha}(u)}{\|\dot{\mathbf{p}}(u)\|}\right|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} v(t) \\ &= v(t)\cos\left[\arg\left(\frac{\dot{\mathbf{p}}(u)}{\|\dot{\mathbf{p}}(u)\|}\right)\right]\Bigg|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} \\ &= v(t)\cos\left[\arg\left(\boldsymbol{\tau}(u)\right)\right]\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)}\end{aligned} \tag{21}$$

$$\begin{aligned}\frac{d}{dt}\beta(u)\Big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} &= \frac{\dot\beta(u)}{\|\dot{\mathbf{p}}(u)\|}\Bigg|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} v(t)\\ &= v(t)\sin\left[\arg\left(\frac{\dot{\mathbf{p}}(u)}{\|\dot{\mathbf{p}}(u)\|}\right)\right]\Bigg|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)}\\ &= v(t)\sin\left[\arg\left(\boldsymbol{\tau}(u)\right)\right]\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)}. \end{aligned} \tag{22}$$

Equations (21) and (22) exactly match the first two equations of model (7)–(9).

The time derivative of (20) can be evaluated bearing in mind (23), shown at the bottom of the page. Thus

$$\begin{aligned}\frac{d}{dt}\arg\left[\boldsymbol{\tau}(u)\right]\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} &= v(t)\,\frac{\dot\alpha(u)\ddot\beta(u)-\ddot\alpha(u)\dot\beta(u)}{\left(\alpha^2(u)+\beta^2(u)\right)^{\frac{3}{2}}}\Bigg|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)}\\ &= v(t)\,\kappa_c(u)\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)}. \end{aligned}\tag{24}$$

Definition 6 makes it possible to rewrite (16) as

$$\omega(t) := v(t)\kappa_c(u)\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} \tag{25}$$

so we can conclude that

$$\frac{d}{dt}\arg\left[\boldsymbol{\tau}(u)\right]\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} = \omega(t) \tag{26}$$

i.e., (26) matches the third equation of model (7)–(9).

We can conclude that the trajectory (18)–(20) is generated by model (7)–(9) owing to the chosen input functions $v(t)$ and $\omega(t)$. Hence, the path generated by (7)–(9) coincides with the planned curve $\mathbf{p}(u)$. ■

Remark 2: In order to generate G^3-paths with model (7)–(9), it is not strictly necessary to guarantee that $v(t) \neq 0$. On the other hand, if such a condition is not satisfied, there exist degenerate situations where the required geometric continuity is lost. For example, assign two command signals $v(t), \omega(t) \in C^1([0, t_f])$ such that there exists $t^* \in [0, t_f]$ where $v(t^*) = 0$ and $\omega(t^*) \neq 0$. It is immediately possible to verify that, according to (10)–(12), we have

$$\kappa(t^*) = \lim_{t\to t^*}\frac{v^2(t)\omega(t)}{\left[v^2(t)\right]^{\frac{3}{2}}} = \infty \tag{27}$$

hence, the curvature κ is not defined at t^*. This implies that, according to *Definitions 3–5*, the corresponding path is neither a G^2-path nor a G^3-path.

IV. Path-Inversion Algorithm

In this section, the proposed motion generation problem is solved by means of a dynamic path-inversion procedure. The aim is to synthesize a feedforward control such that, for any given interval $[0, t_f]$, the UMR starting at time 0 from the extended state

$$A = \{\mathbf{p}_A, \theta_A, v_A, \dot v_A, \omega_A, \dot\omega_A\}$$

will reach, at time t_f, the extended state

$$B = \{\mathbf{p}_B, \theta_B, v_B, \dot v_B, \omega_B, \dot\omega_B\}$$

while the following geometric and kinematics requirements are ensured:

I) The robot Cartesian path connecting $\mathbf{p}_A$ with $\mathbf{p}_B$ is a G^3-path.
II) $v(t) \neq 0\ \forall t \in (0, t_f)$.

The above conditions I and II impose some formal restrictions on both the initial and final extended states. In order to satisfy condition I, all of the following statements must be simultaneously verified.

1) If $v_A = 0$, then $\omega_A = 0$.
2) If $(v_A = 0) \wedge (\dot v_A = 0)$, then $\omega_A = 0, \dot\omega_A = 0$.
3) If $v_B = 0$, then $\omega_B = 0$.
4) If $(v_B = 0) \wedge (\dot v_B = 0)$, then $\omega_B = 0, \dot\omega_B = 0$.

On the other hand, condition II is satisfied only if all of the following statements are true.

5) If $v_A > 0$, then $v_B \geq 0$.
6) If $(v_A > 0) \wedge (v_B = 0)$, then $\dot v_B \leq 0$.
7) If $v_A < 0$, then $v_B \leq 0$.
8) If $(v_A < 0) \wedge (v_B = 0)$, then $\dot v_B \geq 0$.
9) If $(v_A = 0) \wedge (\dot v_A > 0)$, then $v_B \geq 0$.
10) If $(v_A = 0) \wedge (\dot v_A > 0) \wedge (v_B = 0)$, then $\dot v_B \leq 0$.
11) If $(v_A = 0) \wedge (\dot v_A < 0)$, then $v_B \leq 0$.
12) If $(v_A = 0) \wedge (\dot v_A < 0) \wedge (v_B = 0)$, then $\dot v_B \geq 0$.

In light of *Proposition 1*, it is sensible to impose that the path connecting $\mathbf{p}_A$ with $\mathbf{p}_B$ is a G^3-path even for the cases where $v(0) = 0$ and/or $v(t_f) = 0$. Consequently, it is necessary to enforce statements 1)–4). Indeed, if any of the statements from 1) to 4) are violated, the corresponding robot motion path cannot be a G^3-path

Analogously, focusing on statements 5)–12), if any of these are violated then, considering that $v(t)$ must be synthesized as a C^1-function, condition II cannot be satisfied. For example, assume that statement 6) is not true because $v_A > 0$, $v_B = 0$, and $\dot v_B > 0$. Therefore, it is easy to show that there does not exist a function $v(\cdot) \in C^1([0, t_f])$ satisfying $v(0) = v_A$, $v(t_f) = v_B$, $\dot v(t_f) = \dot v_B$, and $v(t) > 0\ \forall t \in (0, t_f)$.

Condition II simply means that we can have either $v(t) > 0$ or $v(t) < 0$ for all of the time instants t belonging to the open interval $(0, t_f)$. This should not be considered a loss of generality. Indeed, if the robot motion direction needs to be inverted as, for example, when there exists $t^* \in (0, t_f)$, such that $v(t) > 0\ \forall t \in (0, t^*)$ and $v(t) < 0\ \forall t \in (t^*, t_f)$, then the supervisor can split the motion generation into two parts, each of them characterized by a well-defined sign of the velocity inside the pertinent time interval. In the following, we denote as a forward movement (FM) the robot motion for which $v(t) > 0\ \forall t \in (0, t_f)$. Conversely, a backward movement (BM) is characterized by $v(t) < 0\ \forall t \in (0, t_f)$.

The overall procedure can be described with four steps. First, the supervisor has to decide for a FM or BM to reach the final extended state B. This choice mainly depends on the given interpolating data. The same data are used in the second step to plan the desired G^3-path. Then, the linear velocity command signal $v(t)$ is synthesized and, finally, the angular velocity command $\omega(t)$ is designed by exploiting the constructive proof provided for *Proposition 1*.

$$\arg\left[\boldsymbol{\tau}(u)\right]\big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)} = \begin{cases}\arctan\left[\frac{\dot\beta(u)}{\dot\alpha(u)}\right]\Big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)}, & \text{if } \dot\alpha(u) > 0\\ \pi + \arctan\left[\frac{\dot\beta(u)}{\dot\alpha(u)}\right]\Big|_{u=f^{-1}\left(\int_0^t v(\xi)d\xi\right)}, & \text{if } \dot\alpha(u) < 0\\ \frac{\pi}{2}, & \text{if } \dot\alpha(u) = 0 \text{ and } \dot\beta(u) > 0\\ -\frac{\pi}{2}, & \text{if } \dot\alpha(u) = 0 \text{ and } \dot\beta(u) < 0\end{cases} \tag{23}$$

TABLE I
SELECTION CRITERIA FOR THE MOTION DIRECTION

$(v_A > 0) \wedge (v_B > 0)$	FM
$(v_A > 0) \wedge (v_B = 0) \wedge (\dot{v}_B \leq 0)$	
$(v_A = 0) \wedge (v_B > 0) \wedge (\dot{v}_A \geq 0)$	
$(v_A = 0) \wedge (v_B = 0) \wedge (\dot{v}_A \geq 0) \wedge (\dot{v}_B < 0)$	
$(v_A = 0) \wedge (v_B = 0) \wedge (\dot{v}_A > 0) \wedge (\dot{v}_B \leq 0)$	
$(v_A < 0) \wedge (v_B < 0)$	BM
$(v_A < 0) \wedge (v_B = 0) \wedge (\dot{v}_B \geq 0)$	
$(v_A = 0) \wedge (v_B < 0) \wedge (\dot{v}_A \leq 0)$	
$(v_A = 0) \wedge (v_B = 0) \wedge (\dot{v}_A \leq 0) \wedge (\dot{v}_B > 0)$	
$(v_A = 0) \wedge (v_B = 0) \wedge (\dot{v}_A < 0) \wedge (\dot{v}_B \geq 0)$	
$(v_A = v_B = 0) \wedge (\dot{v}_A = \dot{v}_B = 0)$	FM or BM

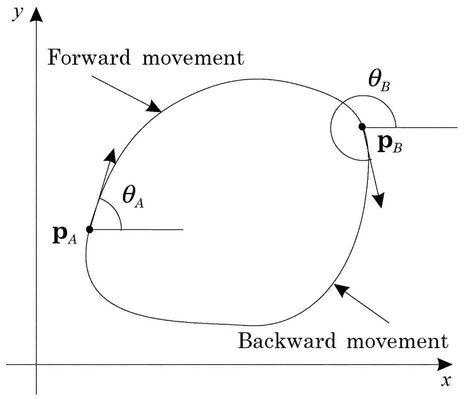

Fig. 1. If $v_A = v_B = 0$ and $\dot{v}_A = \dot{v}_B = 0$, then B can be reached by means of an FM or a BM.

Step 1: The motion direction is decided on the basis of the interpolating data v_A, $\dot{v}_A$, v_B, and $\dot{v}_B$. Table I can be used to select the motion direction (FM or BM). Note that, in the last case of Table I ($v_A = v_B = 0$ and $\dot{v}_A = \dot{v}_B = 0$), the supervisor can arbitrarily choose an FM or a BM (see Fig. 1).

Step 2: Determine a G^3-path connecting $\mathbf{p}_A$ with $\mathbf{p}_B$. This corresponds to finding a G^3-curve $\mathbf{p}(u)$, denoted according to (1), that satisfies the interpolation data deduced from the extended states A and B.

Two cases can be distinguished:

1) a general case: $v_A \neq 0$ and $v_B \neq 0$;
2) a critical case: $v_A = 0$ and/or $v_B = 0$.

For both cases, the curve $\mathbf{p}(u)$ must satisfy the following interpolating conditions:

$$\mathbf{p}(u_0) = \mathbf{p}_A, \quad \mathbf{p}(u_1) = \mathbf{p}_B \tag{28}$$

$$\boldsymbol{\tau}(u_0) = \begin{cases} [\cos\theta_A \ \sin\theta_A]^T, & \text{if FM} \\ [-\cos\theta_A \ -\sin\theta_A]^T, & \text{if BM} \end{cases} \tag{29}$$

$$\boldsymbol{\tau}(u_1) = \begin{cases} [\cos\theta_B \ \sin\theta_B]^T, & \text{if FM} \\ [-\cos\theta_B \ -\sin\theta_B]^T, & \text{if BM.} \end{cases} \tag{30}$$

From Section II, let us recall that $\kappa(s)$ and $\dot{\kappa}(s)$ denote the curvature and its derivative with respect to the arc length s [in particular, see (6)]. Then define

$$\kappa_A := \kappa(0), \quad \dot{\kappa}_A = \dot{\kappa}(0),$$
$$\kappa_B := \kappa(f(u_1)), \quad \dot{\kappa}_B := \dot{\kappa}(f(u_1)).$$

For the general case, the curvatures and their derivatives at the path end-points must be determined according to the formulas

$$\kappa_A = \begin{cases} \frac{\omega_A}{v_A} & \text{if FM} \\ -\frac{\omega_A}{v_A}, & \text{if BM} \end{cases} \tag{31}$$

$$\kappa_B = \begin{cases} \frac{\omega_B}{v_B} & \text{if FM} \\ -\frac{\omega_B}{v_B}, & \text{if BM} \end{cases} \tag{32}$$

$$\dot{\kappa}_A = \frac{\dot{\omega}_A v_A - \omega_A \dot{v}_A}{v_A^3} \tag{33}$$

$$\dot{\kappa}_B = \frac{\dot{\omega}_B v_B - \omega_B \dot{v}_B}{v_B^3}. \tag{34}$$

Now, consider the critical case. If $v_A = 0$, $\dot{v}_A = 0$, $\omega_A = 0$, and $\dot{\omega}_A = 0$, then the supervisor can freely assign any desired κ_A and $\dot{\kappa}_A$. Analogously, if $v_B = 0$, $\dot{v}_B = 0$, $\omega_B = 0$, and $\dot{\omega}_B = 0$, then κ_B and $\dot{\kappa}_B$ can be arbitrarily chosen.

On the other hand, if $v_A = 0$, $\omega_A = 0$, and $\dot{v}_A \neq 0$, then

$$\kappa_A = \begin{cases} \frac{\dot{\omega}_A}{\dot{v}_A}, & \text{if FM } (\dot{v}_A > 0) \\ -\frac{\dot{\omega}_A}{\dot{v}_A}, & \text{if BM } (\dot{v}_A < 0) \end{cases} \tag{35}$$

and $\dot{\kappa}_A$ can be freely assigned. Analogously, if $v_B = 0$, $\omega_B = 0$, and $\dot{v}_B \neq 0$, then

$$\kappa_B = \begin{cases} \frac{\dot{\omega}_B}{\dot{v}_B}, & \text{if FM } (\dot{v}_B > 0) \\ -\frac{\dot{\omega}_B}{\dot{v}_B}, & \text{if BM } (\dot{v}_B < 0) \end{cases} \tag{36}$$

and $\dot{\kappa}_B$ is arbitrarily chosen.

The interpolating conditions on curve $\mathbf{p}(u)$ are then completed by imposing [see (6)]

$$\kappa_c(u_0) = \kappa_A \tag{37}$$
$$\kappa_c(u_1) = \kappa_B \tag{38}$$
$$\dot{\kappa}_c(u_0) = \dot{\kappa}_A \, \|\dot{\mathbf{p}}(u_0)\| \tag{39}$$
$$\dot{\kappa}_c(u_1) = \dot{\kappa}_B \, \|\dot{\mathbf{p}}(u_1)\|. \tag{40}$$

An actual G^3-curve $\mathbf{p}(u)$ can be obtained by means of the closed-form expressions set out in [10]. In that work, a new curve primitive, named G^3-spline, was proposed. Such a primitive makes it possible to satisfy any set of interpolating conditions (28)–(30) or (37)–(40) and, at the same time, to finely shape the resulting curve by means of some freely assignable parameters.

Step 3: Choose $v(\cdot) \in C^1([0, t_f])$ with $v(t) \neq 0 \ \forall t \in (0, t_f)$, such that

$$v(0) = v_A \tag{41}$$
$$v(t_f) = v_B \tag{42}$$
$$\dot{v}(0) = \dot{v}_A \tag{43}$$
$$\dot{v}(t_f) = \dot{v}_B \tag{44}$$

$$\int_0^{t_f} v(\xi) d\xi = \begin{cases} f(u_1), & \text{if FM} \\ -f(u_1), & \text{if BM} \end{cases} \tag{45}$$

(we recall that $f(u_1)$ is the total arc length of the planned G^3-path connecting $\mathbf{p}_A$ with $\mathbf{p}_B$). Choosing $v(t)$ according to the interpolating conditions, (41)–(45) can be accomplished according to various schemes.

A viable velocity planning has been recently proposed in [18]. The command signal $v(\cdot) \in C^1([0, t_f])$ is generated with five properly joined spline curves ($i = 1, 2, \ldots, 5$):

$$v_i(t) = a_{1i} + 2a_{2i}t + 3a_{3i}t^2, \quad t \in [0, h_i] \tag{46}$$

with $\sum_{i=1}^{5} h_i = t_f$. In [18], it has been demonstrated that the generated velocity function, in the absence of velocity and acceleration upper bounds, is C^1 and strictly positive for any $t \in (0, t_f)$ and, moreover, satisfies with certainty constraints (41)–(45) for any set of interpolating conditions. When kinematics bounds have to be considered, as is certainly the case in the practical applications, it is still possible to use this velocity planning by applying a time-scaling procedure on t_f.

Step 4: The angular velocity function $\omega(\cdot) \in C^1([0, t_f])$ is defined according to

$$\omega(t) := v(t)\kappa(s)\big|_{s=\int_0^t v(\xi)d\xi} \quad \forall t \in [0, t_f] \text{ if FM} \tag{47}$$

$$\omega(t) := -\, v(t)\kappa(s)\big|_{s=-\int_0^t v(\xi)d\xi} \quad \forall t \in [0, t_f] \text{ if BM} \tag{48}$$

where $\kappa(s)$ is the curvature expressed as a function of the arc length s [see (6)].

The following result highlights the role of the inversion algorithm in steering the UMR.

Proposition 2: Let us consider any traveling time $t_f > 0$ and any extended states A and B satisfying assumptions 1)–12). Then, the control inputs $v(\cdot), \omega(\cdot) \in C^1([0, t_f])$, synthesized by the proposed procedure, steer the UMR from the extended state A, at time 0, to the extended state B, at time t_f, in such a way that the generated motion path exactly matches the G^3-path planned at step 2 of the procedure.

Proof: This proof mainly relies on the "sufficiency" proof of *Proposition 1*.

First note that, as required, both $v(t)$ and $\omega(t)$ belong to $C^1([0, t_f])$. By definition, velocity $v(t)$ is $C^1([0, t_f])$ when planned according to the method proposed in [18]. It has already been demonstrated that an angular command signal defined according to (16) belongs to $C^1([0, t_f])$ when the planned curve is G^3. Thus, in the case of FM, the command signal (47) belongs to $C^1([0, t_f])$. Using similar reasonings in the case of BM, the command (48) belongs to $C^1([0, t_f])$.

As a second step, it is necessary to prove that the path generated by the robot exactly matches the planned G^3-path. In the case of FM, in the proof of *Proposition 1*, it has been demonstrated that the trajectory (18)–(20) is generated by model (7)–(9) with inputs (46) and (47) and initial state $[x_A\ y_A\ \theta_A]^T$. Thus, the path generated by (7)–(9) coincides with the planned curve $\mathbf{p}(u)$ so that, at time t_f, the robot state exactly coincides with the desired final state $[x_B\ y_B\ \theta_B]^T$. It is relevant to observe that the demonstration does not degenerate even in the case of initial and final velocities equal to zero.

In the case of BM, it is immediately possible to verify that $\theta(\cdot) = \arg\{\boldsymbol{\tau}(\cdot)\}$ so that it is necessary to demonstrate that the following time functions:

$$\alpha(u)\big|_{u=f^{-1}\left(-\int_0^t v(\xi)d\xi\right)} \tag{49}$$

$$\beta(u)\big|_{u=f^{-1}\left(-\int_0^t v(\xi)d\xi\right)} \tag{50}$$

$$\arg\{-\tau(u)\}\big|_{u=f^{-1}\left(-\int_0^t v(\xi)d\xi\right)} \tag{51}$$

satisfy model (7)–(9) with command inputs (46) and (48) and initial state $[x_A\ y_A\ \theta_A]^T$. The demonstration is analogous to that already seen for the FM and is omitted for conciseness.

To conclude the proof, it is necessary to demonstrate that the boundary conditions are satisfied not only for the robot state but also for the whole "extended state." The linear velocity profile is planned by imposing (41)–(44) so that the boundary conditions are automatically satisfied for both $v(t)$ and $\dot{v}(t)$. The demonstration for $\omega(t)$ and $\dot{\omega}(t)$ requires one to consider several cases depending on the initial and final linear velocities $v(0)$ and $v(t_f)$. In the following, only the FM case will be analyzed.

First consider the nondegenerate case $v(0) \neq 0$. Taking into account the command signal (47) and the first equation of (31), we obtain

$$\omega(0) = v(0)\kappa(0) = v_A\kappa_A = v_A\frac{\omega_A}{v_A} = \omega_A.$$

The angular acceleration $\dot{\omega}(0)$ can be obtained by deriving (47) [see also (17)]. The initial angular acceleration is obtained with the help of (31) and (33) as follows:

$$\begin{aligned}\dot{\omega}(0) &= v^2(0)\dot{\kappa}(0) + \dot{v}(0)\kappa(0)\\ &= v_A^2\dot{\kappa}_A + \dot{v}_A\kappa_A\\ &= v_A^2\frac{\dot{\omega}_A v_A - \omega_A\dot{v}_A}{v_A^3} + \dot{v}_A\frac{\omega_A}{v_A}\\ &= \dot{\omega}_A.\end{aligned}$$

Analogously, for $t = t_f$ and $v(t_f) \neq 0$, (32) makes it possible to write

$$\omega(t_f) = v(t_f)\kappa(t_f) = v(t_f)\kappa(f(u_1)) = v_B\kappa_B = v_B\frac{\omega_B}{v_B} = \omega_B.$$

Moreover, considering also (34), we obtain

$$\begin{aligned}\dot{\omega}(t_f) &= v^2(t_f)\dot{\kappa}(t_f) + \dot{v}(t_f)\kappa(t_f)\\ &= v^2(t_f)\dot{\kappa}(f(u_1)) + \dot{v}(t_f)\kappa(f(u_1))\\ &= v_B^2\dot{\kappa}_B + \dot{v}_B\kappa_B\\ &= v_B^2\frac{\dot{\omega}_B v_B - \omega_B\dot{v}_B}{v_B^3} + \dot{v}_B\frac{\omega_A}{v_B}\\ &= \dot{\omega}_B.\end{aligned}$$

This demonstrates that, in case of nondegenerate situations, the initial and final interpolating conditions are exactly matched.

Now consider the critical situation where $v_A = 0$ and $\dot{v}_A = 0$. For any finite value of κ_A and $\dot{\kappa}_A$, we correctly obtain [see statement 2) of Section IV]

$$\begin{aligned}\omega(0) &= v(0)\kappa(0) = v_A\kappa_A = 0\\ \dot{\omega}(0) &= v^2(0)\dot{\kappa}(0) + \dot{v}(0)\kappa(0) = v_A^2\dot{\kappa}_A + \dot{v}_A\kappa_A = 0.\end{aligned}$$

Analogously, when $v_B = 0$ and $\dot{v}_B = 0$, we correctly obtain [see statement 4) of Section IV]

$$\begin{aligned}\omega(t_f) &= v(t_f)\kappa(t_F) = v(t_f)\kappa(f(u_1)) = v_B\kappa_B = 0\\ \dot{\omega}(t_f) &= v^2(t_f)\dot{\kappa}(t_f) + \dot{v}(t_f)\kappa(t_f)\\ &= v^2(t_f)\dot{\kappa}(f(u_1)) + \dot{v}(t_f)\kappa(f(u_1))\\ &= v_A^2\dot{\kappa}_A + \dot{v}_A\kappa_A\\ &= 0.\end{aligned}$$

To conclude, consider the last critical case $v_A = 0$ and $\dot{v}_A \neq 0$. For the FM, taking into account (35), the initial angular velocity is correctly [see statement 1) of Section IV]

$$\omega(0) = v(0)\kappa(0) = v_A\kappa_A = v_A\frac{\dot{\omega}_A}{\dot{v}_A} = 0$$

while the angular acceleration, for any arbitrarily assigned $\dot{\kappa}_A$, coincides with the assigned $\dot{\omega}_A$ as follows:

$$\dot{\omega}(0) = v^2(0)\dot{\kappa}(0) + \dot{v}(0)\kappa(0) = v_A^2\dot{\kappa}_A + \dot{v}_A\kappa_A = \dot{v}_A\frac{\dot{\omega}_A}{\dot{v}_A} = \dot{\omega}_A.$$

Further considering the FM, and taking into account (36), the final angular velocity is correctly [see statement 3) of Section IV]

$$\omega(t_f) = v(t_f)\kappa(t_f) = v_B\kappa_B = v_B\frac{\dot{\omega}_B}{\dot{v}_B} = 0$$

while the angular acceleration, for any arbitrarily assigned $\dot{\kappa}_B$, coincides with the assigned $\dot{\omega}_B$ as follows:

$$\begin{aligned}\dot{\omega}(t_f) &= v^2(t_f)\dot{\kappa}(t_f) + \dot{v}(t_f)\kappa(t_f)\\ &= v^2(t_f)\dot{\kappa}\left(f(u_1)\right) + \dot{v}(t_f)\kappa\left(f(u_1)\right)\\ &= v_B^2\dot{\kappa}_B + \dot{v}_B\kappa_B\\ &= \dot{v}_B\frac{\dot{\omega}_B}{\dot{v}_B}\\ &= \dot{\omega}_B.\end{aligned}$$

The same reasonings are applicable to the BM case.

Thus, it is possible to conclude that the proposed control law makes it possible to steer the UMR from any feasible extended state A to any feasible extended state B while the resulting motion path coincides with the planned G^3-path. ■

Obviously, if the extended state A, used to replan the trajectory, coincides with the current extended state of the system, the C^1 continuity of the command signals is guaranteed also at the updating time. Thus, the composite command signals are globally C^1 and the piecewise path resulting from this iterative steering approach is globally a G^3-path.

V. Motion Generation Example

Consider that, at time 0, the extended state of the UMR is $A = \{[2\ 1]^T, \pi/4, 0, 0, 0, 0\}$. The desired future extended state at the chosen time $t_f = 4$ s is $B = \{[4\ 3]^T, -\pi/6, 0.5, 0, -0.5, 0.05\}$. In A and B, the Cartesian coordinates are expressed in meters, the angles in radians, the (angular) velocities in (radians/s) m/s, and the (angular) accelerations in (radians/s^2) m/s^2. These extended states satisfy all the assumptions 1)–12) (see Section IV) so that the steps of the inversion algorithm can be directly applied.

Step 1: According to Table I, we plan an FM. Indeed, the logical statement $(v_A = 0) \wedge (v_B > 0) \wedge (\dot{v}_A \geq 0)$ is true.

Step 2: The supervisor has to choose a G^3-curve $\mathbf{p}(u)$ satisfying interpolating conditions that depend on the extended states A and B. First, let us assume $u_0 = 0$ and $u_1 = 1$ for simplicity. Therefore, from (28)–(30), we have

$$\mathbf{p}(0) = \mathbf{p}_A = \begin{bmatrix}2\\1\end{bmatrix}$$
$$\mathbf{p}(1) = \mathbf{p}_B = \begin{bmatrix}4\\3\end{bmatrix}$$
$$\boldsymbol{\tau}(0) = \begin{bmatrix}\frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2}\end{bmatrix}$$
$$\boldsymbol{\tau}(1) = \begin{bmatrix}\frac{\sqrt{3}}{2}\\ -\frac{1}{2}\end{bmatrix}$$

Because the extended state A is a critical case with $v_A = 0$, $\dot{v}_A = 0$, $\omega_A = 0$, and $\dot{\omega}_A = 0$, we can arbitrarily choose κ_A and $\dot{\kappa}_A$, for example, $\kappa_A = 1$ and $\dot{\kappa}_A = 0$. Consequently, from (37) and (39), we obtain

$$\kappa_c(0) = 1, \quad \dot{\kappa}_c(0) = 0.$$

On the other hand, from (32) and (34), we compute

$$\kappa_B = -1, \quad \dot{\kappa}_B = 0.2,$$

and, eventually, from (38) and (40)

$$\kappa_c(1) = -1, \quad \dot{\kappa}_c(1) = 0.2\,\|\dot{\mathbf{p}}(1)\|.$$

TABLE II
Coefficients of the Polynomial G^3-Curve

i	0	1	2	3	4	5	6	7
x_i	2.00	2.33	-3.85	0.00	4.75	11.37	-20.61	8.00
y_i	1.00	2.33	3.85	0.00	-15.04	18.79	-10.07	2.13

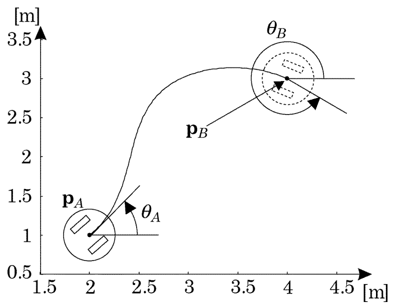

Fig. 2. G^3-path planning example.

TABLE III
Coefficients of the Polynomial Velocity Function $v(t)$ and Corresponding Traveling Times

i	a_{1i}	a_{2i}	a_{3i}	h_i
1	0	0	0.2719	0.8034
2	0.5264	0.6552	-0.1848	0.8000
3	1.2198	0.2116	-0.1134	0.7966
4	1.3412	-0.0593	-0.1820	0.8000
5	0.8969	-0.4961	0.2067	0.8000

The above interpolating conditions are then applied to a seventh-order polynomial curve $\mathbf{p}(u)$ where

$$\begin{aligned}\alpha(u) &:= x_0 + x_1u + x_2u^2 + x_3u^3 + x_4u^4\\ &\quad + x_5u^5 + x_6u^6 + x_7u^7\\ \beta(u) &:= y_0 + y_1u + y_2u^2 + y_3u^3 + y_4u^4\\ &\quad + y_5u^5 + y_6u^6 + y_7u^7.\end{aligned}$$

The coefficients x_i and y_i, listed in Table II, are deduced following the approach proposed in [10]. The resulting G^3-path connecting $\mathbf{p}_A$ with $\mathbf{p}_B$ is plotted in Fig. 2. The total path length is $f(u_1) = 3.3856$ m.

Step 3: The command signal $v(\cdot) \in C^1([0,4])$ is made of five properly joined polynomial curves (46), according to the approach proposed in [18]. The splines coefficients are shown in Table III together with the traveling time of each single curve. The overall velocity function is C^1 and positive for any $t \in (0,4)$. Moreover, it satisfies the boundary conditions (41)–(45).

Step 4: Taking into account both (47) and (6), the command angular velocity $\omega(\cdot) \in C^1([0,4])$ can be numerically computed as

$$\omega(t) = v(t)\kappa_c\left(f^{-1}(s)\right)\Big|_{s=\int_0^t v(\xi)d\xi}. \tag{52}$$

The control inputs $v(t)$ and $\omega(t)$ are plotted in Fig. 3.

It is worth noting that the supervisor can decide to do a path replanning before the running commands (46) and (52) are completed. There may be a variety of reasons to perform an early replanning, for example,

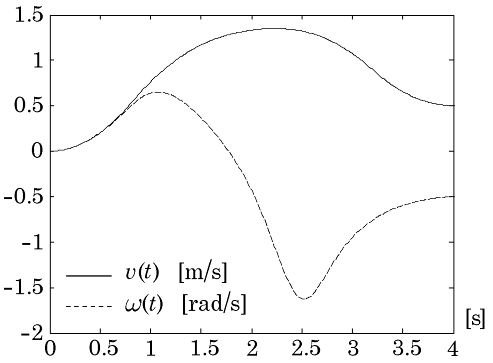

Fig. 3. Command signals $v(t)$ and $\omega(t)$.

to adjust for a sudden obstacle interfering with the robot's motion or to correct for an increasing path following error due to the mismodeling determined by (7)–(9) with respect to the robot actual behavior.

VI. Conclusion

In this paper, a new steering method that guarantees continuous-acceleration control inputs has been proposed for unicycle wheeled mobile robots. This approach relies on a path planning with third-order geometric continuity and on a dynamic path-inversion algorithm.

The exposed method is well suited to be implemented using an iterative steering strategy issued by a supervisory control system to perform sensor-based autonomous navigation. The design of such a supervisor is indeed a promising possible direction for future research work.

References

[1] L. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, pp. 497–517, 1957.

[2] J. Reeds and R. Shepp, "Optimal paths for a car that goes both forward and backward," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.

[3] J. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*, J.-P. Laumond, Ed. Berlin, Germany: Springer, 1998, pp. 1–53.

[4] W. Nelson, "Continuous-curvature paths for autonomous vehicles," in *Proc. IEEE Conf. Robotics and Automation*, vol. 3, May 1989, pp. 1260–1264.

[5] Y. Kanayama and B. Hartman, "Smooth local path planning for autonomous vehicles," in *Proc. IEEE Int. Conf. Robotics and Automation, ICRA89*, vol. 3, Scottsdale, AZ, May 1989, pp. 1265–1270.

[6] K. Komoriya and K. Tanie, "Trajectory design and control of a wheeled-type mobile robot using b-spline curve," in *Proc. IEEE/RSJ Int. Workshop Intelligent Robots and Systems, IROS89*, Tsukuba, Japan, Sept. 1989, pp. 398–405.

[7] H. Delingette, M. Hébert, and K. Ikeuchi, "Trajectory generation with curvature constraint based on energy minimization," in *Proc. IEEE-RSJ Int. Conf. Intelligent Robots and Systems*, Osaka, Japan, Nov. 1991, pp. 206–211.

[8] S. Fleury, P. Souères, J.-P. Laumond, and R. Chatila, "Primitives for smoothing paths of mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, Atlanta, GA, Sept. 1993, pp. 832–839.

[9] A. Piazzi, C. Guarino Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic G^2-splines for the iterative steering of vision-based autonomous vehicles," *IEEE Trans. Intell. Transport. Syst.*, vol. 3, pp. 27–36, Mar. 2002.

[10] A. Piazzi, M. Romano, and C. Guarino Lo Bianco, "G^3-splines for the path planning of wheeled mobile robots," in *Proc. 2003 Eur. Control Conf. ECC 2003*, Cambridge, U.K., Sept. 2003.

[11] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: introductory theory and examples," *Int. J. Contr.*, vol. 61, no. 6, pp. 1327–1361, 1995.

[12] P. Lucibello and G. Oriolo, "Stabilization via iterative state steering with application to chained-form systems," in *Proc. 35th IEEE Conf. Decision and Control*, vol. 3, Kobe, Japan, Dec. 1996, pp. 2614–2619.

[13] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*, J.-P. Laumond, Ed. Berlin, Germany: Springer, 1998, pp. 171–253.

[14] C.-C. Hsiung, *A First Course in Differential Geometry*. Cambrige, MA: International Press, 1997.

[15] B. A. Barsky and J. C. Beatty, "Local control of bias and tension in beta-spline," *Computer Graph.*, vol. 17, no. 3, pp. 193–218, 1983.

[16] J. Peters, "Geometric continuity," in *Handbook of Computer Aided Geometric Design*, G. Farin, J. Hoschek, and M.-S. Kim, Eds. Amsterdam, The Netherlands: North-Holland, 2002, pp. 193–229.

[17] L. de Souza and M. Veloso, "AI planning in supervisory control systems," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 4, Oct. 1996, pp. 3153–3158.

[18] C. Guarino Lo Bianco, A. Piazzi, and M. Romano, "Velocity planning for autonomous vehicles," in *Proc. IEEE Intelligent Vehicles Symp. IV2004*, Parma, Italy, June 14–17, 2004, pp. 413–418.

A Timing Model for Vision-Based Control of Industrial Robot Manipulators

Yanfei Liu, Adam W. Hoover, and Ian D. Walker

***Abstract*—Visual sensing for robotics has been around for decades, but our understanding of a timing model remains crude. By timing model, we refer to the delays (processing lag and motion lag) between "reality" (when a part is sensed), through data processing (the processing of image data to determine part position and orientation), through control (the computation and initiation of robot motion), through "arrival" (when the robot reaches the commanded goal). In this study, we introduce a timing model where sensing and control operate asynchronously. We apply this model to a robotic workcell consisting of a Stäubli RX-130 industrial robot manipulator, a network of six cameras for sensing, and an off-the-shelf Adept MV-19 controller. We present experiments to demonstrate how the model can be applied.**

***Index Terms*—Timing model, visual servoing, workcell.**

I. Introduction

Fig. 1 shows the classic structure for a visual servoing system [1]. In this structure, a camera is used in the feedback loop. It provides feedback on the *actual* position of something being controlled, for example, a robot. This structure can be applied to a variety of systems, including eye-in-hand systems, part-in-hand systems, and mobile robot systems.

In an eye-in-hand system [2]–[6], the camera is mounted on the end-effector of a robot and the control is adjusted to obtain the desired appearance of an object or feature in the camera. Gangloff [2] developed a visual servoing system for a six–degree–of–freedom (DOF)

Manuscript received July 3, 2003; revised January 22, 2004. This paper was recommended by Associate Editor Y.-H. Liu and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported by the South Carolina Commission on Higher Education and the U.S. Office of Naval Research. This paper was presented in part at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, October 2003.

The authors are with the Electrical and Computer Engineering Department, Clemson University, Clemson, SC 29634-0915 USA (e-mail: lyanfei@clemson.edu; ahoover@clemson.edu; iwalker@clemson.edu).

Digital Object Identifier 10.1109/TRO.2004.829460

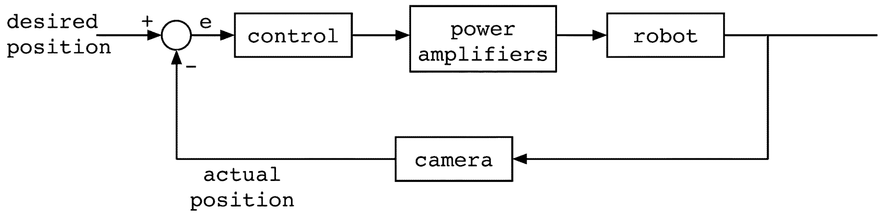

Fig. 1. Classical visual servoing structure.

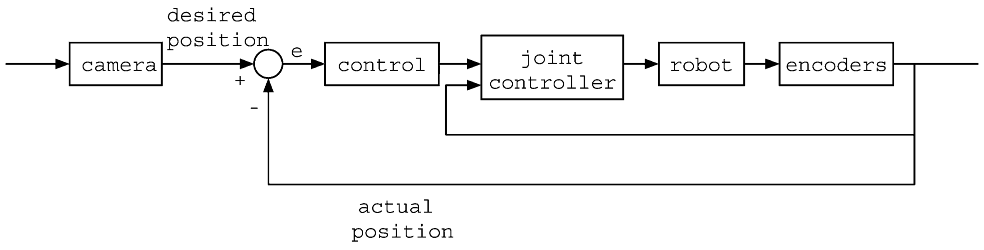

Fig. 2. Vision-guided control structure.

manipulator to follow a class of unknown but structured three-dimensional (3-D) profiles. Papanikolopoulos [3] presented algorithms to allow an eye-in-hand manipulator to track unknown-shaped 3-D objects moving in two-dimensional (2-D) space. The object's trajectory was either a line or an arc. Hashimoto [4] proposed a visual feedback control strategy for an eye-in-hand manipulator to follow a circular trajectory. Corke [5], [6] presented a visual feedforward controller for an eye-in-hand manipulator to fixate on a ping-pong ball thrown across the system's field of view.

In a part-in-hand system [7], the camera is fixed in a position to observe a part which is grasped by a robot. The robot is controlled to move the part to some desired position. For example, Stavnitzky [7] built a system to let the robot align a metal part with another fixed part. Since the part is always grasped by the manipulator, we can also say that the part is something being controlled. In other words, we can say that how the object appears in the camera is controlled.

In some mobile robot problems [8], the camera is mounted over an environment to sense the actual position of the mobile robot as feedback to the controller. For example, Kim [8] built a mobile robot system to play soccer. The camera is fixed over the field and acts as a feedback position sensor. Here, the camera observes something which is directly controlled.

All of these systems, regardless of where the camera is mounted, use the camera in the same control structure. In each case, the system regulates how the object appears in the camera.

In this letter, we consider the problem where the camera is used to provide the *desired* or *reference* position to the internal robot controller. Fig. 2 shows the structure for this system. There are several types of problems that fit this kind of system, where the object of interest cannot be controlled directly. For example, imagine a robot trying to pick up live chickens or a robot trying to manipulate parts hanging on a swaying chain conveyor. Similar problems have been investigated in some works. Houshangi [9] developed a robot manipulator system to grasp a moving cylindrical object. The motion of the object is smooth and can be described by an autoregressive (AR) model. Allen [10] demonstrated a PUMA-560 tracking and grasping a moving model train which moved around a circular railway. Miyazaki [11] built a pingpong robot to accomplish the pingpong task based on virtual targets. Nakai [12] developed a robot system to play volleyball with human beings. From the above systems, we notice that the motion of the object was limited to a known class of trajectories. In this letter, we seek to extend this to let the robot follow an unstructured (completely unknown) trajectory. This will be enabled in part by providing a generic timing model for this kind of system.

Our timing model considers the problem where image processing and control happen asynchronously. The task of the robot is to intercept moving objects in real time under the constraints of asynchronous vision and control. We are faced with the following three problems.

1) The maximum possible rate for complex visual sensing and processing is much slower than the minimum required rate for mechanical control.
2) The time required for visual processing introduces a significant lag between when the object state in reality is sensed and when the visual understanding of that object state (e.g., image tracking result) is available. We call this the *processing lag*.
3) The slow rate of update for visual feedback results in larger desired motions between updates, producing a lag in when the mechanical system completes the desired motion. We call this the *motion lag*.

Consider problem 1). A standard closed-loop control algorithm assumes that new data can be sensed on each iteration of control. Common industrial cameras operate at 30 Hz while common control algorithms can become unstable at rates less than several hundred Hertz. Complex image processing tasks, such as segmentation, pose estimation, and feature matching, typically run even slower than 30 Hz, while control problems can require rates as high as 1 kHz. In general, this gap in rates will not be solved by the trend of increasing computational power (Moore's Law). As this power increases, so will the amount of desired visual processing, and so will the complexity of the control problem. In this letter, we propose to address this

TABLE I
SUMMARY OF RELATED WORK

work	image processing rate (HZ)	control rate (HZ)	processing lag (ms)	motion lag (ms)
Papanikolopoulos et. al. [3]	10	300	100	–
Hashimoto et. al. [4]	4	1000	–	–
Corke and Good [5], [6]	50	70	48	–
Stavnitzky and Capson [7]	30	1000	–	–
Kim et. al. [8]	30	30	90	–
Houshangi [9]	5	36	196	–
Allen et. al. [10]	10	50	100	–
Miyazaki et. al. [11]	60	–	–	–
Nakai et. al. [12]	60	500	–	–
this work	23	250	151	130

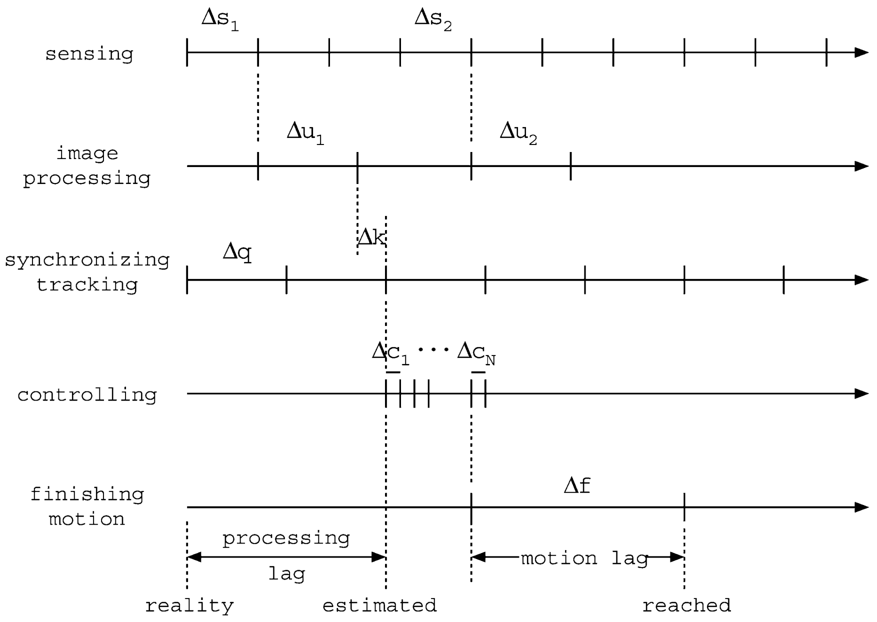

Fig. 3. Timing model for estimating the lag and latency.

problem directly by modeling the visual sensing, processing, and control processes as having fundamentally different rates, where the sensing and processing are at least one order of magnitude slower than the control.

Problem 2) is a consequence of the complexity of the image processing operations. There is always a lag (processing lag) between reality and when the result from processing a measurement of the object state is available. In a high-speed (e.g., 1 kHz) closed-loop control, this lag can usually be ignored. However, as the processing complexity increases, a nonnegligible lag is introduced between when the image was acquired (the object state in reality) and when the image processing result is available (e.g., estimate of object position). We incorporate an estimate of the processing lag directly into our timing model.

Problem 3) is also a consequence of the slow rate of visual sensing and processing. In a high-speed closed-loop control, the motion executed between iterations is expected to be small enough to be completed during the iteration. The motion lag (time it takes to complete the motion) is considered negligible. But, as the sensing rate slows, the tracked object moves farther between iterations, requiring the mechanical system (e.g., robot) to also move farther between iterations. As a consequence, it is possible to have a system that has not completed the desired set point motion prior to the next iteration of control. We address this problem by directly incorporating an estimate of the motion lag into our timing model.

Table I presents a summary of how previous works have featured and addressed these three problems. From this table, we note that the first two of the three problems have been addressed to some extent in previous works. However, no work appears to have explicitly considered problem 3). All of these works neglect the motion time (motion lag) of the robot. One work [10] noted this problem and used an $\alpha - \beta - \gamma$ predictor to compensate for it instead of explicitly modeling it. None of these works has considered the generic modeling of this type of system.

Some works synchronized the image processing rate and control frequency for a more traditional solution. In [2], the frequency of visual sensing, processing and control were all set to 50 Hz. Basically, the control frequency was synchronized to the image processing rate for simplicity. Simulation results of high frequency control, i.e., 500 Hz, were also shown in [2]. Performance of the high-frequency controller was, as expected, better than the low-frequency version, motivating a more thorough investigation of a generic timing model to solve the

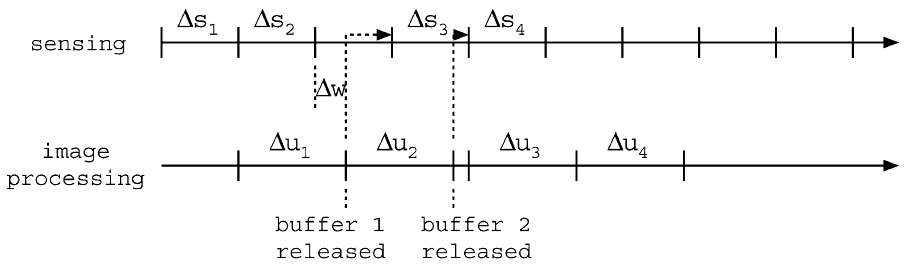

Fig. 4. Timing model using double buffering for processing image data.

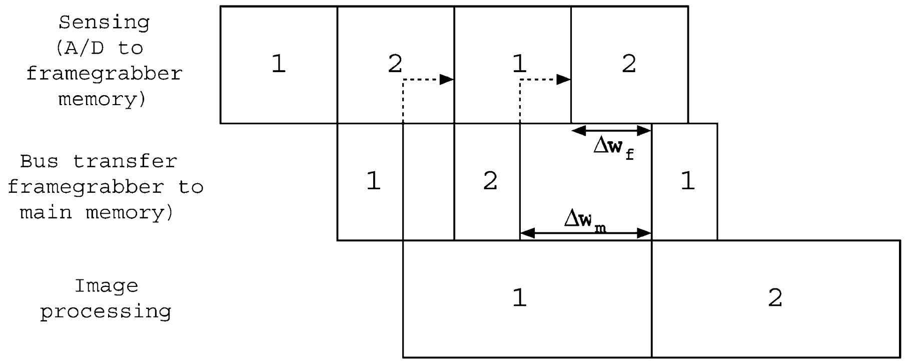
Sensing (A/D to framegrabber memory)
Bus transfer framegrabber to main memory)
Image processing
Δw_f
Δw_m

Fig. 5. Timing model of using consecutive double buffering.

problem. Corke [6] and Kim [8] presented timing diagrams to describe the time delay. Based on the timing diagrams, these works tried to use discrete time models to model the systems. In order to do this, the authors simplify these asynchronous systems to single-rate systems. It is well known that the discrete time model can only be applied into single-rate systems or systems where the control rate and the vision sensing rate are very close. However, from Table I, we notice that most real systems do not satisfy this condition.

Therefore, in this letter, we propose a continuous generic timing model to describe asynchronous vision-based control systems. This letter is the first to explicitly model the motion lag of the robot and presents a general timing model for vision-based robotic systems. The approach is focused on improving real-time trajectory generation based on vision (Fig. 2) and is independent of the control strategy applied.

The remainder of this letter is organized as follows. In Section II, we describe our generic timing model and then apply this model to an industrial robot testbed that uses a network of cameras to track objects in its workcell. In Section III, we demonstrate the importance of the application of our model by using it to derive a "lunge" expression that lets the robot intercept an object moving in an unknown trajectory. Finally, we conclude the letter in Section IV.

II. Methods

Fig. 3 illustrates our timing model. From top to bottom, each line shows a component of the system in process order (e.g., sensing comes before image processing). The horizontal axis represents time. We use this model to quantify the processing lag and motion lag of the system. The processing lag is the time between reality and when an estimate of the object state in reality is available. Similarly, the motion lag is the time between when the control command is issued and when the mechanical system finishes the motion.

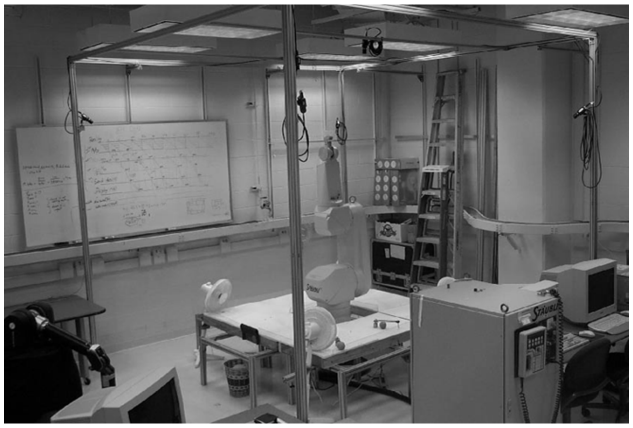

Fig. 6. Our prototype dynamic workcell.

The sensing and control processes operate at fixed intervals Δs and Δc, where $\Delta s > \Delta c$ (sensing is slower than control). The time required for all image processing and tracking operations is designated Δu. This processing starts when an input buffer is filled with image data (on a clock or sync signal defined by the sensing line). An input buffer cannot be filled with new image data until the processing of the previous image data in that buffer is completed. In Fig. 3, this is why Δs_2 starts on the next sync after the end of Δu_1.

Fig. 3 depicts the case where $\Delta u > \Delta s$ (the processing takes longer than the sensing interval) and when there is only one image buffer. Fig. 4 depicts the case where two input buffers are used (commonly called "double buffering"). In this case, a second image buffer is being filled while the image data in the first buffer is being processed. Double

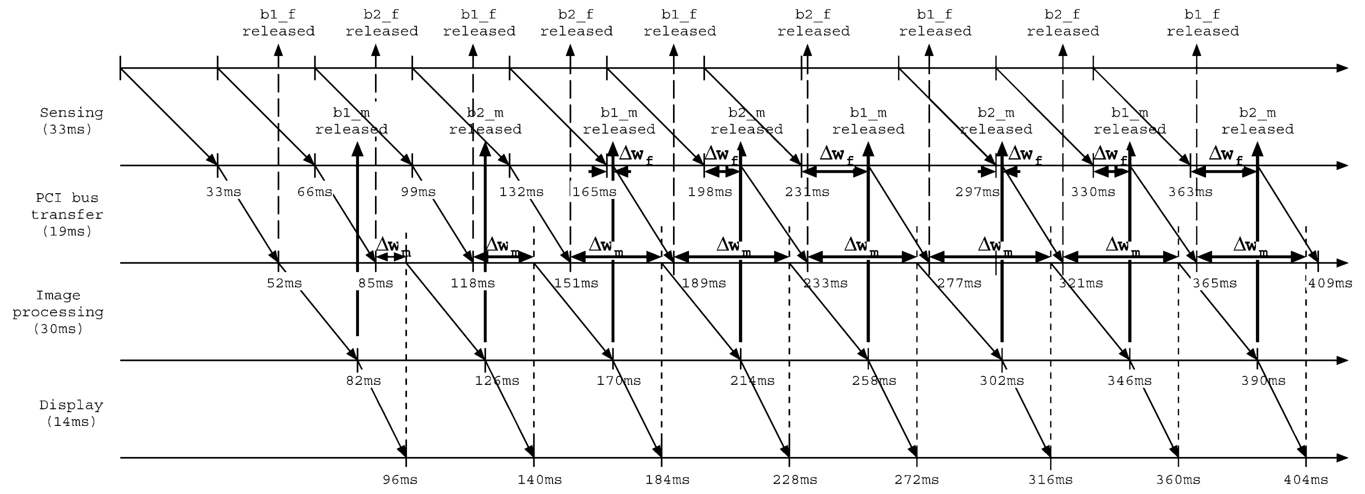

Fig. 7. Information flow through system.

buffering increases the lag (note the extra time Δw between the end of Δs_2 and the start of Δu_2) but increases the throughput (note the greater number of images processed in Fig. 4 as compared to Fig. 3).

Fig. 5 depicts the even more complicated case where double buffering happens consecutively. For example, a framegrabber can be equipped with enough memory to double buffer images on the framegrabber itself as they are digitized. This double buffer can then feed a second double buffer residing in the main (host) memory. Although this again increases throughput, Fig. 5 shows how it also increases the lag. Each box indicates the time spent by an image at each stage. The interval Δw_m is the time an image spends waiting in host memory for the completion of processing of the previous image. This is similar to the term Δw in Fig. 4. The interval Δw_f is the time an image spends waiting in the framegrabber for a host buffer to become available. In this case, the image on the framegrabber is waiting for the host to finish processing the previous image residing in the buffer needed for transfer of the new image.

In all of these cases, we have assumed that the processing takes longer than the sensing interval ($\Delta u > \Delta s$). In the case where $\Delta u < \Delta s$ (the processing is faster than the sensing rate), double buffering makes it possible to process every image. In any case, there is always a minimum lag of $\Delta s + \Delta u$, but, depending on the buffering used and the relation of Δs to Δu, the lag can be larger.

In order to handle all of these cases, we introduce a synchronous tracking process (line 3 in Fig. 3) operating at a rate of Δq. The tracking line takes the most recent result from the image processing line and updates it for any additional delay (depicted as Δk) using a Kalman filter. In general, we desire $\Delta q \approx \Delta u$ so that tracking is updated approximately as fast as new results become available. A secondary benefit of the synchronous tracking line is that it satisfies standard control algorithm requirements that assume synchronous input data. Without this line, the results from image processing can arrive asynchronously (as in Figs. 4 and 5).

The fourth and fifth lines in Fig. 3 represent the control process and completion of motion. We consider the case where the distance traveled by an object between tracking updates is larger than a robot could safely or smoothly move during a single iteration of control. The control is therefore broken up into a series of N submotion commands occurring at a rate of Δc. Additionally, we expect the motion requested by any new iteration of control to take Δf time to complete. Fig. 3 depicts the case where control commands are cumulative (each new control command is relative to the last commanded goal). In Section II-A, we describe our prototype, which uses an off-the-shelf Adept controller that operates in this manner. For this controller, the motion is completed some time Δf after the last given control command. It is of course possible to have an open architecture controller that finishes the motion just prior to the next iteration of control. In this case, $\Delta f \approx \Delta c$.

Once values are known for the variables Δs, Δu, Δq, Δc, and Δf, it is possible to derive various expressions for controlling a robot to solve specific problems, for example, to intercept a moving object. In the next section, we describe our prototype workcell and derivation of the timing variables. In Section III, we derive an expression for implementing a "lunge" of the robot to intercept an object moving with an *a priori* unknown trajectory.

A. Prototype

Fig. 6 shows a picture of our prototype workcell for this project. We use a Stäubli RX130 manipulator with its conventional controller, the Adept Corporation model MV-19. A network of six cameras surrounds the workcell, placed on a cube of aluminum framing. The cameras are wired to two imaging technology PC-RGB framegrabbers (A/D video converters) mounted in a Compaq Proliant 8500 computer. The Compaq has a standard SMP (system multiprocessor) architecture equipped with eight 550-MHz Intel Pentium 3-Xeon processors. In [14], we detailed the workcell configuration, calibration, image differencing and real-time robot motion planning. In [15], we presented some tracking experiments to show that our system can track different kinds of objects using continuous visual sensing.

Fig. 7 shows a timing diagram describing the flow of information through the system, along with how long each step takes. Some of these estimates were derived analytically from knowledge of the hardware, while other estimates were derived from measurements taken while the system was operating. We will discuss each part in detail in the following paragraphs.

Fig. 7 starts with an event that is happening in real time (e.g., an object moves). The cameras operate at 30 Hz using the standard National Television System Committee (NTSC) format, so that for this system $\Delta s = 33$ ms. The object state in reality that is imaged takes 33 ms to transfer from the camera to the framegrabber. The six cameras are synchronized on a common clock for the vertical sync (start of image).

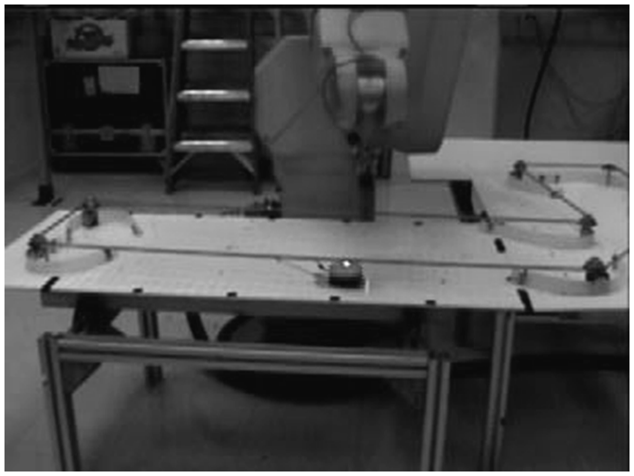

Fig. 8. Experimental setup for lag measurement.

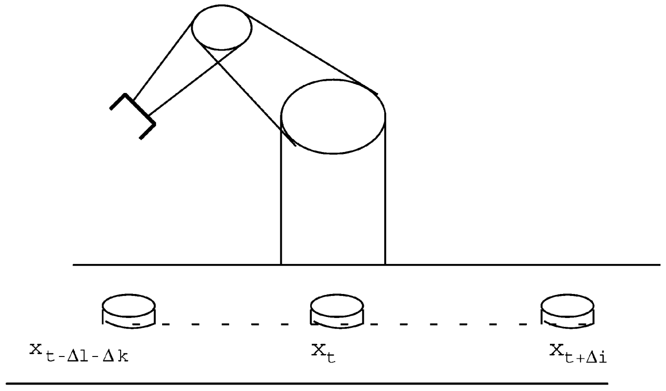

Fig. 9. Scenario for intercepting objects.

Each camera is greyscale, so that three cameras may be wired to the red, green, and blue components of an RGB input on a framegrabber. After 33 ms, all six images are digitized and residing in framegrabber memory.

The framegrabbers have microcontrollers that can operate as peripheral component interconnect (PCI) bus masters, initiating transfers from framegrabber memory to main memory. The Compaq is programmed to use a ring buffer (with room to hold two sets of six images) to facilitate double buffering. While one set of six images is being processed, a second set can be transferred from framegrabber memory to main memory at the same time. Assuming negligible traffic on the PCI bus, the time for this transfer can be computed as the total number of image bytes divided by the bandwidth of the bus: $(640 \times 480 \times 4 \times 2 \text{ bytes})/(4 \text{ bytes} \times 33 \text{ MHz}) = 19$ ms, where the bandwidth is the theoretical maximum provided by the 32-b 33-MHz PCI standard.

The image processing portion of our system creates a 2-D occupancy map of the space in a horizontal plane of interest in the workcell, locates the centroid of an object in this space, and displays the result on-screen [16]. Based upon empirical measurements of the run-time of the image processing methods on the Compaq, we observed them to take approximately 30 ms on average each iteration. This time can vary by approximately ± 2 ms depending upon the content of the images. We discuss the variances of our measurements in more detail at the end of this section. We also measured that the image display takes 14 ms on average. Therefore, the total processing time Δu for this system is $19 + 30 + 14 = 63$ ms.

The images may wait in buffers before being processed, due to our use of consecutive double buffering (see Fig. 5). The waiting time can vary depending on the phasing of Δs and Δu, as shown in Fig. 7. We noticed that, after several iterations, the waiting time repeats in a pattern. The main memory waiting time Δw_m becomes a constant (39 ms) after four iterations. From Fig. 7, we can observe that the PCI bus transfer always happens right after image processing finishes. Therefore, the main memory waiting time Δw_m equals the display time plus image processing time, minus PCI bus transfer time, i.e., $14 + 14 + 30 - 19 = 39$ ms. The framegrabber waiting time Δw_f is repeating in three numbers: 5, 16, and 27 ms. So we take the average waiting time $\overline{\Delta w_f}$ as $(5 + 16 + 27)/3 = 16$ ms. Thus, we can get the total average waiting time $\overline{\Delta w} = 39 + 16 = 55$ ms. For our system, the syncing time Δk is a variable that we get in real time. Adding up the appropriate terms $(\Delta s + \Delta u + \overline{\Delta w} + \Delta k)$, the processing lag for this system is $33 + 63 + 55 + \Delta k = (151 + \Delta k)$ ms. To unify the terms, we use Δl to express the computable partial lag time (151 ms).

After synchronization, the state of the object is sent through a 10-Mb ethernet link from the Compaq to the Adept. Based on empirical measurements, these operations were observed to collectively take less than 1 ms. For this system, we set $\Delta q = 40$ ms which is near but slightly under the occupancy map time plus the image display time $(30 + 14$ ms).

At this point, the Adept (robot) has a new goal. This goal is directly forwarded to the motor-level controller through the "Alter" command [13]. According to the manufacturer (Adept), the maximum issue rate for the Alter command is 500 Hz (once every 2 ms), but through experimentation we observed that this rate could not always be maintained. Therefore, we set $\Delta c = 4$ ms. The precise details of the motor-level controller are proprietary to Adept Corporation and could not be determined. Therefore, we determined Δf empirically through repeated measurements of the time it took to complete an Alter command. We observed that the time vary from 120 to 150 ms, with a commonly occurring median value of 130 ms. We therefore set Δf to be 130 ms.

In our model, we assume constants for most of the lag terms. Our estimates for all parameters, except for occupancy map computation time, display time, and robot motion lag, were generated via analysis of models based on known timing constraints, as discussed above. The remaining three parameters listed above required information not directly available to us and were obtained from averages via empirical measurement. It is important to note that all of these terms have variances, some of them having appreciable size in this context (more than 1 ms). For the sensing and processing terms, it would be ideal to timestamp each image upon acquisition and measure the terms precisely. However, in order to solve problems involving estimates into the future (for example to plan a catch of an object), it is necessary to have averages. Therefore, we only note the variances here and leave a more thorough understanding of their effect to future work.

In order to validate our estimate of total lag for our system, we conducted a series of experiments. Fig. 8 shows a picture of the experimental setup. We set up another camera, completely external to our system, to observe its operation. A conveyor was constructed to move in a constant path at a constant velocity. A light bulb was attached to the conveyor and the system was directed to track the object, keeping the robot positioned above the light bulb. A small laser was mounted in the end effector of the robot, so that it pointed straight down onto

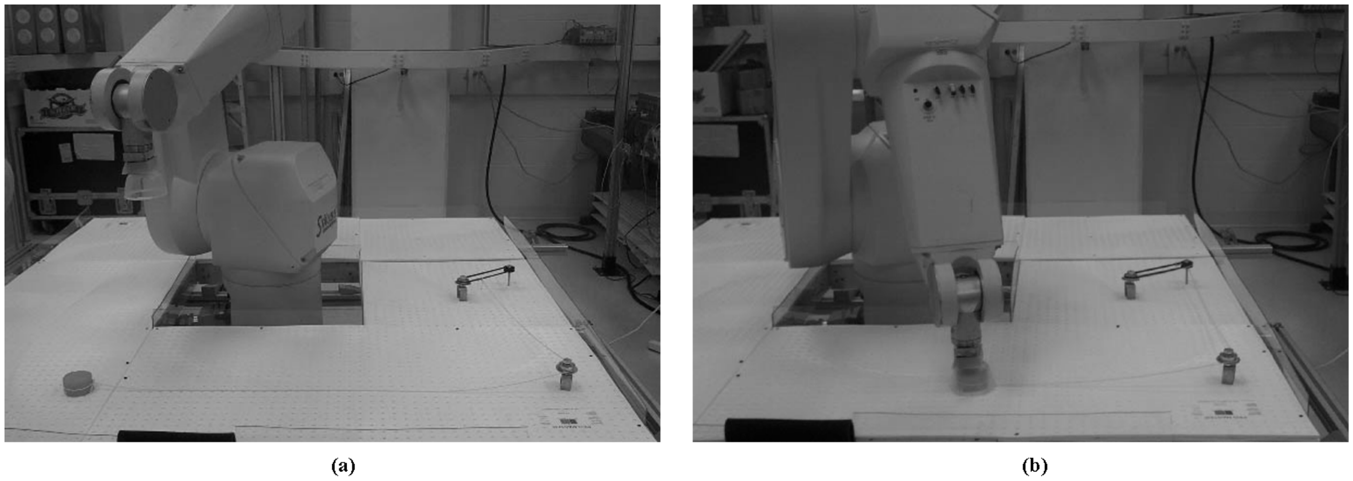

Fig. 10. Experimental setup for catching the moving object. (a) Initial position. (b) Impact scene.

TABLE II
EXPERIMENTAL RESULTS FOR CATCHING THE MOVING OBJECT

$\Delta q = 40$			$\Delta q = 80$		
velocity (mm/s)	stdev (mm/s)	catch percentage	velocity (mm/s)	stdev (mm/s)	catch percentage
84.4 – 97.4	1.3 – 3.8	100%	85.9 – 95.1	2.5 – 3.7	100%
129.8 – 146.7	1.7 – 3.2	100%	126.1 – 137.7	1.7 – 3.3	100%
177.6 – 195.1	0.5 – 2.6	100%	175.8 – 192.8	1.1 – 2.7	100%

the conveyor. The experiment was configured to make it possible for the external camera to estimate the distance between the light bulb and the footprint of the laser on the conveyor. Knowing the velocity of the light bulb, we were able to empirically measure the total lag of our system, and verify that it matched the estimate derived from our timing model. Complete details of this experiment can be found in a technical report[17].

III. EXPERIMENTS

In order to test our methods, we experiment with the problem of catching a moving object. Fig. 9 depicts the scenario. The object is moving at an unknown constant velocity in a straight line. In this example, the object is moving in one dimension; however, we formulate the solution using vectors to indicate that the solution is also applicable to 2-D and 3-D problems. Due to the processing lag, the most recently measured position of the object is where the object was $(\Delta l + \Delta k)$ ms previously. We denote this location as $\vec{x}_{t-\Delta l-\Delta k}$. The current position of the object, i.e., the time when the robot starts to lunge toward the object, is denoted as $\vec{x}_t$. The current velocity of the object is denoted as $\vec{v}_t$, and is assumed to be equal to the last measured velocity $\vec{v}_{t-\Delta l-\Delta k}$. Therefore, the relationship between $\vec{x}_t$ and $\vec{x}_{t-\Delta l-\Delta k}$ is described in the following equation:

$$\vec{x}_t = \vec{x}_{t-\Delta l-\Delta k} + \vec{v}_{t-\Delta l-\Delta k}(\Delta l + \Delta k). \tag{1}$$

It will take some amount of time Δi ms for the robot to reach the point of impact where it will catch the object. We denote the impact location as $\vec{x}_{t+\Delta i}$.

There is more than one way to consider implementing a catch. One approach is to keep the robot's position at the most recently measured position $(\vec{x}_{t-\Delta l-\Delta k})$ and then lunge to the final impact position. The advantage to this approach is that the robot is only moving to locations where the object has actually traveled. The disadvantage to this approach is that the robot must lunge a distance that covers the lag time plus the time to impact. A second approach is to keep the robot's position at the current position $(\vec{x}_t)$ or even keep the robot's position near the predicted impact position $(\vec{x}_{t+\Delta i})$. If the object is moving at a constant velocity or in a fixed pattern, this approach will not suffer from misprediction and will always decrease the lunge distance. However, if the object motion is not so controlled, this approach could cause unfavorable behavior, for example increasing the lunge distance in the case where the object reverses direction. In either approach, we can describe the intercept problem as the following: if the robot desires to intercept the object at time t while following the object, how many control commands (N) should be issued between time t and the time when the robot intercepts the object, and what is the constant distance $(\Delta\vec{d})$ that each single control command should move.

For some problems, N is fixed. The solution for a fixed N problem involves only one equation

$$\vec{x}_{t+\Delta i}[i] = \vec{x}_t[i] + \vec{v}_t[i](\Delta c \times (N-1) + \Delta f). \tag{2}$$

For our problem, N is varying. Suppose that $\vec{x}_{t-\Delta q}$ is the position where the robot was last commanded to move to, i.e., the position where the object is at time $t - \Delta q, \vec{n}$ is the number of alters which will be executed, and d is the maximum distance a single alter can move. The solution now involves the following two equations:

$$\vec{x}_{t+\Delta i}[i] = \vec{x}_t[i] + \vec{v}_t[i](\Delta c \times (\vec{n}[i] - 1) + \Delta f) \tag{3}$$

$$|\vec{x}_{t+\Delta i}[i] - \vec{x}_{t-\Delta q}[i]| = \vec{n}[i] \times d. \tag{4}$$

Combining (3) and (4), based on the assumption that the object does not change velocity direction between $t \ldots t - \Delta q$, we obtain

$$\vec{n}[i] = \left\lceil \frac{(\Delta f - \Delta c)|\vec{v}_t[i]| + |\vec{x}_t[i] - \vec{x}_{t-\Delta q}[i]|}{d - \Delta c|\vec{v}_t[i]|} \right\rceil. \tag{5}$$

When we solve for $\vec{n}$, the following constraint exists:

$$|\vec{v}_t[i]| < \frac{d}{\Delta c}. \tag{6}$$

This equation represents the constraint for successful intercept between the velocity of the moving object and the key parameters in the timing model.

Then N is chosen as the maximum element of vector $\vec{n}$. Therefore

$$\Delta\vec{d}[i] = \frac{\vec{x}_{t+\Delta i}[i] - \vec{x}_{t-\Delta q}[i]}{N}. \tag{7}$$

A. Experimental Setup and Results

To verify that our model is effective, we design an experiment to let our industrial manipulator catch a moving object. A small cylindrical object is dragged by a string tied to a belt moving at a constant velocity. The belt is moved by a dc motor. The object moves along a straight line. Fig. 10 shows our experimental setup. The robot follows the object with the end-effector pointing straight down approximately 300 mm above the object. When the robot is commanded to intercept the object, the robot will lunge and cover the object on the table with a modified end-effector, a small plastic bowl. The diameter of the object is 70 mm, the diameter for the small bowl is 90 mm. Therefore, the error should be less than 10 mm on each side in order to successfully catch the object.

In order to test the applicability of our timing model, we conducted two sets of experiments. We set Δq to two different values, 40 and 80, in these two sets of experiments. We varied the voltage of the motor driving the conveyor to let the object move at three different velocities. For each velocity, we kept the voltage of the motor constant to make the object move in a relatively fixed velocity and ran the experiment ten times. Table II shows the results of the experiments. The velocity column is filled with the range of the average velocity in the ten experiments. The standard deviation column is the range of the standard deviation of the velocity of each experiment. The results demonstrate that, if the object moves at a relatively fixed velocity, the robot catches the object 100% of the time independent of the velocity of the object and the position update time (Δq). We have additionally conducted numerous experiments with more unpredictable object trajectories (using fans to blow objects in the workspace in semipredictable ways), and found the results to generalize well. These results will be reported in more detail in future publications.

IV. Conclusion

In this letter, we present a generic timing model for a robotic system using visual sensing, where the camera provides the *desired* position to the robot controller. We demonstrate how to obtain the values of the parameters in the model, using our dynamic workcell as an example. Finally, we show how this timing model can be used to solve problems, using as an example the problem of our industrial robot intercepting a moving object. The results of the experiments show that our model is highly effective and generalizable.

Acknowledgment

The authors would like to thank the Stäubli Corporation, Duncan, SC, for donating in part a state-of-the-art RX-130 industrial manipulator.

References

[1] S. Hutchinson, D. Hager, and P. Corke, “A tutorial on visual servo control,” *IEEE Trans. Robot. Automat.*, vol. 12, pp. 651–670, Oct. 1996.

[2] J. Gangloff and M. F. de Mathelin, “Visual servoing of a 6-DOF manipulator for unknown 3-D profile following,” *IEEE Trans. Robot. Automat.*, vol. 18, pp. 511–520, Aug. 2002.

[3] N. Papanikolopoulos, P. K. Khosla, and T. Kanade, “Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision,” *IEEE Trans. Robot. Automat.*, vol. 9, pp. 14–33, Feb. 1993.

[4] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura, “Manipulator control with image-based visual servo,” in *Proc. IEEE Int. Conf. Robotics and Automation*, Scramento, CA, Apr. 1991, pp. 2267–2272.

[5] P. Corke and M. Good, “Dynamic effects in visual closed-loop systems,” *IEEE Trans. Robot. Automat.*, vol. 12, pp. 671–683, Oct. 1996.

[6] ——, “Dynamic effects in high-performance visual servoing,” in *Proc. IEEE Int. Conf. Robotics and Automation*, Nice, France, May 1992, pp. 1838–1843.

[7] J. Stavnitzky and D. Capson, “Multiple camera model-based 3-D visual servo,” *IEEE Trans. Robot. Automat.*, vol. 16, pp. 732–739, Dec. 2000.

[8] S. H. Kim, J. S. Choi, and B. K. Kim, “Visual servo control algorithm for soccer robots considering time-delay,” *Intell Automation Soft Comput.*, vol. 6, no. 1, pp. 33–43, 2000.

[9] N. Houshangi, “Control of a robotic manipulator to grasp a moving target using vision,” in *Proc. IEEE Int. Conf. Robotics and Automation*, Cincinnati, OH, May 1990, pp. 604–609.

[10] P. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, “Automated tracking and grasping of a moving object with a robotics hand-eye system,” *IEEE Trans. Robot. Automat.*, vol. 9, pp. 152–165, Apr. 1993.

[11] F. Miyazaki, M. Takeuchi, M. Matsushima, T. Kusano, and T. Hashimoto, “Realization of the table tennis task based on virtual targets,” in *Proc. IEEE Int. Conf. Robotics and Automation*, Washington, DC, May 2002, pp. 3844–3849.

[12] H. Nakai, Y. Taniguchi, M. Uenohara, and T. Yoshimi, “A volleyball playing robot,” in *Proc. 1998 IEEE Int. Conf. Robotics and Automation*, Leuven, Belgium, May 1998, pp. 1083–1089.

[13] *V+ User's Manual*. Adept Corporation.

[14] Y. Liu, A. Hoover, and I. Walker, “Sensor network based workcell for industrial robots,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Maui, HI, Oct. 2001, pp. 1434–1439.

[15] ——, “Experiments using a sensor network based workcell for industrial robots,” in *Proc. IEEE Int. Conf. Robotics and Automation*, Washington, DC, May 2002, pp. 2988–2993.

[16] A. Hoover and B. Olsen, “A real-time occupancy map from multiple video streams,” in *Proc. 1999 IEEE Int. Conf. Robotics and Automation*, Detroit, MI, May 1999, pp. 2261–2266.

[17] C. Hermanson, A. Hoover, M. Joseph, B. Judy, Y. Liu, and I. Walker, “A Timing Model for a Dynamic Robotic Workcell,” Dept. Elect. Comput. Eng., Clemson Univ., Tech. Rep., Nov. 2002.

[18] Y. Liu, A. Hoover, I. Walker, B. Judy, M. Joseph, and C. Hermanson, “A new generic model for vision based tracking in robotics systems,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, NV, Oct. 2003, pp. 248–253.