Industrial-Scale Autonomous Wheeled-Vehicle Path Following by Combining Iterative Learning Control with Feedback Linearization*

Lukas G. Dekker¹, Joshua A. Marshall¹,², and Johan Larsson³

Abstract—This paper presents a path following method for autonomous wheeled vehicles that combines iterative learning control (ILC) with nonlinear feedback linearization (FBL) to provide anticipatory control action based on stored path following errors over repeated driving trials. By implementing ILC in a fully feedback-linearized space, control corrections are applied to a transformed input, thus allowing for a single back-computation to the nonlinear vehicle’s control input. Hence, the approach is comparatively easy to implement and also computationally inexpensive. We first outline the mathematical formulation for this control method and then describe field results from tests conducted by using an industrial-scale wheeled underground mining vehicle in a representative environment to demonstrate effectiveness.

I. INTRODUCTION

For autonomous ground vehicle applications a fundamental function is path following, wherein the vehicle is required to track a prescribed path (route) irrespective of time. There exist a number of approaches to this problem, however many have been tested on only small mobile robots in highly idealized environments or show effectiveness for industrial-scale applications through only simplified simulations; e.g., see Juan [1] and Yong et al. [2] for a review. This paper examines a solution to the path following problem for nonlinear wheeled vehicles that combines iterative learning control (ILC) with feedback linearization (FBL) and was found to achieve high-performance results at high speeds (for large hydraulically-driven vehicle), yet is also comparatively easy to implement. We studied the effectiveness of the approach by conducting field experiments using a very large industrial-scale mining vehicle in a realistic operating environment.

We begin with a two-timescale control architecture [3], where path following is executed by an outer-loop control and navigation system and low-level vehicle dynamics (e.g., steering rate and vehicle speed regulation) are carried out by a set of inner-loop controllers. However, if the closed-loop bandwidth of the inner-loop is low—as is common for many large industrial vehicles due to slow actuator and vehicle dynamics—this two-timescale architecture can limit and even destabilize outer-loop (i.e., path following) performance. To partially account for limited steering bandwidth, for instance, Marshall et al. [3, Section 4.2] used a patch that instructs the controller to use heading-error feedback from a point ahead (in time) along the prescribed path. In contrast, the iterative nature of ILC allows for the same anticipatory function by looking ahead on the error signal from the previous iteration in order to apply corrections to the current iteration’s input at that discrete step, similar to the approach used by Wang and Ye [4]. Thus, a simple kinematic model can be used for controller design while vehicle dynamics and environmental disturbances (e.g., wheel slip) are handled by iteratively correcting the steering inputs.

This paper proposes the combination of ILC+FBL for autonomous vehicle path following. Here, ILC+FBL is implemented by performing controller design (for a SISO system) in a fully feedback-linearized error space. This facilitates ILC design by producing a linearized error system that does not change at each discrete time step. Additionally, the ILC implementation is efficient when used in combination with FBL because learned corrections are applied to the transformed control input before conducting only one single back-computation to the nonlinear system control input (in our case, the steering rate). This approach was validated through field trials by using an automated industrial-scale wheeled underground mining vehicle outfitted with teach-and-repeat navigation, localization, and sensor systems similar to the ones described by Marshall et al. [3].

A. Motivation and Related Work

ILC is a relatively well established control methodology and has been successfully implemented for a variety of practical applications. Iterative learning is normally used for systems that repeatedly execute a given task. Thus, it has found most of its applications in industrial manufacturing as described in a survey of ILC designs and applications by Bristow and Tharayil [5]. For example, ILC has been successfully used for motion control of assembly line components, such as chain conveyors [6] and manipulator-type robots [4], [7]–[9]. These examples examine various ILC algorithm types (e.g., P-type, PD-type, and anticipatory designs), however, apply to only stationary robots and systems.

Comparatively, few have worked on the application of ILC to tracking for robotic ground vehicles. However, some successful results have been demonstrated [10]–[14] that imply that ILC is a useful control methodology for mobile robot applications. These examples, however, provide...
It can be shown that a (nonholonomic) kinematic model of the vehicle is given by
\[
\mathbf{q} = \begin{bmatrix} \cos \theta & 0 \\ -\sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},
\]
where \( \ell(\phi) := \ell_F + \ell_R \cos \phi \). For simplicity we write (1) as \( \dot{\mathbf{q}} = \Gamma(\mathbf{q}) \mathbf{u} \), where \( \mathbf{u} = (v, \omega) \) are the rate inputs to the vehicle model. Positive values of \( v \) and \( \omega \) correspond to forward movement and steering to the right, respectively.

B. Path Following Control by FBL

We define two path following errors, as shown in Fig. 3. The lateral error \( \varepsilon_L \) is the distance from \( F \) to the tangent to the desired path and the heading error \( \varepsilon_H \) is the difference between the angle (in the configuration space) of this instantaneous tangent and the vehicle’s heading \( \theta \). These error states \( (\varepsilon_L, \varepsilon_H) \) are repeatedly computed (estimated) by UKF-based navigation algorithms described by Marshall et al. [3].

Without loss of generality, the tangent to the desired path is assumed to correspond (instantaneously) with the \( x \)-axis shown in Fig. 2. Therefore, \( \varepsilon_L = y \) and \( \varepsilon_H = \theta \) and, by using

![Image](image-url)

**Fig. 1.** ST18 underground LHD machine at the Kvarntorp test mine. Note the road surface and tunnel wall conditions, as well as the onboard LiDAR.

**Fig. 2.** LHD vehicle configuration.
the instantaneous rates of change of the errors become

\begin{align}
\dot{e}_L &= v \sin \epsilon_H \\
\dot{e}_H &= -v \sin \phi - \frac{\dot{\ell}_R}{\ell(\phi)} \omega.
\end{align}

(2a) Consider the nonlinear change of coordinates

\begin{align}
z_1 &:= e_L \\
z_2 &:= v \sin \epsilon_H,
\end{align}

which can be derived by well-known techniques for feedback linearization [18]. The error dynamics (2) can then be written as a second-order linear system of the form

\begin{equation}
\begin{bmatrix}
z_1 \\
z_2
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
z_1 \\
z_2
\end{bmatrix} +
\begin{bmatrix}
0 \\
1
\end{bmatrix} \eta,
\end{equation}

where \( \eta \in \mathbb{R} \) is the transformed control input given by

\begin{equation}
\eta = -v^2 \sin \phi \cos \epsilon_H - \frac{\ell_{RV} \cos \epsilon_H}{\ell(\phi)} \omega.
\end{equation}

(5) Therefore, to stabilize the error system (4) we use a PD-type controller where the control input is of the form

\begin{equation}
\eta = Kz =
\begin{bmatrix}
k_P \\
k_D
\end{bmatrix}
\begin{bmatrix}
z_1 \\
z_2
\end{bmatrix},
\end{equation}

which is asymptotically stable for \( k_P, k_D < 0 \).

Finally, the steering input \( \omega \) is back-computed by inverting the transformed input (5) to obtain

\begin{equation}
\omega = -v \sin \phi - \frac{\ell(\phi)}{\ell_{RV} \cos \epsilon_H} \eta.
\end{equation}

(7) Note that, from (7), it must be that \( \epsilon_H \in [-\frac{\pi}{2}, \frac{\pi}{2}] \).

III. ILC+FBL DESIGN AND IMPLEMENTATION

In our implementation of ILC+FBL, a discrete-step ILC formulation is added to augment the FBL path following controller by applying additive corrections to the control input (steering rate). These corrections are applied for a specific path and learning must be conducted for each new desired route. We perform ILC in the feedback-linearized space defined by the coordinates transformation (3). This is slightly different from others who compute \( \omega \) directly as the feedforward signal—e.g., [15]. In our case, the learned corrections are applied to the transformed input (5) before performing a nonlinear transformation back to the steering input \( \omega \). Thus, we keep the single back-computation (7) unchanged over time; i.e.,

\begin{equation}
\omega = -v \sin \phi - \frac{\ell(\phi)}{\ell_{RV} \cos \epsilon_H} \eta + \bar{\eta},
\end{equation}

where \( \bar{\eta} \) is the computed additive correction.

A. Spatially-Indexed Discrete Formulation

ILC is normally implemented by indexing on discrete time steps [5], such that the system state is given in discrete time as \( q(k) := q(t = kT), k = 0, 1, 2, \ldots \) (e.g., see Fig. 4), where \( T \) is the time step. However, in the path following case, we cannot define errors as a function of time because the vehicle will likely have a different pose at the same time for every subsequent trial that depends on controller performance and vehicle speed. Thus, we define \( i \) as the time index when the \( i \)-th desired path point is reached and spatially index our system at these path points in the \( x-y \) plane, similar to the approach by Ostafew et al. [15]. The system state is then given in discrete time as \( q(i) := q(t = iT), i = 0, 1, 2, \ldots \), where we define \( T = nT \). Here we let \( n \) be the number of discrete time steps \( k \) between each path point, which may be variable. If \( 0 < T \ll 1 \) and \( n \) is small enough, then \( 0 < T \ll 1 \) is reasonable in our case assuming learning occurs at typical tramming speeds (i.e., \( v \) is not low or null). Recalling (4), the discrete linearized error system becomes

\begin{equation}
\begin{bmatrix}
z_1(i+1) \\
z_2(i+1)
\end{bmatrix} =
\begin{bmatrix}
1 & T \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
z_1(i) \\
z_2(i)
\end{bmatrix} +
\begin{bmatrix}
0 \\
T
\end{bmatrix} \eta(i)
\end{equation}

(9a) \begin{equation}
\psi(i) =
\begin{bmatrix}
1 & 0
\end{bmatrix}
\begin{bmatrix}
z_1(i) \\
z_2(i)
\end{bmatrix}
\end{equation}

(9b) where \( \psi \) is the error (output) over which we have chosen to learn (in this case, the lateral error \( \epsilon_L \)).

This formulation ignores vehicle states at discrete steps between path points. However, we contend that the assumption that \( n \) is small enough such that \( 0 < T \ll 1 \) is sufficient for the purposes of controller design (since it plays no part in the final design). A more complete treatment is described by Ostafew et al. [15], although the result is the same.

B. Error System Model

The goal of ILC+FBL is to learn the feedforward control inputs \( \bar{\eta}(i) \) over trial iterations indexed by \( j = 0, 1, 2, \ldots, J \).
1. We apply this additive correction to the real-time control inputs. Thus, from (6) and (9), we use the error model

\[ z_j(i+1) = Fz_j(i) + G(\eta_j(i) + \psi_j(i)) \]  
\[ \text{where} \quad \psi_j(i) = H(qI - \hat{F})^{-1}G\hat{n}_j(i) + HQ\tilde{z}_0. \]  

Because this system is a second-order state model, we have a relative degree of 1. Thus, from (6) and (9), we use the error model (10). We apply this additive correction to the real-time control inputs.

\[ z_j(i+1) = (F + G\hat{K})z_j(i) + G\hat{n}_j(i). \]

Now, we assume the initial condition \( z_j(0) = z_0 \) for all \( j \) and let \( q \) be the forward discrete time-shift operator, such that \( qz(i) = z(i+1) \). Thus, by Z-transforms and by using (9) and (10), the system is written as

\[ \psi_j(i) = H(qI - \hat{F})^{-1}G\hat{n}_j(i) + HQ\tilde{z}_0. \]

Assuming a desired system output \( \psi_0 \equiv 0 \) for path following, we write the performance error signal as

\[ e_j(i) = \psi_d(i) - \psi_j(i) \]
\[ = -\psi_j(i), i = 2, 3, 4, \ldots, N + 1. \]

C. Lifted Form

The above system can be written in the lifted form

\[ y_j = \Phi j + d \]
\[ e_j = -y_j, \]

where, for each iteration \( j \),

\[ y_j = \begin{bmatrix} \psi_j(2) \\ \psi_j(3) \\ \vdots \\ \psi_j(N+1) \end{bmatrix}, \]
\[ \Phi j = \begin{bmatrix} \hat{n}_j(0) \\ \hat{n}_j(1) \\ \vdots \\ \hat{n}_j(N-1) \end{bmatrix}, \]
\[ p_j = \begin{bmatrix} p_2 \\ p_3 \\ \vdots \\ p_{N+1} \end{bmatrix}, \]
\[ d_j = \begin{bmatrix} d(2) \\ d(3) \\ \vdots \\ d(N+1) \end{bmatrix}, \]
\[ \text{and the entries of } P = p_j = \hat{H}^{-1}G. \]

Note that because we use ILC in the feedback linearized space, we do not need to linearize at each discrete step and thus, the system model \( P \) is not a function of time.

D. Anticipatory Learning Algorithm

We employ the commonly-used iterative learning law [5]

\[ \hat{h}_{j+1} = Q(\hat{h}_j + Le_j), \]

which is a simple proportional law, where \( Q \) is referred to as the \( Q \)-filter and \( L \) is the learning function. The \( Q \)-filter is sometimes also called a “forgetting factor” because it is used to discount non-repeated disturbances, increasing robustness. Alternatively, the learning function can be thought of as a gain determining how aggressively the controller will alter the control input. This proportional law is causal, however, and therefore provides little learning benefit as it lacks the ability to “look ahead” along the path [5].

To introduce anticipatory behaviour, a phase lead ILC controller, as used by Wang and Ye [4], is employed. This follows from the proportional-type algorithm and has the lifted form

\[ \hat{h}_{j+1} = Q(\hat{h}_j + L\tilde{e}_j), \]

where \( L \) is the phase lead learning function expressed as

\[ \tilde{e}_j = \begin{bmatrix} 0 & \ldots & 0 & k_p & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & k_p \end{bmatrix}. \]

Here \( k_p \) is a proportional gain and \( L \) is non-causal (non-zero entries appear above the main matrix diagonal). For one discrete step the learning law (18) is expressed as

\[ \hat{n}_{j+1}(i) = k_q(\tilde{n}_j(i) + k_re_j(i + u)), \]

where \( u \geq 1 \) is the phase lead, representing the number of discrete steps to look ahead on the error signal from the previous iteration, and \( k_q \) and \( k_r \) are proportional gains. Thus, for the next iteration \( j + 1 \), the input at the same discrete step \( i \) is a function of the previous input and the error signal from \( u \) path points ahead in the previous iteration \( j \).

Stability for this control law can be shown by examining the error dynamics. By combining the lifted form system (15) and the learning algorithm (18), the error propagation can be expressed as

\[ \tilde{e}_{j+1} = Q(I - PL)\tilde{e}_j + (Q - I)d. \]

Thus, for stability we must choose the eigenvalues of \( W \) such that the spectral radius \( \rho(W) < 1 \).

IV. UNDERGROUND FIELD EXPERIMENTS

The ILC+FBL path following controller described in Section III was implemented and tested by using an industrial-scale center-articulated vehicle with sensor systems similar to those used by Marshall et al. [3]. The specific vehicle used was an Atlas Copco Scooptram ST18 (shown in Fig. 1). Testing was conducted at the Kvarntorp underground mine near Örebro, Sweden.

Illustrated within the laser-generated 2D occupancy grid map shown in Fig. 5 is the chosen (desired) path. In order to test the limits of the developed path following controller, two rounded 90-degree corners were prescribed because sharp corners prove the greatest challenge for accurate path
following. Additionally, each learning trial was carried out at a constant desired forward vehicle speed $\bar{v}$ for the entire path—i.e., no slow-down in the corners. The ILC+FBL controller was run for $J = 10$ learning iterations of the desired two-turn path and was tested at desired forward vehicle speeds ranging from 2.0 m/s to 5.0 m/s in 1.0 m/s increments.

A. Gain Selection

From the stability analysis in Section III-D, gains were selected for the learning function and the Q-filter ($k_p$ and $k_q$, respectively). In practice, Q-filter tuning begins with setting $k_q$ to unity and slowly decreasing until the desired trade off of performance and robustness is reached [5]. In this case, no Q-filtering was found to be necessary and was set to $k_q = 1.0$. The learning gain was set to $k_p = 0.4$ and was selected to ensure stability and to achieve desired learning aggression (convergence after minimal learning iterations). Additionally, the chosen $k_p$ and $k_q$ for FBL maintain separation between the inner and outer-loop bandwidths, as determined by Marshall et al. [3, Section 4.2]

The phase lead value $u$ in (20) was selected to provide anticipatory control and must account for the forward vehicle speed $\bar{v}$. Higher vehicle speeds required greater phase lead values due to the slow response of the vehicle. Thus, we devised an exponential relationship for the phase lead amount

$$u = m\bar{v}^a + b,$$

where $m$, $a$, and $b$ are tunable parameters. For the results presented here we chose $m = 2.0$, $a = 1.4$, and $b = 3.0$.

B. ST18 Field Results

Results from tests with the Atlas Copco ST18 underground mining vehicle show that the ILC+FBL algorithm successfully reduced both the lateral and heading path following errors ($\epsilon_L$ and $\epsilon_H$, respectively) by a drastic amount after $J = 10$ learning iterations at a constant high forward speed (results are shown for the highest speed of 5.0 m/s). Fig. 6 and Fig. 7 show approximate reductions in maximum lateral error by 93% and in maximum heading error by 70%. Additionally, the root mean squared (RMS) error for both lateral and heading errors show a significant decrease to a stable level, implying that the maximum error was not rectified at the expense of tracking performance elsewhere along the path. Note that the ILC error system output contains only the lateral error ($\bar{y} = z_1 := \epsilon_L$), yet learning improvement was evident in both the lateral and heading path following errors due to the dependence shown in their dynamic equations (2).

We observed only small path following errors in the straight portions of the path from the first trial, implying that the FBL controller alone handled straight sections with good performance. The spikes in the path following errors, however, corresponded to each corner in the desired path (see Fig. 5) where the FBL controller on its own was not sufficient for tracking at high speed ($\epsilon_L \approx 3.0$ m). Within only five learning iterations, however, the maximum lateral error was reduced by approximately 82% and the maximum heading error by approximately 55%.

To demonstrate repeatability of learned control inputs for ILC+FBL, a test was conducted (for 4.0 m/s vehicle speed) wherein the learning function was terminated after $J = 10$ iterations and the final learned inputs were applied to the path for five more path following trials solely using FBL. Fig. 8 shows the mean lateral and heading error along with the $3-\sigma$ (99.7%) confidence bounds for these five repeated trials. In the corners, we observed a very low variance (approximately $\epsilon_L \pm 0.1$ m and $\epsilon_H \pm 3.0$ deg). With such a large vehicle, these variances are considered to be extremely small and explain the slight up and down nature of the decrease in maximum heading error (see Fig. 7).

Initial conditions for each learning iteration were not identical because the vehicle was manually driven back to the start point after each trial. For this reason, approximately the first 10 m of the path showed spikes in error between each iteration while the vehicle corrected itself and returned to the desired path. These variations were discounted when
In future versions of ILC+FBL for path following, we plan on implementing speed incrementation. Fig. 6 shows very high errors in 90-degree corners for the first learning trial. This is unacceptable for tight tunnel scenarios, thus learning must commence at a lower forward vehicle speeds and include an algorithm to selectively increase the desired vehicle speed at each path point.

**REFERENCES**