Policy iteration-based integral reinforcement learning for online adaptive trajectory tracking of mobile robot

Tatsuki Ashida and Hiroyuki Ichihara

Department of Mechanical Engineering Informatics, Meiji University, Chiyoda City, Tokyo, Japan

1. Introduction

Trajectory tracking control is a fundamental and essential task for mobile robots. Model predictive control [1] and nonlinear control [2, 3] cope with the task so far. In these model-based control, the controller depends on the model, in which the control performance is constant. This fact limits the control performance when implementing the controller on other robots with different physical parameters and the environment. Adaptive dynamic programming (ADP) [4–7] does not need a complete model of the plant and information of the environment to update the parameters in the controller. There are several ADP methods to track trajectories for mobile robots [8–10]. For example, ADP is available for a discrete-time linearized tracking error model around the reference trajectory [8]. Otherwise, ADP-based adaptive update of the control law is also available for a continuous-time nonlinear tracking error model [9, 10]. These methods require the linearization around the system equilibrium point or information about the reference trajectory.

ADP deals with the Hamilton–Jacobi–Bellman (HJB) equation under the Bellman’s principle of optimality in mind without a model of the system but with the data of the robot’s trajectory [18]. ADP gives approximate solutions to the HJB equation by assuming the value function to a parametric function. In this context, the neural network is often useful in representing the parametric function [9]. The value function updates the control policy, i.e. the control law, iteratively with dynamic programming. One way to estimate the approximated value function and update the control policy is to use the actor–critic structure. The critic estimates the parameters while the actor improves the control policy [5, 6, 9, 11]. In particular, the article [11] discusses the convergence and stability properties of the policy iteration of ADP for discrete-time nonlinear systems.

ADP has been discussed typically for discrete-time systems. On the other hand, a method called integral reinforcement learning (IRL) [7, 12–14] allows us to consider learning control for continuous-time systems. Since existing systems follow some differential equations, learning methods of continuous-time systems are attractive. There are two ways to obtain the optimal control law in the IRL: the value iteration and policy iteration. The policy iteration is suitable for online learning, while the value iteration requires some noise to add when collecting the data. The control policy in the IRL-based policy iteration converges to the optimal one in the cases of offline learning [15, 16], and online learning [12], if one gets an approximate solution to the HJB equation. The IRL-based policy iteration gives an optimal solution to the HJB equation from limited real-time information on the control system. Online learning enhances control performance in real-time.

This paper considers a trajectory tracking control problem for a nonholonomic mobile robot. A time-invariant input-affine continuous-time nonlinear
system represents the tracking error dynamics. We propose an IRL-based policy iteration algorithm to design the control inputs for tracking the reference trajectory by using the data collected along the robot trajectory. Since approximation of the value functional is necessary for implementing the IRL-based policy iteration, this paper studies a selection of the basis function vector in an actor-critic structure. This paper also employs a batch least-squares minimization to update the control policy. The proposed algorithm sequentially updates the value function from online data. Finally, numerical examples illustrate to demonstrate the performance of the proposed policy iteration algorithm. A preliminary version of this paper is [17]. This paper includes comments on the convergence of the IRL-based policy iteration, a detail of the online learning algorithm, and additional numerical examples.

2. Problem formulation

Let \( X \times Y \subset \mathbb{R} \times \mathbb{R} \) be the coordinate on the plane. The continuous-time kinematic model of a nonholonomic unicycle mobile robot is written by

\[
\begin{align*}
\dot{x}(t) &= v(t) \cos(\theta(t)), \\
\dot{y}(t) &= v(t) \sin(\theta(t)), \\
\dot{\theta}(t) &= \omega(t),
\end{align*}
\]

where \( t \geq 0 \) is the time, \((x, y) \in X \times Y\) is the position of the mobile robot, and \( \theta \in [0, 2\pi) \) is the orientation with respect to \( X \times Y \). In addition, \( v \in \mathbb{R} \) and \( \omega \in \mathbb{R} \) are the translational velocity and the angular velocity, respectively, to drive the robot. \((x(t), y(t), \theta(t))\) is the robot trajectory.

Let \((x^f(t), y^f(t), \theta^f(t))\) be the reference trajectory. We assume that the virtual point satisfies Equation (1). The tracking error between the trajectory of the robot and the reference one is defined by

\[
\begin{bmatrix}
    e_x(t) \\
    e_y(t) \\
    e_\theta(t)
\end{bmatrix}
= \begin{bmatrix}
    \cos \theta(t) & \sin \theta(t) & 0 \\
    -\sin \theta(t) & \cos \theta(t) & 0 \\
    0 & 0 & 1
\end{bmatrix}
\times
\begin{bmatrix}
    x^f(t) - x(t) \\
    y^f(t) - y(t) \\
    \theta^f(t) - \theta(t)
\end{bmatrix}
\]

where \( e = [e_x, e_y, e_\theta]^\top \) represents the error between \((x, y, \theta)\) and \((x^f, y^f, \theta^f)\) as shown in Figure 1. The time derivative of the tracking error (2) is given by

\[
\begin{align*}
\dot{e}_x(t) &= x^f(t) \omega^f y^f(t) - v(t) + e_\theta(t)\omega(t), \\
\dot{e}_y(t) &= x^f(t) \omega^f y^f(t) - e_\theta(t)\omega(t), \\
\dot{e}_\theta(t) &= \omega^f - \omega(t),
\end{align*}
\]

where \( \omega^f \) and \( \omega^f \) are the reference trajectory counterpart of the translational velocity and the angular velocity, respectively. We assume that \( v^f \) and \( \omega^f \) are constant to make the error dynamics time invariant.

### Figure 1. Tracking error between robot state and reference trajectory.


3. Integral reinforcement learning

To solve the problem in the previous section, we discuss IRL in this section that requires a value functional,
a control policy, the Bellman equation, update of the
control policy, and properties of the policy iteration.

3.1. Value functional

Consider the error dynamics (5). The cost hat has the value associated with the error and a control policy is defined as the value functional. The value functional is defined as

\[ V^\mu(e(t)) = \int_t^{\infty} r(e(\tau), u(\tau)) d\tau, \]

where \( u = \mu(\cdot) \in \mathbb{R}^2 \) is the control inputs, \( \mu : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \) represents the admissible control policy that satisfies \( \mu(0) = 0 \), and \( r : \mathbb{R}^3 \times \mathbb{R}^2 \rightarrow \mathbb{R} \) is the local cost. Note that \( e = 0 \) in equation (5) results in \( u = 0 \), which implies \( \mu(0) = 0 \).

In this paper, the local cost is a quadratic form such that

\[ r(e(t), u(t)) = e(t)^T Q e(t) + u(t)^T R u(t), \]

where \( Q \in \mathbb{S}^3 \) and \( R \in \mathbb{S}^2 \) are positive semidefinite and positive definite matrices, respectively. Since \( \mu(0) = 0 \) holds, we have that \( V^\mu(0) = 0 \).

3.2. IRL form

The purpose of the IRL is to determine a control policy that minimizes the value functional from the data \( \{e(\tau), u(\tau)\} (0 \leq \tau \leq t) \). To do that, we rewrite Equation (6) as the following one [5]:

\[ V^\mu(e(t)) = \int_t^{t+T} r(e(\tau), u(\tau)) d\tau + V^\mu(e(t+T)), \]

where \( T > 0 \) is the sampling period, and

\[ V^\mu(e(t+T)) = \int_{t+T}^{\infty} r(e(\tau), u(\tau)) d\tau. \]

The continuous-time Bellman Equation (8) is called the IRL form.

3.3. Optimal control policy

The optimal value functional, that is a minimum of \( V^\mu(e) \) in the IRL form (8), is written by

\[ V^*(e(t)) = \min_{u(\cdot)} \left( \int_t^{t+T} r(e(\tau), u(\tau)) d\tau + V^*(e(t+T)) \right), \]

where \( u(\cdot : t+T) = \{u(\cdot) : t \leq \tau < t+T\} \) is the optimal control input. Under the assumptions of the existence of an optimal control policy, the optimal control policy can be written as

\[ \mu^*(e(t)) = \arg\min_{u(\cdot:t+T)} \left( \int_t^{t+T} r(e(\tau), u(\tau)) d\tau + V^*(e(t+T)) \right). \]

Let us define the Hamiltonian by

\[ H(e, u, \nabla V^\mu) = r(e, u) + (\nabla V^\mu)^T f(e) + g(e)u. \]

By differentiating Equation (6) with respect to the time, the following HJB equation [6, 18] hold:

\[ 0 = \min_{\mu(\cdot)} H(e, \mu(e), \nabla V^\mu). \]

Partial differentiation of the right-hand side of the Hamiltonian with respect to \( \mu \) yields

\[ \frac{\partial H}{\partial \mu} = 2u^T R + \nabla V^*(e(t)) g(e(t)) = 0. \]

Since \( R \) is positive definite, there is the unique control policy

\[ \mu^*(e(t)) = -\frac{1}{2} R^{-1} g^T(e(t)) \nabla V^*(e(t)) \]

that achieves the minimum of the Hamiltonian. Equation (13) is the optimal control policy in (10) satisfying Equation (11).

Substituting Equation (13) into the HJB Equation (11), (11) becomes the following nonlinear partial differential equation such that

\[ 0 = \nabla V^* f(e(t)) - \frac{1}{4} \nabla V^* g(e(t)) R^{-1} g^T(e(t)) \nabla V^* + x^T Q x. \]

In general, it is difficult to solve Equation (14) analytically to obtain \( \nabla V^* \).

3.4. IRL-based policy iteration

An initial control policy \( \mu_0(\cdot) \) that stabilizes the error dynamics (5) is needed to start policy iteration. The subsequent control policies are expected to be capable of stabilizing the error dynamics. This characteristic is suitable for online control policy update.

The policy iteration consists of two steps: the policy evaluation step and the policy improvement step. The iteration lasts until one gets a convergence while the data along the robot trajectory is collected online. In policy evaluation step that concerns the IRL form (8), we solve the following equation for \( V_{j+1}^\mu \) with the data
(e(τ), μ_j(τ)) (t ≤ τ ≤ t + T):

\[ V^μ_j(t + 1)(e(t)) = \int_t^{t+1} r(e(τ), μ_j(e(τ))) dτ + V^μ_j(t + 1)(e(t + T)), \]

where \( j \) is the iteration number. In policy improvement step that concerns the optimal control policy (13), we shall update the control policy by

\[ μ_{j+1}(e(t)) = -\frac{1}{2} R^{-1} g(e(t))^T ∇V^μ_j(e(t)). \]

Note that control policy (16) requires \( g(e) \) but not require \( f(e) \) in error dynamics (5). This implies that the IRL-based policy iteration does not need references \( ω^f \) and \( ν^f \) in (5). The implementation of the policy iteration will be discussed in Section 4.

### 3.5. Convergence of IRL-based policy iteration

A convergence property resides in the IRL-based policy iteration with online policy update as follows:

**Lemma 3.1** ([12]): Solving Equation (15) for \( V^μ_{j+1} \) is equivalent to finding the solution to the following equation:

\[ r(e(t), μ_j(e(t))) + (∇V^μ_{j+1})^T (f(e(t))) + g(e(t))μ_j(e(t)) = 0. \]

**Lemma 3.2** ([12]): The IRL-based policy iteration of Equations (15) and (16) converges to the optimal control policy \( μ^*(e) \) in (10) on the trajectories with the initial state \( e(0) \).

Let us comment on these lemmas. In the case of offline policy update, the policy iteration also has a convergence property based on Equations (16) and (17) [15, 16]. Since Equations (15) and (17) are equivalent according to Lemma 3.1, the IRL-based policy iteration converges to the optimal control policy (10) with both offline and online policy updates.

### 4. Approximate solution

Since the IRL form (8) is difficult to solve for the value functional \( V^μ(e) \) and may not have an analytic solution. Thus, we utilize a neural network for approximating the solution to (8). Then this section provides a detailed algorithm of the IRL-based policy iteration.

#### 4.1. Approximation of value functional

We assume that the value functional (6) has the form of the following single hidden layer neural network:

\[ V^μ_j(e(t)) = W^T_j φ(e(t)), \]

where \( W_j \in \mathbb{R}^L \) is the learning weight vector and \( φ : \mathbb{R}^3 \rightarrow \mathbb{R}^L \) is the basis function vector. Since \( V^μ_j(0) = 0, φ \) must satisfy \( φ(0) = 0 \).

The input layer of the neural network in Equation (18) takes the tracking error \( e(t) \) as the input. For a given basis function vector, the hidden layer finds \( L \) signals from the input \( e(t) \). Then, the output layer yields the estimate of the value functional.

#### 4.2. Online implementation of IRL-based policy iteration

To estimate the learning weight vector \( W_j \) in Equation (18) from the collected data, we introduce the actor-critic structure shown in Figure 2, where the critic and actor correspond to the policy evaluation step and the policy improvement step, respectively.

There is room to implement the IRL-based policy iteration in Subsection 3.4 and the article [12]. A pseudo-code of the IRL-based policy iteration is shown in Algorithm 1. An initial weight vector \( W_0 \) must be chosen so that the initial control policy \( μ_0(·) \) stabilizes the error dynamics (5). The weight vector \( W_j (j ≥ 1) \) is estimated repeatedly while \( ∥W_{j+1} − W_j∥ > ε \) holds for a given threshold \( ε > 0 \). In searching the weight vector, we solve a batch least-squares problem instead of recursive least-squares problem in [12].

**Policy Evaluation Step**

In the policy evaluation step, we first collect the data of the error dynamics (5) such that

\[ \{e(τ) : t + ℓT ≤ τ ≤ t + (ℓ + 1)T\} \]

for \( ℓ = 0, 1, \ldots, k_b - 1 \), where \( k_b \) is the batch size that must satisfy \( k_b > L \).

From the IRL form (15) and the approximate value function (18), one can find a learning weight vector \( W_{j+1} \) that satisfies the following relationship:

\[ \int_{t + ℓT}^{t + (ℓ + 1)T} r(e(τ), μ_j(e(τ))) dτ = W^T_{j+1}[φ(e(t + ℓT)) − φ(e(t + (ℓ + 1)T))]. \]

To determine \( W_{j+1} \), we solve the batch least-squares problem, i.e.

\[ \min_{W_{j+1}} \sum_{ℓ=0}^{k_b-1} δ(t + ℓT : t + (ℓ + 1)T)^2, \]

where \( δ(t + ℓT : t + (ℓ + 1)T)^2 \) is the tracking error at time step \( ℓT \).
where
\[
\delta(t + \ell T : t + (\ell + 1)T)) = \int_{t+\ell T}^{t+(\ell+1)T} r(e(\tau), \mu_j(e(\tau))) \, d\tau - W_{j+1}^T \left[ \phi(e(t + \ell)) - \phi(e(t + (\ell + 1)T)) \right]
\]

is the temporal difference error. The solution to problem (20) is
\[
W_{j+1} = (\Phi^T \Phi)^{-1} \Phi^T w,
\]
where
\[
\Phi = \begin{bmatrix}
\phi(e(t + T)^T - \phi(e(t))^T \\
\phi(e(t + k_b T)^T - \phi(e(t + (k_b - 1)T))^T \\
\vdots
\end{bmatrix}
\in \mathbb{R}^{k_b \times L},
\]
\[
w = \begin{bmatrix}
f_t^{T+T} r(e(\tau), \mu_j(e(\tau))) \, d\tau \\
\vdots \\
f_{t+(k_b-1)T} r(e(\tau), \mu_j(e(\tau))) \, d\tau
\end{bmatrix}
\in \mathbb{R}^{k_b}.
\]

The policy evaluation corresponds to the critic and approximates the value functional by updating the weights.

**Policy Improvement Step**

In the policy improvement step, from Equations (16) and (18), we shall determine an improved control policy by the following policy:
\[
\mu_{j+1}(e) = -\frac{1}{2} R^{-1} g(e)^T \nabla \phi(e)^T W_{j+1}.
\]

The weight vector \(W_{j+1}\), which has been updated at the policy evaluation step, determines the subsequent control policy. The policy improvement corresponds to the actor.

**Algorithm 1 Policy Iteration.**

Given \(\varepsilon, T, \phi(\cdot), \) and \(k_b\),

1. Select \(W_0\) to determine \(\mu_0(\cdot)\) using (21).
2. Set \(t = 0\) and \(j = 0\).
3. while \(\|W_{j+1} - W_j\| > \varepsilon\) do
   1. for \(\ell = 0\) to \(k_b - 1\) do
      1. Collect \(\{e(\tau) : t + \ell T \leq \tau \leq t + (\ell + 1)T\}\).
   2. end for
4. Solve (20) to obtain \(W_{j+1}\).
5. Determine \(\mu_{j+1}(\cdot)\) using \(W_{j+1}\) and (21).
6. \(t \leftarrow t + k_b T, j \leftarrow j + 1\)
7. end while

**5. Numerical example**

This section provides numerical experiments of the IRL-based policy iteration for the error dynamics (5) with Algorithm 1 to obtain the auxiliary control input \(u\) in Equation (4) with control policy (21). Note that vector \(g(e)\) is known while vector \(f(e)\) is unknown in Equation (5). We consider two types of reference trajectories in Subsections 5.1, 5.2 and 5.3: circular and straight line trajectories with different length of the basis function vector. These trajectories comes from constant \(v^f\) and \(\omega^f\). Subsection 5.4 discusses on the selection of the basis function vector. We also consider a set of time-dependent \(v^f\) and \(\omega^f\) in Subsection 5.5 that generates a figure-of-eight reference trajectory.

In all examples, we give initial learning weight vector \(W_0\) and attempt to improve the control performance by updating the vector.

**5.1. Circular trajectory \(L = 9\)**

Let us consider the reference trajectory generated by reference translational and angular velocities \((v, \omega) = (v^f, \omega^f)\) in equation (1). Given \((v^f, \omega^f) = (\pi/10, \pi/10)\), we have the corresponding reference trajectory \((x^f(t), y^f(t), \theta^f(t))\) as follows:
\[
\left( \cos \left( \frac{\pi}{10} t \right), \sin \left( \frac{\pi}{10} t \right), \frac{\pi}{2} + \frac{\pi}{10} t \right) \quad (t \geq 0).
\]

The basis function vector is given by
\[
\phi(e) = [e_x^2, e_x e_y, e_x e_\theta, e_y^2, e_y e_\theta, e_\theta^2, e_x^3, e_y^3, e_\theta^3]^T,
\]
that is, \(L = 9\), the weighting matrices \(Q\) and \(R\) in the local cost (7) are \(I_3\) and \(I_2\), respectively, the sampling period \(T\) is 0.1 (s), and the batch size of the least squares \(k_b\) is 22 and the iteration threshold \(\varepsilon\) is \(10^{-7}\). The initial conditions are the following: \((x(0), y(0), \theta(0)) = (0.5, 0, 0)\) and \(W_0 = [0.2, 0, 0, 0.2, 0, 0.2, 0, 0, 0, 0]^T\), that is, \(W_0^T \phi(e) = 0.2 e^T e\).

Then, we conduct a numerical experiment with the time from 0 to 20 (s). Figure 3 shows the robot and reference trajectories of the mobile robot. The solid and broken lines represent the robot and reference trajectories, respectively, while the dotted line represents the robot trajectory by the initial control policy. The dotted line seems to converge to the reference after a rotation. However, the trajectory does not converge exactly. On the other hand, the solid line converges to the reference at a half rotation. This figure shows that the policy iteration improves the tracking performance. Figures 4 and 5 show the tracking error and the control inputs, respectively. These figures illustrate that the algorithm works effectively to update the control policy. Figure 6 shows the transition of the learning weight vector \(W \in \mathbb{R}^9\). We can see that the learning weight vector converges at 6 iterations, which implies that the control policy also converges. In Figure 5, since the initial error...
Figure 3. Robot and circular reference trajectories ($L = 9$).

Figure 4. Evolution of tracking error for circular reference trajectory ($L = 9$).

Figure 5. Evolution of translational and angular velocities for circular reference trajectory ($L = 9$).

is far from the equilibrium point $(e_x, e_y, e_\theta) = (0, 0, 0)$, after the control policy update, we can see that spikes occur in the control inputs, $v$ and $\omega$, around 2 (s).

5.2. Circular trajectory ($L = 21$)

For the same circular trajectory, we use another basis function vector as follows:

$$
\phi(e) = [e_x^2, e_x e_y, e_x e_\theta, e_y^2, e_y e_\theta, e_\theta^2, e_x^3 e_y, e_x^2 e_y^2, e_x^2 e_\theta^2, 3 e_x e_y e_\theta, e_x^2 e_y e_\theta, e_x^2 e_y e_\theta, e_x^2 e_\theta, e_x^3 e_\theta, 4 e_x^2 e_y^2, 2 e_x^2 e_y e_\theta, 2 e_y^2 e_\theta, 4 e_y^2 e_\theta, 4 e_\theta^2]^	op
$$

that is, $L = 21$. The other parameters are the same as in Subsection 5.1, where $W_0$ is chosen so that $W_0^\top \phi(e) = 0.2e^\top e$ holds.

Figures 7 and 8 show the tracking error and the transition of the learning weight vector $W \in \mathbb{R}^{21}$. These figures indicate that, as in Subsection 5.1, the robot tracks the reference trajectory with updating the control policy. The error converges to zero at 4 iterations.

Figure 6. Transition of learning weight for circular reference trajectory ($L = 9$).

Figure 7. Evolution of tracking error for circular reference trajectory ($L = 21$).

Figure 8. Transition of learning weight for circular reference trajectory ($L = 21$).
5.3. Straight-line trajectory (L = 21)

Given \((v_{\text{rf}}, \omega_{\text{rf}}) = (0.1, 0) \quad (t \geq 0)\), we have the corresponding reference trajectory \((x_{\text{rf}}(t), y_{\text{rf}}(t), \theta_{\text{rf}}(t)) = (0.5 + 0.1t, 0, 0) \quad (t \geq 0)\). The weighting matrices \(Q\) and \(R\) are \(\text{diag}(1, 1, 0.01)\) and \(I_2\), respectively. The basis function vector and \(W_0\) are the same as in Subsection 5.2, i.e. \(L = 21\). The other parameters follow Subsection 5.1 except that the initial state is \((x(0), y(0), \theta(0)) = (-0.2, 0, -1/4\pi)\).

Then, we conduct a numerical experiment with the time from 0 to 30 (s). Figure 9 shows the robot and reference trajectories. Figures 10 and 11 show the tracking error and the transition of the learning weight vector \(W \in \mathbb{R}^{21}\), respectively. From Figures 9 and 10, the tracking errors converge to zeros asymptotically. From Figure 11, we can see that the learning weight vector converges at 7 iterations. This example shows that the mobile robot tracks the reference trajectory even when the reference trajectory is different.

5.4. Discussion on selection of basis function vector

Let us give a comment on the selection of the basis function vector \(\phi(e)\). The basis function vector needs to be chosen to which the parametric function is capable of sufficiently approximating the value functional. However, since the true vector is unknown and there is no constructive way to choose the basis function vector, we adopt a heuristic way. In Subsection 5.1, the basis function vector consists of monomials of \(e\). If we select monomials as the second order alone, i.e. \(L = 6\), then the learning weight vectors diverge. Similarly, in Subsection 5.3, the basis function vector of \(L = 9\) does not attain the control performance with that of \(L = 21\). The basis function should be selected according to the reference trajectory and the initial conditions of the robot.

The selection of monomial basis vectors in the article [12] experiences that the second order monomials are enough to obtain convergent trajectories for linear dynamics. In this case, the learning weights correspond to the third and higher order monomials vanish. On the other hand, the second order monomials are not enough and make nonlinear dynamics diverge. These observation are consistent with those in the examples in this paper.

5.5. Figure-of-eight trajectory

Let us give the time-dependent reference as follows:

\[
v_{\text{rf}}(t) = \frac{1}{10} \sqrt{4 \left(\cos \left(\frac{t}{5}\right)\right)^2 + \left(\cos \left(\frac{t}{10}\right)\right)^2},
\]

\[
\omega_{\text{rf}}(t) = -\frac{3 \sin \left(\frac{t}{10}\right) - 2 \sin \left(\frac{t}{10}\right)^3}{80 \sin \left(\frac{t}{10}\right)^2 - 85 \sin \left(\frac{t}{10}\right)^2 + 25}
\]

Figure 12. Robot and figure-of-eight trajectories.
for $t \geq 0$. Then, we have the corresponding reference trajectory $(x^M(t), y^M(t)) = (1.0 + \sin(t/10), 1.0 + \sin(t/5))(t \geq 0)$ where $\theta^M(t)$ is a complicated function of $t$. The weighting matrices $Q$ and $R$ are $\text{diag}(1, 10, 0.01)$ and $I_2$, respectively. $\phi(\epsilon)$ and $W_0$ are the same as in Subsection 5.1, i.e. $L = 9, T = 0.03$ (s), $k_\theta$ is 30, $\epsilon$ is $10^{-3}$, and $(x(0), y(0), \theta(0)) = (0.7, 0.8, \pi/4)$.

A numerical experiment is conducted with the time from 0 to 63 (s). Figure 12 shows the robot and reference trajectories. The trajectory with the IRL-based policy iteration converges to the reference while the trajectory with the fixed initial control policy $\mu_0$ obviously remain the tracking error. Since $v^M$ and $\omega^M$ are time-dependent, there is no guarantee that the convergent policy is optimal. The convergence of this example depends on the threshold $\epsilon$. If we take $\epsilon = 10^{-7}$ as in other examples, the trajectory diverges. From this observation, the optimal control policy is sensitive to time-dependent $v^M$ and $\omega^M$.

6. Conclusion

This paper adopted the IRL-based policy iteration to solve the trajectory tracking problem for a nonholonomic unicycle mobile robot. This paper adopts the reference trajectories whose translational and angular velocities are constant to make the error dynamics a time-invariant system. The obtained control policy determines the translational and angular velocities that drive the robot based on the data collected along the robot trajectory. To do that, we approximated the value functional to the linear combination of the monomials with coefficients. The data collected online train the coefficients as the parameters in the value function. In particular, this paper solves the batch least-squares minimization to update the control policy online. Numerical experiments demonstrated that the data updates the control policy to improve tracking performance by the proposed algorithm. Time-varying settings of the error dynamics based on time-dependent translational and angular velocities are the future direction.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP19H02158, JP19K04444.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by JSPS KAKENHI Grant Numbers JP19H02158, JP19K04444.

Notes on contributors

Tatsuki Ashida

He received his B.S. degree from Meiji University, Japan, in 2020. He has been a Master Student of the School of Science and Technology, Meiji University. His research interests include machine learning-based control engineering.

Hiroyuki Ichihara

He received his B.S., M.S., and Ph.D. degrees from Meiji University, Japan, in 1995, 1997, and 2000, respectively. From 2002 to 2009, he was an Assistant Professor of the Department of Systems Design and Informatics, Kyushu Institute of Technology. In 2019, he has been a Professor of the Department of Mechanical Engineering Informatics, Meiji University. His research interests include robust control and numerical optimization in control. He is a member of IEEE.

ORCID

Hiroyuki Ichihara http://orcid.org/0000-0001-8334-7349

References

[1] Klančar G, Škrjanc I. Tracking-error model-based predictive control for mobile robots in real time. Rob Auton Syst. 2007;55(6):460–469.

[2] Ailon A, Zohar I. Control strategies for driving a group of nonholonomic kinematic mobile robots in formation along a time-parameterized path. IEEE ASME Trans Mechatron. 2012;17(2):326–336.

[3] Yang J-M, Kim J-H. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. IEEE Trans Rob Autom. 1999;15(3):578–587.

[4] Lewis FL, Vrabie D. Reinforcement learning and adaptive dynamic programming for feedback control. IEEE Circ Syst Mag. 2009;9:32–50.

[5] Lewis F, Vrabie D, Vamvoudakis K. Reinforcement learning and feedback control using natural decision methods to design optimal adaptive controllers. IEEE Control Syst Mag. 2012;32:76–105.

[6] Lewis FL, Vrabie D, Syrmos V. Optimal control. 3rd ed. New York: John Wiley and Sons; 2012.

[7] Vrabie D, Vamvoudakis KG, Lewis FL. Optimal adaptive control and differential games by reinforcement learning principles. London: IET Press; 2013.

[8] Miah MS. Value iteration based approximate dynamic programming for mobile robot trajectory tracking with persistent inputs. Proc. of the IEEE International Symposium on Robotics and Intelligent Sensors (IRIS); 2017. p.44–49.

[9] Liu D, Wei Q, Wang D, et al. Adaptive dynamic programming with applications in optimal control. London: Springer; 2017.

[10] Walters P, Kamalapurkar R, Andrews L, et al. Online Approximate Optimal Path-Following for a Kinematic Unicycle. 2013. Available from: arXiv: 1310.0064.

[11] Liu D, Wei Q. Policy iteration adaptive dynamic programming algorithm for discrete-Time nonlinear systems. IEEE Trans Neural Netw Learn Syst. 2014;25(3):621–634.
[12] Vrabie D, Lewis FL. Adaptive optimal control algorithm for continuous-time nonlinear systems based on policy iteration. Proc. of the 47th IEEE Conference on Decision and Control; 2008. p. 73–79.

[13] Vamvoudakis KG, Vrabie D, Lewis FL. Online adaptive learning of optimal control solutions using integral reinforcement learning. Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL); 2011. p. 250–257.

[14] Vrabie D, Lewis FL. Neural network approach to continuous-time direct adaptive optimal control for partially-unknown nonlinear systems. Neural Netw. 2009;22(3):237–246.

[15] Abu-Khalaf M, Lewis FL. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. Automatica. 2005;41(5):779–791.

[16] Beard RW, Saridis GN, Wen JT. Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation. Automatica. 1997;33(12):2159–2177.

[17] Ashida T, Ichihara H. Online Adaptive Trajectory Tracking for Mobile Robot Using Integral Reinforcement Learning. Proceedings of the SICE Annual Conference; 2020. p. 1337–1340.

[18] Liberzon D. Calculus of variations and optimal control theory: A concise introduction. New Jersey: Princeton University Press; 2011.