Machine Learning for the Efficient Control of a Multi-Wheeled Mobile Robot

Uladzimir Dziomin, Brest State Technical University
(prof. Vladimir Golovko, Brest State Technical University)

Abstract
This paper presents an application of the multi-agent reinforcement learning approach for the efficient control of a mobile robot. This approach is based on a multi-agent system applied to multi-wheel control. The robot’s platform is decomposed into driving modules agents that are trained independently. The proposed approach incorporates multiple Q-learning agents, which permits them to effectively control every wheel relative to other wheels. The power reward policy with common error reward is adjusted to produce efficient control. The proposed approach is applied for the distributed control of a multi-wheel platform, in order to provide energy consumption optimization.

1. Introduction
An efficient robot control is one of the important tasks for the application of mobile robot in production. The important control tasks are power consumption optimization and optimal trajectory planning. Control subsystems should provide energy consumption optimization in a robot control system. The power consumption problem is solved by motor power control optimization [1], [2] and by efficient motion planning [3]. The robot control subsystem cannot influence on motor parameters, but must have policy for the efficient control (optimal speed parameter, maximum start driving power, safe slow down distance).

The trajectory planning usually implements by planning subsystem [4], [5]. Such subsystem builds trajectory and divides it into different parts, which are reproduced by circles and straight lines. The robot control subsystem should provide movement along the trajectory parts.

The problem of efficient control is an important prerequisite for the application of mobile robot platform that was developed in Hochschule Ravensburg-Weingarten. The 3D image of the Robot is illustrated in the fig. 1a. This platform is based on four innovated vehicle steering modules [6]. The steering module is shown in the fig. 2b and consists of two wheels powered by separate motors and behaves like a differential drive. It is mounted to the platform by means of a bearing which allows unlimited rotation of the module with respect to the platform. The platform can contain three or more modules.

![Fig.1. a) Robot platform 3D model; b) The driving module.](image)

The problem of multi-agent control is researched usually as problem of formation composition, trajectory planning, distributed control and others. In this paper we consider problem of circular motion for single and multi-module case. One solution of this problem [7]–[9] is to calculate kinematics of one wheeled robot for circle driving and after generalize it for multi-vehicle system. This approach has shown good modeling results. The disadvantage of this technique is small flexibility and high computational complexity. Alternative approach is to use coordination architecture with one or more leaders [10] where virtual coordinate frame follows the circle trajectory. The experimental results using a multi-robot platform have shown the effectiveness of this approach. The limitation of such a technique is explicit leader requirement.

In this paper we solve problem of optimal control for multi-module case in cooperative circular motion. The objective is to achieve a circular motion around a virtual reference beacon with optimal forward and angular speed. In this paper we develop reinforcement learning technique producing efficient control rule, based on the relative pose of the module with respect to the beacon. In order to illustrate the main features of the problem, the single module case and the multi-module scenario followed one by one are examined.

The key contribution of this paper are purposed reinforcement learning model for robot positioning...
that allow module positioning around beacon even if such beacon is dynamically change its position, and purpose reinforcement learning model for multi-module scenario that allow to adjustment module speed to required value within multi-module platform. This model require only positions of modules relative to the center of the multi-module platform.

By combining positioning and speed control modules are able to produce cooperative control law for efficient circular motion. As a result the developed system can be easy scaled for any number of modules.

2. Robot Control

The conventional approach for the platform control is kinematics calculation and inverse kinematics modeling [6]. If any modules on platform will be added or removed, it will require recalculation of kinematics equations and reconfiguration of control subsystem. Kinematics calculations can apply only for symmetric turning. For example we cannot drive using Ackerman (car-like) driving scheme [11], because it isn’t enough for the moving along the difficult trajectory parts.

![Fig.3. Robot kinematics for the symmetric turning.](image)

It should be noted that the previously developed control model uses only one rotation center that lies on the line, which is perpendicularly to the robot center (Fig. 3). The point G is the rotation center, the point S is the robot center and the SG line is the robot turning radius. It is an important restriction for the industrial control system, because the robot cannot drive in others ways.

3. Steering Module Agent

Let’s decompose the robot’s platform into the independent driving module agents.

Agent stays in physical 2D environment with reference beacon as it is shown in Fig. 4. Beacon position is defined by coordinates \((x_b, y_b)\). Rotation radius \(\rho\) is the distance from the center of module to the beacon.

![Fig.4. State of the module with respect to reference beacon.](image)

The angle error is calculated by the following equations:

\[
\varphi_{\text{center}} = \arctan(2(x_b - x, y_b - y))
\]

\[
\varphi_{\text{err}} = \varphi_{\text{center}} - \varphi_{\text{robot}}
\]

Here \(\varphi_{\text{center}}\) and \(\varphi_{\text{robot}}\) are known from environment.

In this paper environment is represented by physical 2d model simulated by Player/State. Environment provides the all necessary information about the agent and rotation point relative positions.

The environment information states are illustrated in the table 1.

<table>
<thead>
<tr>
<th>№</th>
<th>Robot Get</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X robot position, (x)</td>
<td>Coordinate, m</td>
</tr>
<tr>
<td>2</td>
<td>Y robot position, (y)</td>
<td>Coordinate, m</td>
</tr>
<tr>
<td>3</td>
<td>X of beacon center, (x_b)</td>
<td>Coordinate, m</td>
</tr>
<tr>
<td>4</td>
<td>Y of beacon center, (y_b)</td>
<td>Coordinate, m</td>
</tr>
<tr>
<td>5</td>
<td>Robot orientation angle, (\varphi_{\text{robot}})</td>
<td>Float number, radians</td>
</tr>
<tr>
<td>6</td>
<td>Beacon orientation angle relative to robot, (\varphi_{\text{center}})</td>
<td>Float number, radians</td>
</tr>
<tr>
<td>7</td>
<td>Radius size, (\rho)</td>
<td>Float number, m</td>
</tr>
</tbody>
</table>

The navigation subsystem of real steering uses odometer sensors for navigation purposes in the presented platform.

The full set of actions available to the agent is presented in the table 2. The agent can change the angle error \(\varphi_{\text{err}}\) around beacon, using control of linear \(v\) and angular speed \(\omega\).

<table>
<thead>
<tr>
<th>№</th>
<th>Robot actions</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Increase force, (v)</td>
<td>+0.01 m/s</td>
</tr>
<tr>
<td>2</td>
<td>Reduce force, (v)</td>
<td>-0.01 m/s</td>
</tr>
<tr>
<td>3</td>
<td>Increase turning left, (\omega)</td>
<td>+0.01 rad/s</td>
</tr>
<tr>
<td>4</td>
<td>Increase turning right, (\omega)</td>
<td>-0.01 rad/s</td>
</tr>
<tr>
<td>5</td>
<td>Do nothing, (\emptyset)</td>
<td>0 m/s, 0 rad/s</td>
</tr>
</tbody>
</table>
4. Multi-agent system of driving modules

One solution of formation control is the virtual structure approach [10]. The basic idea is to specify a virtual leader or a virtual coordinate frame located at the virtual center of the formation as a reference for the whole group such that each module’s desired states can be defined relative to the virtual leader or the virtual coordinate frame. Once the desired dynamics of the virtual structure are defined, then the desired motion for each agent is derived. As a result, single module path planning and trajectory generation techniques can be employed for the virtual leader or the virtual coordinate frame while trajectory tracking strategies can be employed for each module.

Let, N steering module agent’s with virtual leader forms a multi-agent system called platform.

Fig. 5 shows an illustrative example of such a structure with a formation composed of four modules, where \((x_0, y_0)\) represents the beacon and C represents a virtual coordinate frame located at a virtual center \((x_i, y_i)\) with an orientation \(\theta_i\) relative to beacon and rotation radius \(r_i\).

![Fig.5. The steering modules platform.](image)

Platform contains additional information such as square of platform and required module topology including its desired positions relative to the centroid of platform. Virtual leader seen by environment analogously to single steering module agent – it has a state, and can perform action. It receives the same information from environment defined in the table I, and action set defined in the table 2. It should be noted, that modules not directly controlled by virtual leader. The modules remain independent entities and adopt their behavior to conform desired position in platform.

In the fig. 6, \((x_i, y_i)\) and \((x_i^{opt}, y_i^{opt})\) represent, respectively, the \(i\)-th module’s actual and desired position, and \(d_i^{err}\) represent the desired deviation of the \(i\)-th module relative to desired position, where

\[
d_i^{err} = d_i^{err} - d_i^{opt}
\]  

Here \(d_i\) – distance from virtual center to current module position and \(d_i^{opt}\) required distance between virtual center and \(i\)-th module position derived from platform topology.

The virtual center position is derived from centroid of platform area. The virtual leader agent knows the goal optimal forward speed of whole platform limited in bounds \(v_{f\text{opt}} \in [v_{f\text{optmin}}, v_{f\text{optmax}}]\), where \(v_{f\text{optmin}}\) and \(v_{f\text{optmax}}\) is respectively minimum and maximum values of optimal speed.

5. Cooperative circular motion problem

Let’s examine the control problem of circular motion. The main objective is to build the control strategy so that all the modules within platform achieve circular motion around the beacon, with prescribed radius of rotation and distances between neighbors.

Then the task is to create cooperative control rule for any configuration of \(N\) modules within platform in circular motion around a fixed beacon, with rotation radius \(\rho\) defined for center of platform. Fig.7 depicts such a behavior. If each module can track its desired position accurately, then the desired formation shape can be preserved accurately. Further requirements are to take into account module positioning before movement, adaptation of angular and linear speed during circular movement. The control strategy should be scalable for various numbers of agents and its configurations.

The problem of finding multi-agent control law for circular motion can be decomposed in the following steps (a) module positioning and (b) forward speed adjustment to fit desired radius and position. For both problems we use reinforcement learning, which permits to achieve generalization ability. For example, the new beacon position can be dynamically assigned to platform, and the same control low can be used for module positioning.

6. Module positioning

The section discusses a reinforcement learning method producing efficient control law for module orientation around beacon.
6.1 Reinforcement Learning Framework

Reinforcement learning (RL) is used as one of the techniques to learn optimal control for autonomous agents in unknown environment [12].

The reinforcement learning framework is shown in the Fig. 8. The main idea is that agent execute action \( a_t \) in particular state \( s_t \) and receive reward \( r_{t+1} \) as a feedback of recent action. Agent should explore state space and for every state find actions, which is more rewarded than other in some finite horizon.

**Fig. 7.** Reinforcement learning framework.

Let, \( Q(s, a) \) – is a \( Q \)-function reflects quality of selecting specified action \( a \) in state \( s \). For given \( Q \)-function, the optimal action \( a^* \) in specified state \( s \) is defined as follows:

\[
a^* = \arg \max_{a \in A(s)} Q(s, a)
\]  

(4)

The initial values of \( Q \)-function are unknown and equal to zero. The learning goal is to approximate optimal \( Q \)-function, e.g. finding true \( Q \)-values for each action in every state using received sequences of rewards during state transitions.

For the moment of time \( t \), the change of \( Q \)-value can be calculated as follows:

\[
\Delta Q(s', a') = \alpha \delta'
\]

(5)

Where value \( \alpha \in (0, 1) \) – is learning rate, and \( \delta' \) – Temporal Difference (TD) error. The agent is learned in such a way that the TD error is decreased.

Using \( Q \)-learning rule [12], the temporal difference error calculated by:

\[
\delta' = r' - \gamma \max_{a \in A(s') - Q(s', a')}
\]

(6)

Where \( r' \) – reward value obtained for action \( a' \) selected in \( s' \), and \( \gamma \) – discount rate, \( A(s') \) – set of actions available at \( s' \).

5.2 RL-model for Module Positioning

Using defined above \( Q \)-learning rule define more precisely RL-model for module positioning including state, action and reward function description. It can be formulated as learning to find such a behavior policy that minimizes \( q_{\text{err}} \).

Let, state of agent will be pair of values \( s = [\phi, \omega] \). Action set \( A_\omega = \{\phi, \omega +, \omega -\} \) is represented by value of angular speed from table II.

Action of robot \( a \in A_\omega \) is a change of angular speed \( \Delta \omega \) for given moment of time \( t \).

The learning system is given a positive reward when the robot orientation closer to the goal orientation \( (\phi_{\text{goal}} = 0) \) using optimal speed \( \omega_{\text{opt}} \) and a penalty when the orientation of the robot deviates from the correct or selected action does not optimal for the given position. The value of the reward is defined as:

\[
r' = R(\phi_{\text{opt}}^{-1}, \omega^{-1})
\]

(7)

Where \( R \) is reward value which is represented by decision function depicting in the fig 8.

**Fig. 8.** Reward function decision tree.

Here \( \phi_{\text{opt}} \) – the value of angle, where robot reduce speed to stop at the correct orientation, \( \omega_{\text{opt}} \in [0.6...0.8] \) rad/s – optimal speed minimizing module power consumption. Angular speed within this range is given the highest award with exceptions in cases of acceleration and deceleration.

7. Cooperative Moving

In this section, we consider a multi-agent reinforcement learning model for cooperative moving problem. The problem is to control module’s individual speed in order to achieve stable circular motion of whole platform. Modules with different distances to beacon should have a different speed: for two modules \( i \) and \( j \), with distances to beacon \( \rho_i \) and \( \rho_j \) respectively, the speed \( v_i \) will more than \( v_j \) if the distance to beacon \( \rho_i \) more than \( \rho_j \). Every module should have additional policy to control its forward speed with respect to speed of other modules.

7.1 Multi-Agent Reinforcement Learning Framework

The main principles of such technique are described in [13]–[14]. The basic idea of selected approach is to use influences between module and platform virtual leader to determine sequences of correct actions in order to coordinate behavior.
among them. The good influences should be rewarded and negative should be punished. The code design question is how to determine such influences in terms of received individual reward.

RL-framework used for such control problem is illustrated in the fig. 9:

![Fig. 9. Multi-Agent RL framework.](image)

The i-th module at the state $s'_i$ selects action $a'_i$ using current policy $Q$, and goes to next state $s'^{i+1}$ taking action to environment. Platform observes changes done by executed action, calculates and assigns reward $r'^{i+1}$ to module as a feedback reflecting successiveness of specified action.

The same $Q$-learning rule (5)–(6) can be used to update module control policy. The main difference between both rules is that in second case reward is assigned by a virtual leader instead of environment:

$$
\Delta Q_i(s'_i, a'_i) = \alpha_{\text{p-o}} r'^{i+1} + \gamma \max_{a \in \mathcal{A}(s'^{i+1})} Q_i(s'^{i+1}, a) - Q_i(s'_i, a'_i)
$$

Instead of trying to build global $Q$-function $Q(s_1, s_2, \ldots, s_n, \{a_1, a_2, \ldots, a_n\})$ for $n$ modules we decompose the problem and build set of local $Q$-functions $Q_i(s, a), Q_2(s, a), \ldots, Q_n(s, a)$, where every policy contains specific control rule for each module.

The combination of such individual policies produces cooperative control law.

7.2 RL-model for cooperative moving

Let, state of the module is pair of $s = \{v, d^m\}$, where $v$ – current value of linear speed, and $d^m$ – distance error calculated by (8). Action set $\mathcal{A}_v = \{O_v, v_{+}, v_{-}\}$ is represented by increasing/decreasing of linear speed from the table II and action $a_i \in \mathcal{A}_v$ is a change of forward speed $\Delta v$ for given moment of time $i$.

The virtual agent receives error information for each module and calculates displacement error. This error can be positive (module ahead of the platform) or negative (the module behind of the platform). The learning process follows toward to minimization of $d^m$ for every module. The maximum reward is given for case where $d^m \rightarrow 0$, and a penalty given when the position of the module deviates from the predefined.

8. Simulation results

For simulation purposes the Player/State [15] modeling environment was used. Environment is represented by physical-like 2d world with four modules, virtual leader, platform description and beacon position.

The preparing part of collective movement simulation is learning of robot positioning. This step is done once for individual module before any cooperative simulation sessions. The Learned policy is stored and copied for other modules. The topology of $Q$-function trained during 720 epochs is shown in the fig. 10.

![Fig. 10. Result topology of Q-function.](image)

When the correct orientation of the robot is determined, angular speed is calculated to fit the specified driving radius using linear speed produced by cooperative control law:

$$
\omega = \frac{v}{\rho}
$$

Fig. 11 shows the platform initial state (left) and positioning auto-adjustment (right) using learned policy.

![Fig. 11. Initial and final agents position.](image)

Fig. 12 shows the experimental result of cooperative movement after learning positioning. It takes 11000 epochs in average.

The external parameters of simulation are summarized in the table 3.

In the case of modeling $\omega_{opt}$ is chosen from predefined bounds to show the applicability of proposed approach. For real robot, bounds of optimal speed is derived from documentation.
9. Conclusions and future works

The experimental results shows that described multi-agent reinforcement learning framework can solve the problem of efficient multi-wheel robot control. The proposed approach incorporates multiple Q-learning agents, which permits them to effectively control every wheel relative to the virtual leader. The reward functions designed in order to produce efficient control. A virtual leader is used to coordinate module speeds. Meanwhile, this role could be assigned to any module which has access to global information on platform level.

The advantages of this method are follows:

- **Decomposition** means that instead of trying to build global Q-function we build a set of local Q-functions.
- **Adaptability** – the platform will adapt its behavior for dynamically assigned beacon and will auto reconfigure moving trajectory.
- **Scalability and generalization** – the same learning technique is used for every agent, for every beacon position and every platform configuration.

Acknowledgement

I express sincere gratitude to my advisors Prof. Vladimir Golovko and Prof. Ralf Stetter, to my colleague Anton Kabysh for ideas and discussing.

Bibliography


Authors:

Mr. Vladimir Diomin
Brest State Technical University
267 Moskovskaja str.
224017 Brest
The Republic of Belarus
tel. (+375) 297 91 84 22
email: spas.work@gmail.com