Improved Model Predictive-Based Underwater Trajectory Tracking Control for the Biomimetic Spherical Robot under Constraints

Xihuan Hou 1, Shuxiang Guo 1,2,*, Liwei Shi 1, Huiming Xing 1, He Yin 1, Zan Li 1, Mugen Zhou 1 and Debin Xia 1

1 Key Laboratory of Convergence Medical Engineering System and Healthcare Technology, The Ministry of Industry and Information Technology, Beijing Institute of Technology, Beijing 100081, China; houxihuan@bit.edu.cn (X.H.); shiliwei@bit.edu.cn (L.S.); xinghuiming@bit.edu.cn (H.X.);
yinhe@bit.edu.cn (H.Y.); lizan@bit.edu.cn (Z.L.); zhoumugen@bit.edu.cn (M.Z.); xiadebin@bit.edu.cn (D.X.)
2 Department of Intelligent Mechanical Systems Engineering, Faculty of Engineering, Kagawa University, Takamatsu 761-0396, Japan
* Correspondence: guoshuxiang@bit.edu.cn; Tel.: +86-159-1102-2732

Received: 31 August 2020; Accepted: 13 November 2020; Published: 16 November 2020

Abstract: To improve the autonomy of the biomimetic sphere robot (BSR), an underwater trajectory tracking problem was studied. Considering the thrusters saturation of the BSR, an improved model predictive control (MPC) algorithm that features processing multiple constraints was designed. With the proposed algorithm, the kinematic and dynamic models of the BSR are combined in order to establish the predictive model, and a new state-space model is designed that is based on an increment of the control input. Furthermore, to avoid the infeasibility of the cost function in the MPC controller design, a new term with a slack variable is added to the objective function, which enables the constraints to be imposed as soft constraints. The simulation results illustrate that the BSR was able to track the desired trajectory accurately and stably while using the improved MPC algorithm. Furthermore, a comparison with the traditional MPC shows that the designed MPC-based increment of the control input is small. In addition, a comparative simulation using the backstepping method verifies the effectiveness of the proposed method. Unlike previous studies that only focused on the simulation validations, in this study a series of experiments were carried out that further demonstrate the effectiveness of the improved MPC for underwater trajectory tracking of the BSR. The experimental results illustrate that the improved MPC is able to drive the BSR to quickly track the reference trajectory. When compared with a traditional MPC and the backstepping method used in the experiment, the proposed MPC-based trajectory is closer to the reference trajectory.

Keywords: biomimetic spherical robot; trajectory tracking control; constraint; model predictive control

1. Introduction

With the rapid development of scientific ocean exploration, underwater robots have become one of the most important tools for exploiting and utilizing marine resources [1,2]. When compared with autonomous underwater vehicles (AUVs), the core feature of bionic small-scale robots refers to the capability of operating missions such as tracking ocean creatures and monitoring the marine environment in narrow underwater spaces. Therefore, these small-scale biomimetic robots have been receiving increasing interest from academia. Most researchers have focused on a prototype design [3–5] and the control methods of the driving system [6–9]. However, the closed-loop motion control (such as trajectory tracking, etc.) has seldom been considered, which is of primary importance for most applications. Over the past few years, a large number of studies have been dedicated to the underwater
trajectory tracking control for AUVs. Sliding mode control (SMC) is preferred for achieving underwater trajectory tracking, because it is insensitive to model uncertainties [10–12]. In [13], the authors combined SMC with the proportional-integral-derivative (PID) algorithm to enhance the trajectory tracking performance. In [14], an adaptive high order sliding mode controller was designed in order to achieve a AUV tracking trajectory on the yaw and depth. However, the “chattering” effect is the main drawback associated with SMC. In [15], a new adaptive term was developed to achieve a chattering-free sliding mode controller. The neural network algorithm is another commonly utilized algorithm in underwater trajectory tracking. In [16], the authors proposed an efficient neural network approach with a single-layer structure for tracking control. In [17], a novel adaptive neutral network tracking controller was constructed by combining dynamic surface control (DSC) and a minimal learning parameter (MLP), and a radial basis function neural network was employed to account for tracking error. The simulation results illustrate that the algorithm guarantees convergence of the tracking errors into a small neighborhood of the desired trajectory. Neutral-network based tracking control is insensitive to the dynamic model of the robot and has strong adaptability and learning capabilities [18]. However, because of the changeable underwater environment, the training and learning process is difficult, which leads to a poor real-time performance of the tracking algorithm. Backstepping control has been widely applied to the trajectory tracking of underwater robots. In [19], a bio-inspired model-based filter was integrated with the backstepping control for three-dimensional trajectory tracking of a manned submarine vehicle. The simulation results illustrate that this algorithm achieved satisfactory tracking results. In [20], a controller combined with the Lyapunov theory and a backstepping algorithm was designed for AUV motion control. A common drawback of backstepping control is that the speed sharply increases when there is a large tracking error, which leads to poor tracking results.

Most of the above studies on trajectory tracking control have achieved a good performance, but they do not consider actual system constraints such as the saturation of the control input. To solve the practical constraints in the controller design, model predictive control (MPC), which is the ability to cope with hard system constraints through optimization procedures, is an ideal algorithm. MPC-based trajectory tracking of AUVs has been conducted in recent years [21]. Path planning and trajectory tracking problems were integrated, which was resolved using nonlinear MPC in [22]. To improve the efficiency in solving the optimization problem, in [23] the authors incorporated log barrier functions into the cost function and modified the C/GMRES algorithm for use with a traditional MPC algorithm. In [24], to solve the trajectory tracking control problem, a novel Lyapunov-based nonlinear model predictive control (LMPC) scheme was developed for AUVs. The advantages of this new algorithm include a guaranteed closed-loop stability for optimization-based trajectory tracking control. Nevertheless, most of the above trajectory tracking control algorithms have been applied to AUVs models and they were demonstrated based on numerical simulations. The propulsion model of the biomimetic spherical robot differs from an AUV and the hydrodynamics are complex, which makes trajectory tracking for biomimetic sphere robot (BSR) difficult to achieve experimentally.

This paper focuses on the achievement of trajectory tracking with thrust constraints for the biomimetic spherical robot in an underwater plane. Inspired by the guidance in [24], a trajectory tracking controller that is based on a model predictive control algorithm is designed in this study. The main contribution of this paper is three-fold. First, the kinematic and dynamic models of the BSR are established. To establish the predictive model of the BSR, the identification of the dynamic model in yaw and surge is proposed. Second, an improved MPC control algorithm is developed in order to solve the BSR trajectory tracking problem. Third, simulation and experimental evaluations of the developed algorithm are conducted.

The remainder of this paper is organized, as follows. Section 2 describes the prototype of the biomimetic spherical robot and the established model of the robot for trajectory tracking. In Section 3, the MPC-based trajectory tracking controller is detailed. Simulation results are presented in Section 4. Section 5 describes the experimental results and discusses the comparison with the other tracking
algorithms. Finally, some concluding remarks and possible areas of future study are summarized in Section 6.

2. Modeling of Biomimetic Spherical Robot

2.1. Design of Biomimetic Spherical Robot

Figure 1 shows the prototype of the turtle-inspired biomimetic spherical robot that was developed for operation in a narrow aquatic environment. As introduced in references [25,26], the robot consists of upper and lower hemispheres, which are joined with a mid-plate. The upper hemisphere contains two parts: a water storage cabin, which is applied to adjust the buoyancy, and a sealed cabin, where an electrical board is placed. In addition, the robot is equipped with sensors for environmental perception, including 12 pressure sensors, an inertial measurement unit (IMU), a stereo camera, and an acoustic communication module. The lower hemisphere contains the vector propulsion mechanism, which consists of four sets of propulsion units [26], as depicted in Figure 2. Each propulsion unit is composed of three servo motors, a water-jet thruster, and three connections. A detachable battery cabin with 13,200 mAh was installed under the mid-plate. The maximum duration of the robot operation is approximately 100 min. The on-off switch on the top of the robot is used for a power-on or power-off. Furthermore, an optical fiber is installed on the robot, which is used for communication. Table 1 lists the concrete technical specifications of the BSR [27]. Currently, the robot is able to cruise in the underwater three-dimensional environment [26]. The maximum depth that the robot can reach is approximately 10 m.

![Figure 1. The prototype of the biomimetic spherical robot.](image1)

![Figure 2. Detailed structure of the vector propulsion mechanism.](image2)

2.2. Simplified Modeling of BSR for Trajectory Tracking

It is necessary to build a robot model to establish a state-space model. To clearly describe the dynamic and kinematic models of the BSR, two reference frames are established: a body reference frame (b-frame) and an inertial reference frame (i-frame). The body reference frame is attached to the vehicle with the origin selected to be the center of gravity. The motion of the BSR is described as the motion of the b-frame with in the inertial reference frame, which fixes the origin at a certain point on Earth. Figure 3 depicts the two coordinate systems. In the i-frame, the BSR position and orientation are represented, as follows:
Table 1. The Technical Specification of the Robot.

<table>
<thead>
<tr>
<th>Items</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension (Width × Length × Height)</td>
<td>30 cm × 60 cm × 30 cm</td>
</tr>
<tr>
<td>Mass in air</td>
<td>6.5 Kg</td>
</tr>
<tr>
<td>Processors</td>
<td>NVIDIA Jetson Tk1</td>
</tr>
<tr>
<td></td>
<td>STM32F407</td>
</tr>
<tr>
<td>Max thrust</td>
<td>3.8 N</td>
</tr>
<tr>
<td>Sensors</td>
<td>Pressure sensor (MS5803-14BA)</td>
</tr>
<tr>
<td></td>
<td>IMU (3DM-GX5-45)</td>
</tr>
<tr>
<td></td>
<td>Stereo camera</td>
</tr>
<tr>
<td></td>
<td>Acoustic communication module (Micron Sonar)</td>
</tr>
<tr>
<td>Power</td>
<td>7.4 V rechargeable Ni-MH batteries (13,200 mAh)</td>
</tr>
<tr>
<td>Operation time</td>
<td>average 100 min</td>
</tr>
</tbody>
</table>

Figure 3. The reference coordinate system.

\[ \eta = [x, y, z, \varphi, \psi, \theta] \] (1)

where \((x, y, z)\) is the position and \(\varphi, \psi, \text{ and } \theta\) are the Euler angle of the roll (along the x-axis), pitch (along the y-axis), and yaw (along the z-axis). The motion states of the BSR with respect to the i-frame are expressed in the following manner:

\[ \mathbf{v} = [u, v, w, p, q, r] \] (2)

where, \(u, v, \text{ and } w\) are the linear velocity in the x-, y-, and z-direction, respectively, and they are denoted as Surge, Sway, and Heavy. In addition, \(p, q, \text{ and } r\) are the angular velocity around the x-, y-, and z-axis, respectively, and they are denoted as Roll, Pitch, and Yaw, respectively.

The dynamics equation is established according to Newton [28,29]:

\[ (M_{RB} + M_A) \ddot{\mathbf{v}} + C(\mathbf{v}) \mathbf{v} + (D_l + D_q(\mathbf{v})) \mathbf{v} + g(\eta) = \tau \] (3)

where \(M_{RB}\) is the inertial matrix and \(M_A\) is the added matrix. In addition, \(C(\mathbf{v})\) denotes the Coriolis and centripetal matrix, \(D_l\) is the linear damping matrix, \(D_q(\mathbf{v})\) is the nonlinear damping matrix, \(g(\eta)\) represents the restoring force, and \(\tau\) is a six-dimensional vector that consists of the driving force and
moment generated by four thrusters. As the structure of the robot is symmetrical, the inertial matrix can be expressed, as follows:

\[
M_{RB} = \begin{bmatrix}
m & 0 & 0 & 0 & 0 & 0 \\
0 & m & 0 & 0 & 0 & 0 \\
0 & 0 & m & 0 & 0 & 0 \\
0 & 0 & 0 & I_x & 0 & 0 \\
0 & 0 & 0 & 0 & I_y & 0 \\
0 & 0 & 0 & 0 & 0 & I_z \\
\end{bmatrix}
\] (4)

where \( m = 6.3 \) kg is the quality of the BSR. Because the shell of the robot is symmetrical (neglected effects of the thrusters), there is no coupling term. The added matrix can be represented, as follows:

\[
M_A = -\begin{bmatrix}
X_\dot{u} & 0 & 0 & 0 & 0 & 0 \\
0 & Y_\dot{v} & 0 & 0 & 0 & 0 \\
0 & 0 & Z_\dot{w} & 0 & 0 & 0 \\
0 & 0 & 0 & K_p & 0 & 0 \\
0 & 0 & 0 & 0 & M_q & 0 \\
0 & 0 & 0 & 0 & 0 & N_r \\
\end{bmatrix}
\] (5)

The Coriolis and centripetal matrix \( C(v) \) can be ignored, because the speed of the robot is slow. The surrounding flow of the robot can be seen as a Stokes flow because of the low velocity of the robot. The flow viscosity is then \( C_l = 1 \times 10^{-4} \), and the linear hydrodynamic damping force is expressed as \( D_l = C_l \times \text{diag} \{1, 1, 1, 1, 1\} \). Because of the symmetrical structure of the robot, the nonlinear hydrodynamic damping force can be represented, as follows:

\[
D_q(v) = -\begin{bmatrix}
X_u|u| & 0 & 0 & 0 & 0 & 0 \\
0 & Y_v|v| & 0 & 0 & 0 & 0 \\
0 & 0 & Z_w|w| & 0 & 0 & 0 \\
0 & 0 & 0 & K_p|p| & 0 & 0 \\
0 & 0 & 0 & 0 & M_q|q| & 0 \\
0 & 0 & 0 & 0 & 0 & N_r|r| \\
\end{bmatrix}
\] (6)

Because the center of gravity and the buoyant center are at the same position, which is the geometric center of the BSR, the restoring force \( g(\eta) \) is set as \( 0^{6 \times 1} \). The trajectory tracking control of the BSR was studied on the local level plane because the thruster layout does not allow for active control of the roll and pitch motion. Usually, three degrees of freedom (surge, sway, and yaw) are considered in the trajectory tracking problem. However, the motions of the BSR on the degree of freedom in terms of Sway can be overlooked, because the thruster layout can only prove the surge force and yaw moment. Figure 4 shows the thruster layout when the robot tracks a trajectory.

The simplified dynamics equation is established, as follows:

\[
M\dot{v} + D(v)v = \tau
\] (7)

where \( v = [u, r]^T \); \( M = \text{diag}(M_u, M_r) \); \( D(v) = \text{diag}(C_l, C_l) + \text{diag}(X_u|u|, N_r|r|) \); \( \tau = [F_u, F_r]^T = Lu \); \( u = [u_1, u_2, u_3, u_4]^T \).

The simplified kinematics equation is expressed, as follows:

\[
\dot{\eta} = R(\theta)v
\] (8)
where \( \eta = [x, y, \theta]^T \) denotes the position and heading of the BSR in the i-frame, and \( \mathbf{v} = [u, r]^T \) is the velocity and yaw angle of the BSR in the b-frame. In addition, \( R(\theta) \) is the rotation matrix, depending on the heading angle \( \theta \).

\[
R(\theta) = \begin{bmatrix}
cos(\theta) & 0 \\
sin(\theta) & 0 \\
0 & 1
\end{bmatrix}
\]  

(9)

Figure 4. The propulsion distribution in surge motion. (a) Front view of the propulsion distribution. (b) Upward view of the propulsion distribution.

2.3. Model Parameter Identification of the BSR

In Section 2.2, the dynamic model of the robot is established. However, some parameters are not given. A concrete dynamic model is the basis of the design of the model predictive controller. In light of the modeling of the BSR that is described in Section 2.2, the coupling terms between different degrees of freedom (DOF) have been ignored. Subsequently, the identification of the whole model is converted into the individual identification of each DOF. The parameters of the DOFs of the Yaw and Surge are identified to achieve trajectory tracking. The dynamics equation of the DOF of the Yaw can be expressed, as follows:

\[
a_1 \dot{r} + a_2 r + a_3 r|r| = \tau_r
\]  

(10)

where, \( a_1, a_2, \) and \( a_3 \) need to be identified. Because there is no sensor for measuring the angle velocity, the difference in heading angle is used to compute the angle velocity. The online recursive least squares (RLS) technique is adopted to identify the unknown parameters [30–32]. According to the typical RLS algorithm, \( \Theta(t) = [a_1(t), a_2(t), a_3(t)] \). The initial estimate of \( \Theta(t) \) is the zero vector, and the initial error covariance matrix is set as \( 10^5 \times I_3 \), where \( I_3 \) is the three-dimensional identity matrix. The greater the number of data that are used (the sampling time was 20 Hz), the more accurate a model that can be obtained. The results are \( a_1 = 0.0021, a_2 = 0.1225, \) and \( a_3 = -3.2702 \times 10^{-4} \).

The dynamics equation of the DOF of a Surge is represented, as follows:

\[
b_1 \dot{u} + b_2 u + b_3 u|u| = \tau_u
\]  

(11)

Similarly, the unknown parameters, \( b_1, b_2, \) and \( b_3, \) are identified. Because there is no sensor for measuring the surge velocity, the difference in distance is used in order to compute the surge velocity. The distance of every adjacent moment is mapped with the top camera-based pixel distance. Finally, the results are \( b_1 = 0.4212, b_2 = 4.8180, \) and \( b_3 = -4.8276 \).

3. Improved MPC-Based Trajectory Tracking Controller Design

The underwater trajectory tracking controller for the BSR is depicted in Figure 5. The MPC controller is able to revise the control input according to not only the current state of the robot, but also the predictive future state error. In addition, during this process, the practical constraints can be copied with in the actual application. In order to develop a controller, the state-space-based predictive model,
the cost function, and the constraints need to be developed. The following subsections will build the MPC controller according to the above three steps above.

Figure 5. Diagram of the proposed model predictive control (MPC) controller for BSR trajectory tracking.

3.1. Linearized Error Model of BSR

Because the lower control input is the PWM signal driving the thruster to generate the thrust, in the predictive model, the relationship between the robot states (the position \(x, y\), and yaw angle \(\theta\)) and the driving force is considered. The dynamics of the robot reflect the relationship between the velocity and the driving force, and the kinematics describes the relationship between the robot states and velocity. Therefore, the predictive model for the BSR trajectory tracking is established by combining Equations (7) and (8):

\[
\dot{x} = \begin{bmatrix} R(\theta)v \\ M^{-1}(\tau - D(v)v) \end{bmatrix} = f(x, \tau) \tag{12}
\]

Here, the state is defined as \(x = [x, y, \theta, u, r]^T\), and the control input is defined as \(\tau = [F_u, F_r]^T\). It was determined that the above state-space equation is nonlinear. It is essential to linearize the Equation (12) in order to develop a state-space-based model predictive control algorithm. Each point on a referenced trajectory satisfies the above state-space equation, and \(d\) represents the reference. The general form is expressed, as follows:

\[
\dot{x}_d = f(x_d, \tau_d) \tag{13}
\]

where, \(x_d = [x_d, y_d, \theta_d, u_d, r_d]^T\) is a reference point and \(\tau_d = [F_{ud}, F_{rd}]^T\) is the corresponding control input. The Taylor series expansion on the reference point is applied to Equation (12), and high-order terms are ignored in order to obtain the following expression:

\[
\dot{x} = f(x_d, \tau_d) + \frac{\partial f(x, \tau)}{\partial x} \bigg|_{x=x_d, \tau=\tau_d} (x - x_d) + \frac{\partial f(x, \tau)}{\partial \tau} \bigg|_{x=x_d, \tau=\tau_d} (\tau - \tau_d) \tag{14}
\]

The continuous-time linearized error model of the BSR can be written through the following by subtracting Equation (14) from Equation (13).

\[
\dot{x} = A(t)\tilde{x} + B(t)\tilde{\tau} \tag{15}
\]
where, \( \tilde{x} = [x - x_d, y - y_d, \theta - \theta_d, u - u_d, r - r_d] \), and \( \tilde{\tau} = [F_u - F_{ud}, F_r - F_{rd}]^T \). The matrix \( A(t) \) and \( B(t) \) are described in the Appendix A.

By using the approximately discrete method, the discrete form of Equation (15) can be expressed, as follows:

\[
\tilde{x}(k + 1) = \tilde{A}\tilde{x}(k) + \tilde{B}\tilde{\tau}(k)
\]

(16)

where, \( \tilde{A} = I + TA(t), \tilde{B} = TB(t), I \) is an identity matrix.

3.2. Improved MPC Trajectory Tracking Control Law for the BSR

Setting an appropriate optimization objective is the key to tracking the desired trajectory as accurately as possible. According to the traditional MPC algorithm, the cost function in this study is designed, as follows:

\[
\min J = \sum_{j=1}^{N} \tilde{x}^T(k + j)Q\tilde{x}(k + j) + \tilde{\tau}^T(k + j - 1)R\tilde{\tau}(k + j - 1)
\]

(17)

where, \( Q \) and \( R \) are the weighting matrices, and \( N \) is the prediction horizon. The optimization objective of this function is the control input \( \tau \), which is unable to avoid a sudden increment of the control input and it will affect the continuity of the control input.

Equation (18) is transformed, as follows:

\[
\xi(k) = \begin{bmatrix} \tilde{x}(k) \\ \tilde{\tau}(k - 1) \end{bmatrix}
\]

(18)

A new state-space expression is obtained:

\[
\xi(k + 1) = A\xi(k) + B\Delta\tau(k)
\]

\[
\eta(k) = C\xi(k)
\]

(19)

where,

\[
A = \begin{bmatrix} \tilde{A} & \tilde{B} \\ 0_{m \times n} & I_m \end{bmatrix}
\]

(20)

\[
B = \begin{bmatrix} \tilde{B} \\ I_m \end{bmatrix}
\]

(21)

\[
C = \begin{bmatrix} I_n \\ 0_{n \times m} \end{bmatrix}
\]

(22)

Here, \( m \) is the dimension of the control input and \( n \) is the dimension of state value.

In addition, a relaxation factor is designed and added to the objective function in order to avoid no solution. The objective function designed for BSR trajectory tracking control can be established, as follows:

\[
\min J(k) = \sum_{j=1}^{N_p} \|\eta(k + j) - \eta_d(k + j)\|^2_Q + \sum_{j=1}^{N_c} \|\Delta\tau(k + j)\|^2_R + \rho\epsilon^2
\]

(23)

where, \( N_p \) is the predictive horizon, \( N_c \) is the controlling horizon, \( \rho \) is a weighting factor, and \( \epsilon \) is the relaxation factor.

3.3. Design of the Constraints

In a practical application scene, some constraints need to be considered. This paper considers the constraints of the thruster saturation and the increase in thrust. Constraints that need to be satisfied at the sampling time \( k \) are listed, as follows:
\[ \Delta \tau_{\text{min}}(k) \leq \Delta \tau(k) \leq \Delta \tau_{\text{max}}(k) \]  

(24)

\[ \tau_{\text{min}}(k) \leq \tau(k) \leq \tau_{\text{max}}(k) \]  

(25)

Because the optimization objective of the cost function is the increase in the control input, Equation (25) needs to be represented with the following control increment:

\[
\begin{bmatrix}
\tau(k) \\
\tau(k+1) \\
\tau(k+2) \\
\vdots \\
\tau(k+N_c)
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 & \cdots & \cdots & 0 \\
1 & 1 & 0 & \cdots & 0 \\
1 & 1 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & 1 & 1
\end{bmatrix}_{N_c \times N_c}
\begin{bmatrix}
\Delta \tau(k) \\
\Delta \tau(k+1) \\
\Delta \tau(k+2) \\
\vdots \\
\Delta \tau(k+N_c)
\end{bmatrix}
+ 
\begin{bmatrix}
1 \\
1 \\
1 \\
\vdots \\
1
\end{bmatrix}_{N_c} \tau(k-1)
\]  

(26)

For simplicity, the above formula can be expressed, as follows:

\[ U_k = S \Delta \tau_k + 1_{N_c} \tau(k-1) \]  

(27)

### 3.4. Solution to Proposed Algorithm

The optimization problem that is expressed in Equation (23) can be converted into the following quadratic programming (QP) problem with the following constraints:

\[
\begin{aligned}
\text{min} & \quad J = [\Delta \mathbf{U}(k)^T, \varepsilon]^T H [\Delta \mathbf{U}(k)^T, \varepsilon] + G [\Delta \mathbf{U}(k)^T, \varepsilon] \\
\text{s.t.} & \quad \Delta \mathbf{U}_{\text{min}} \leq \Delta \mathbf{U}(k) \leq \Delta \mathbf{U}_{\text{max}} \\
& \quad U_{\text{min}} \leq S \Delta \mathbf{U}_k + 1_{N_c} \tau(k-1) \leq U_{\text{max}}
\end{aligned}
\]  

(28)

where, \( H = \begin{bmatrix} \Theta^T Q \Theta & 0 \\ 0 & \rho \end{bmatrix} \), \( G = \begin{bmatrix} 2 \varepsilon^T Q \Theta & 0 \end{bmatrix} \), the definition of matrix \( \Theta \) is shown in the Appendix A. \( e_t = \Psi \xi(k) \) is the tracking error in the predictive horizon, and the definition of matrix \( \Psi \) is shown in Appendix A.

During each control period, the solution to Equation (28) can be obtained and expressed, as follows:

\[ \Delta \mathbf{U}_k = [\Delta \tau_k, \Delta \tau_{k+1}, \cdots, \Delta \tau_{k+N_c-1}]^T \]  

(29)

In this control series, the first element, as the practical controlling input increment, is applied to the robot system. The real control input is \( \tau(k) = \tau(k-1) + \Delta \tau(k) \).

The MPC-based trajectory tracking control will be implemented in a horizontal fashion. The control algorithm is summarized in Algorithm 1.

---

**Algorithm 1** MPC-based trajectory tracking control algorithm

1. Set the reference trajectory \( p(t) \);
2. Discretize the reference trajectory \( p(t) \) by \( t = K \times T \), obtain a series of reference point and form a matrix \( P \);
3. Initialize the state of the robot and weighting matrices \( R \) and \( Q \);
4. Compute the state error \( \hat{x}(k) \);
5. Solve the MPC problem (28), obtain the first element of the optimal solution, \( \Delta \tau \);
6. \( \tau(k) = \Delta \tau + \tau(k-1) \);
7. Input \( \tau(k) \) into the robot;
8. Measure the current state of the robot;
9. At next sampling time instant, then repeat from step 4;
4. Simulation Results

In this section, the simulation results of the trajectory tracking for the biomimetic spherical robot on the surface of the water are provided, which highlights the feasibility of the proposed MPC-based trajectory tracking control method. All of the simulations were conducted while using MATLAB 2015a on a Windows 7 equipped with an Intel Core i7 CPU at 3.6 GHz and 8 GB of RAM. During the simulations, the updating of the robot state is based on the identification dynamic model in Section 2.3 and Equation (9). Six parameters need to be set in the MPC controller, including the prediction horizon $N_p$, the control horizon $N_c$, the weight matrix $Q$ and $R$, the sampling time $T$, and the weight coefficient $\rho$. The sampling time $T$ is set as 0.05 s, which is consistent with the sampling time of the sensors installed on the robot. The prediction horizon $N_p$ can affect the tracking effect of the proposed algorithm. The larger $N_p$, the better the stability of the controller, but the slower the response. The control horizon, $N_c$, is almost half of the $N_p$. According to the experience and the simulation results, $N_p$ is set to 25, and $N_c$ is set to 15. The weight matrix $R$ prevents the control input from fluctuating too much. Matrix $Q$ determines the weight of the error between the real and desired trajectories in the cost function. The larger $Q$ is, the faster the response of the controller. In general, $Q$ is set to the identity matrix and $R$ is set to a matrix that is smaller than the identity. According to the simulation effect, the weighting matrices $R$ are adjusted to $5 \times I$, where $I$ is an identity matrix and $Q$ is adjusted to an identity matrix. The weight coefficient $\rho$ is set to avoid the case of no solution in the quadratic optimization, which has little effect on the simulation result. Parameter $\rho$ is set to 10 in the simulation. In addition, the limit of each thruster is 3 ($N$), and torque limit based on each thruster is 0.086 ($N \ast m$). The constraint ranges of the force and torque increment are $[-0.08, 0.08]$ ($N$) and $[-0.02, 0.02]$ ($N \ast m$), respectively.

4.1. Tracking Performance

Two desired trajectory tracking cases are used in order to verify the effectiveness of the improved MPC for the biomimetic spherical robot. The first (Case I) is a line trajectory defined, as follows:

$$
\begin{align*}
    p(t) = \begin{cases}
        x_d = 0.5t & 0 \leq t \leq 15s \\
        y_d = 0.5t & 0 \leq t \leq 15s
    \end{cases}
\end{align*}
$$

In case I, the initial state of the robot is $x(0) = [0, 0, 0, 0, 0]^T$. Figure 6 shows the trajectory tracking result, where the blue curve is the expected trajectory and the red curve represents the simulated trajectory. It was found that the proposed MPC-based algorithm is able to drive the robot to land on and track the reference trajectory. To clearly visualize the tracking result, the state errors between the simulated trajectory and the expected trajectory are depicted in Figure 7. Although some state errors $(x, y, r)$ are zero initially, the yaw angle of the robot is inconsistent with the reference value, which results in the change of the state errors at the next sampling time. However, it is obvious that all of the state errors converge to zero by the adjustment of the proposed controller at the end, which means that the robot lands on and tracks the desired trajectory accurately.

Figure 8 shows the required control input signals. The force for controlling the surge changes with the position errors, and the torque that is used to control the yaw angle changes with the yaw angle error. The controller generates a slightly large input initially, aiming to make the robot converge to the desired trajectory as quickly as possible. Nevertheless, the change in input is moderate throughout the entire process, which respects the increased constrains of the input. In addition, the input is in the physical limit of the thrusters during tracking. As the errors between the expected and simulated values decrease, the control signals tend toward a constant value.
The second one (case II) tracks a square trajectory that is defined by the following equation.
\begin{align*}
p(t) &= \begin{cases} 
  x_d = 0.5t \\
  y_d = 3 
\end{cases} \quad 0 \leq t \leq 35s \\
p(t) &= \begin{cases} 
  x_d = 17.5 \\
  y_d = 3 - 0.5(t - 35) 
\end{cases} \quad 35s < t \leq 70s \\
p(t) &= \begin{cases} 
  x_d = 17.5 - 0.5(t - 70) \\
  y_d = -14.5 
\end{cases} \quad 70s < t \leq 105s \\
p(t) &= \begin{cases} 
  x_d = 0 \\
  y_d = -14.5 + 0.5(t - 105) 
\end{cases} \quad 105s < t \leq 140s
\end{align*}

The initial condition of the robot in case II is $x(0) = [0, 4, \pi/4, 0, 0]^T$. Figure 9 shows the simulation result of case II. As this figure shows, the robot tracked the desired trajectory quickly although there were some sudden changes in yaw angle.

Figure 9. The biomimetic spherical robot trajectory tracking on a xoy plane-Case II.

Figure 10 shows the changes in the state errors while tracking a square trajectory. All of the errors increase sharply when the desired trajectory exhibits a sudden change in yaw. Moreover, the position errors oscillate slightly at the corners of the square trajectory, which is caused by the constraints of the input increment. However, all of the state errors tend toward nearly zero when the robot converges to the reference trajectory at the end of each line stage. The control input signals of the robot during the square-trajectory tracking are depicted in Figure 11. To drive the robot to converge as quickly as possible, the control input signal changes suddenly at the corresponding sampling time point when the yaw angle of the robot changes, as shown in this figure. Owing to the limit of the input increment, the input that was obtained by the proposed MPC method changes slightly several times to copy with the sudden change. In addition, all of the control signals are within the physical limit range of the thruster. When the robot tracks the reference trajectory accurately, the control input remains constant.

According to the simulated results of the two cases above, an improved MPC-based algorithm for the biomimetic spherical robot trajectory tracking on a two-dimensional plane is feasible. In addition, the control input and its increase satisfy the constraints.
4.2. Comparative Simulations

In order to verify the advantages of the proposed MPC, a comparison with the traditional MPC algorithm that considers the control input directly and no constraint to the increase of the control input was conducted. Figure 12 depicts the comparison results. The two algorithms were able to drive the robot to track the desired trajectory quickly, as shown in Figure 12. In order to clearly observe the tracking results, the end of each line stage is amplified. The proposed method converges to the desired trajectory faster, as depicted in the enlarged part of the figure. Besides, the traditional MPC-based tracking trajectory exhibits a small error with the desired trajectory when the BSR converges to the desired trajectory. Figure 13 plots a comparison of the state errors. As the limit of the increase in control, the state errors are slightly larger when the yaw angle suddenly changes. As the tracking result converges to the desired trajectory, the errors that are based on the improved MPC tend toward zero. While, as compared with the improved MPC, the traditional MPC-based tracking results exhibits few static errors at the end of each line stage. Figure 14 depicts a comparison of the control increments. When compared with the improved MPC, the traditional MPC generates a sharp increase in control when the yaw angel of the robot suddenly changes, which is likely to damage the thrusters. It was verified that the improved MPC respects the constraints of the control input and its increase while achieving the trajectory tracking task.
Figure 12. The comparison result in square tracking.

Figure 13. The comparison of the state errors in square tracking.

Figure 14. The comparison of control increment in square tracking.
Furthermore, a comparative simulation using a typical backstepping algorithm [19] that performs well in trajectory tracking was conducted. The simulation results are depicted in Figures 15–17. The simulation trajectories based on the improved MPC and the backstepping methods both catch up and land on the reference trajectory at the end, as indicated in Figure 15. However, the improved MPC is able to drive the robot in order to track the desired trajectory more smoothly. The state errors during the tracking process are plotted in Figure 16. As the constraints of the input increase, the position errors at the corner of the square are slightly larger than that based on the backstepping method. The error of the yaw angle based on the proposed MPC is clearly smaller during the entire tracking process. Figure 17 shows the comparison results of the control inputs. When compared with the backstepping method, owing to the ability to deal with the constraints of the proposed method, the control inputs are smooth and they strictly respect the saturation of the thruster. While, the control inputs obtained by the backstepping method jump sharply and reach $-6 \ N$ and $-0.18 \ N \times m$ when the tracking errors occur at the initial time, which overshoots the constraints of the thruster. A comparison of the control inputs reveals the advantage of the proposed algorithm in coping with the constraints.

**Figure 15.** The simulation results compared with the backstepping method in square tracking.

**Figure 16.** The comparison of the state errors with the backstepping method in square tracking.
5. Experiments and Results

5.1. Experimental Environment Settings and State Information Acquisition

Figure 18 depicts an experimental environment containing a water pool, a global camera system, and a computer. The dimensions of the water pool are $3.5 \times 2.5 \times 1$ m (length $\times$ width $\times$ depth). The global camera USB8MP02G that is used to obtain the position of the BSR is mounted on a shelf, which is 2.6 m above the ground, as shown in Figure 18a. Table 2 shows the parameters of the global camera.

The state information of the robot, which includes the position, yaw angle, surge velocity, and yaw velocity, is crucial for developing the MPC algorithm. The yaw angle is measured while using a high-precision inertial measurement unit (IMU) mounted on the robot. A global camera-based real-time localization system is utilized in order to obtain the position of the robot. The real-time localization system is achieved using the Kalman consensus filter (KCF) algorithm, which is described concretely in reference [33]. The position of the BSR was computed in the host computer. The BSR, as the slave computer, obtains the position information while using the optical fiber communication system. Figure 19 shows the information transmitted between the host computer and the BSR. Based on the position information and the yaw information at each sampling time, the surge velocity and yaw velocity are computed by the difference method.

---

**Figure 17.** The comparison of the control input with the backstepping method in square tracking.

**Figure 18.** Experiment set up for biomimetic sphere robot (BSR) trajectory tracking. (a) Experiment setup. (b) Image of water pool in the view of global camera perspective.
Table 2. The Parameters of the global camera.

<table>
<thead>
<tr>
<th>Items</th>
<th>Parameters</th>
</tr>
</thead>
</table>
| Intrinsic matrix       | \[
|                        | \begin{bmatrix}
|                        | 551.746 & 0 & 0 \\
|                        | 0.238 & 550.579 & 0 \\
|                        | 370.092 & 220.353 & 1 \\
|                        | \end{bmatrix} \] |
| Radial Distortion      | \[
|                        | \begin{bmatrix}
|                        | 0.103 & -1.330 & 3.596 \\
|                        | \end{bmatrix} \] |
| Tangential Distortion  | \[
|                        | \begin{bmatrix}
|                        | 1.223 \times 10^{-4} & -0.002 \\
|                        | \end{bmatrix} \] |

Figure 19. The information transmitting system of the robot.

5.2. Experimental Results

A series of experiments were carried out in order to validate the effectiveness of the MPC-based trajectory tracking algorithm. During the experiments, the sampling time of the robot was 20 HZ, the prediction horizon was 25, the control horizon was 15, and the weighting matrices were consistent with those in the simulations.

The first experimental case involves tracking a desired line. The initial state of the robot is $x(0) = [0.45, 1.0, 35, 0, 0]^T$. Figure 20 depicts the clip of the line trajectory tracking experiment, where the red dotted line is the desired trajectory. The robot mostly converges at the desired trajectory after 15 s. Figure 21 shows the tracking results of the robot in terms of the line tracking. Figure 21a shows the time evolution of the states. It can be seen that the proposed algorithm is able to force the robot to catch up with and be loaded onto the desired trajectory. The maximum following error of the position when the robot converges to the desired trajectory is less than 0.03 m (1/10 of the diameter of the robot). In Figure 21a, the control behavior on $\theta$ seems to be significantly poorer than that on $x$ and $y$. In fact, when the real yaw angle reached the desired yaw the first time, the position $y$ was slightly far from the desired position. If the real yaw angle is kept constant at zero, there will be a static error in position $y$. In order to eliminate the position error of $y$, the yaw angle increased again and then tended toward the desired yaw as the error of position $y$ approached zero. In addition, the size of the indoor pool is insufficient to remove the effects of the reflective waves at present, such that the robot is subject to external disturbances as the robot is initially close to the edge of the pool at the
beginning. The required control forces of each thruster are depicted in Figure 21b. It was determined that the magnitudes of the control forces remain within the permitted range, and the changes in forces are moderate.

![Image of the BSR tracking a line trajectory](image1)

**Figure 20.** Image sequences of the BSR tracking a line trajectory.

![Graphs showing the tracking results of the robot in line tracking](image2)

**Figure 21.** The tracking results of the robot in line tracking. (a) States of the robot at different times. (b) Thrusts varying of the four thrusters.

The second experimental case involves tracking a desired square-shape trajectory. The initial state of the robot is \( x(0) = [1.0, 0.25, -140, 0]^T \). Figures 22 and 23 show the experimental results of the second case. Figure 22 shows the image sequences of the square-tracking experiment. The robot can converge to and track along the desired trajectory. The states of the robot during the tracking of the square are plotted in the Figure 23a. During the square tracking, the yaw angle of the robot exhibited three sudden changes, whereas the positions (including \( x \) and \( y \)) were continuous, even at three corners. The sudden change in the yaw angle leads to a larger error in the yaw angle than that of the position. Furthermore, it was found that the errors when the robot converged to the desired trajectory at each edge of the square were almost zero. The control forces of the four thrusters are depicted in Figure 23b. When the yaw angle of the robot changes suddenly, the control force changes clearly to drive the robot close to the desired state as quickly as possible. Whereas, the thrusts are within the limit range of the control inputs during the entire process. A video of the square tracking experiment is attached in the Appendix B.
5.3. Discussion

5.3.1. Further Analysis of the Experiment Results

According to the analysis of the two experimental results above, the improved MPC-based tracking controller is able to make the biomimetic spherical robot converge to the desired trajectory on the X-Y plane. Table 3 summarizes the mean square errors (MSE) for all experimental cases. In the tracking of the line trajectory, the position errors are slightly larger than in other case. The main reason is a large error in the initial position. The error of the yaw angle in the case of tracking the square trajectory is the maximal. This is because there are three times of yaw angle adjustments. On the one hand, these state errors of the robot mainly originate from the localization algorithm and the sensors. On the other hand, four thrusters are of the same type, but they also have minor difference, which indicates that different thrusts are outputted under the same PWM signal. In general, these errors are acceptable relative to the size of the BSR.
### Table 3. Mean square errors (MSE) of Tracking Different Trajectories.

<table>
<thead>
<tr>
<th></th>
<th>Line</th>
<th>Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x[m^2]$</td>
<td>0.208</td>
<td>0.080</td>
</tr>
<tr>
<td>$y[m^2]$</td>
<td>0.210</td>
<td>0.067</td>
</tr>
<tr>
<td>$\theta[rad^2]$</td>
<td>0.246</td>
<td>0.437</td>
</tr>
</tbody>
</table>

#### 5.3.2. Comparison Experiment in Square Tracking

A comparison experiment was conducted in order to verify the advantages of the proposed method. Because the square tracking including four line tracking and three yaw angle changes, is a more difficult tracking task, the comparison experiment is only conducted in square tracking. In the experiment, the BSR tracked the same reference trajectory based on the proposed MPC algorithm, the traditional MPC, and the backstepping algorithm. Figure 24 depicts the comparison results. The asterisk designates the start point of the reference trajectory. The backstepping trajectory tracking algorithm is almost able to drive the robot to track the reference trajectory. As a sudden change in the yaw angle at the corner, a huge increment in the control input will be generated based on the backstepping method, which will damage the motors of the thrusters. In the experiment, a hard constraint was added to prevent such damage. Consequently, the backstepping algorithm performs somewhat poorly in the second and forth line stages. The traditional MPC can drive the robot to track and land roughly on the desired trajectory. While, as compared with the proposed MPC, there is a little oscillation in the second and last line stages. In addition, at the corner, the real trajectory based on the traditional MPC deviates the reference trajectory more seriously than that based on the proposed MPC. The lack of restriction on the control input increments is the main reason for the above results. Meanwhile, the proposed MPC performs well and stably in every line stage. Although the real trajectory slightly overshot the reference trajectory during the change in yaw angle, the robot was finally able to converge to the desired trajectory at the end. The mean square errors (MSEs) of the robot states (including $x$, $y$ and yaw angle $\theta$) during the tracking of the desired trajectory are shown in Table 4, which clearly reflects the better tracking effect of the proposed algorithm. It is clear that the real trajectory that is based on the proposed MPC is closer to the desired trajectory.

![Figure 24.](image-url) Results of the comparison experiments in square tracking.
Table 4. MSEs of Tracking square trajectory based on different control methods.

<table>
<thead>
<tr>
<th></th>
<th>Traditional MPC</th>
<th>Backstepping</th>
<th>Proposed MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x[m^2]$</td>
<td>0.046</td>
<td>0.043</td>
<td>0.037</td>
</tr>
<tr>
<td>$y[m^2]$</td>
<td>0.038</td>
<td>0.027</td>
<td>0.026</td>
</tr>
<tr>
<td>$\theta[\text{rad}^2]$</td>
<td>0.336</td>
<td>0.329</td>
<td>0.317</td>
</tr>
</tbody>
</table>

6. Conclusions and Future Studies

In this paper, an improved MPC method for underwater trajectory tracking is proposed to improve the automatic capability of the biomimetic spherical robot. In the new MPC controller design, the dynamics and kinematics were combined to establish a predictive model. The new state space model that is based on the control increment was then designed. Considering the actual application, the saturation of the thrust and the limit of the increase in thrust were considered when the objective function was formulated. The simulation results of tracking different trajectories revealed that the proposed MPC method was able to drive the robot to track the reference trajectory accurately under the constraints. When compared with the traditional MPC and the backstepping algorithm, the proposed MPC has a more accurate tracking effect and it requires a smaller increase in input. During the experiments, the improved MPC algorithm was able to achieve a different desired trajectory tracking with acceptable errors, which further indicated the effectiveness of the proposed control method for the biomimetic spherical robot. In addition, the comparison experiment illustrated that the accuracy of the tracking result based on the proposed method increased compared with that based on the backstepping method and the traditional MPC.

In the present study, the experiments were carried out in an indoor pool. Because the pool is not sufficiently large, the robot is subject to an external disturbance of the reflective waves. Thus, a controller with the ability to reject an external disturbance will be designed in the future. In addition, a series of field experiments will be conducted in order to verify the robustness and autonomy of the robot control system.

Underwater trajectory tracking is the basic ability to accomplish underwater tasks. For example, when hunting for an object, the robot needs to track the trajectory of the object. In addition, the research of the multi-robot formation system is popular at present. In fact, studies on multi-robot system tracks the desired trajectory as required, which will achieve the function of formation tracking. Many other underwater tasks, such as docking, cruising, and object hunting, require the accurate trajectory tracking. In general, it is necessary to accurately achieve trajectory tracking for many further underwater complex tasks.

Author Contributions: X.H. conceived and designed the algorithm and validated the algorithm by simulations and experiments, then wrote the paper. S.G. and L.S. guided the algorithm design and revised the manuscript. H.X. designed the compared algorithm and conducted the comparison experiment. H.Y., M.Z. and D.X. performed the experiment. Z.L. analyzed the date. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was partly supported by National High Tech. Research and Development Program of China (No.2015AA043202), the National Natural Science Foundation of China (61773064, 61503028), the National Key Research and Development Program of China (No. 2017YFB1304401).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In Section 3.1, the dynamic model of BSR for trajectory tracking and the linearized error model of BSR was given. The following describe the concrete values of matrixes $A(t), B(t)$ in Equation (15).
$A(t) = \begin{bmatrix}
0 & 0 & -u_d \sin(\theta_d) & \cos(\theta_d) & 0 \\
0 & 0 & u_d \cos(\theta_d) & \sin(\theta_d) & 0 \\
0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}_{2 \times 3}
\begin{bmatrix}
A_{22} & 0 & 0 & 0 \\
\end{bmatrix}_{5 \times 5}$  \hspace{1cm} (A1)

$A_{22} = M^{-1}D$  \hspace{1cm} (A2)

$B(t) = \begin{bmatrix}
0 & 0 \\
M^{-1} & 0 \\
\end{bmatrix}_{5 \times 2}$  \hspace{1cm} (A3)

In the Equation (23), the matrix $\Theta$ in the definition of $H$ is expressed as the following:

$\Theta = \begin{bmatrix}
CB & 0 & 0 & 0 \\
CAB & CB & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots \\
CA^{N_c-1}B & CA^{N_c-2}B & \cdots & CB \\
CA^{N_c}B & CA^{N_c-1}B & \cdots & CAB \\
\cdots & \cdots & \cdots & \cdots \\
CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c-1}B \\
\end{bmatrix}$  \hspace{1cm} (A4)

The matrix $\Psi$ in the definition of $G$ is written as follows:

$\Psi = \begin{bmatrix}
CA \\
CA^2 \\
\cdots \\
CA^{N_c} \\
\cdots \\
CA^{N_p} \\
\end{bmatrix}$  \hspace{1cm} (A5)

Appendix B

A video of the square trajectory tracking experiment is available on the website https://youtu.be/ZxMNDqL_MA8.

References


**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).