Journal of Marine Science and Application (2019) 18:228–238
https://doi.org/10.1007/s11804-019-00089-3

RESEARCH ARTICLE

Reinforcement Learning Based Obstacle Avoidance for Autonomous Underwater Vehicle

Prashant Bhopale[1] · Faruk Kazi[1] · Navdeep Singh[1]

Received: 24 September 2017 / Accepted: 19 March 2018 / Published online: 8 April 2019

Abstract
Obstacle avoidance becomes a very challenging task for an autonomous underwater vehicle (AUV) in an unknown underwater environment during exploration process. Successful control in such case may be achieved using the model-based classical control techniques like PID and MPC but it required an accurate mathematical model of AUV and may fail due to parametric uncertainties, disturbance, or plant model mismatch. On the other hand, model-free reinforcement learning (RL) algorithm can be designed using actual behavior of AUV plant in an unknown environment and the learned control may not get affected by model uncertainties like a classical control approach. Unlike model-based control model-free RL based controller does not require to manually tune controller with the changing environment. A standard RL based one-step Q-learning based control can be utilized for obstacle avoidance but it has tendency to explore all possible actions at given state which may increase number of collision. Hence a modified Q-learning based control approach is proposed to deal with these problems in unknown environment. Furthermore, function approximation is utilized using neural network (NN) to overcome the continuous states and large state-space problems which arise in RL-based controller design. The proposed modified Q-learning algorithm is validated using MATLAB simulations by comparing it with standard Q-learning algorithm for single obstacle avoidance. Also, the same algorithm is utilized to deal with multiple obstacle avoidance problems.

Keywords Obstacle avoidance · Autonomous underwater vehicle · Reinforcement learning · Q-learning · Function approximation

1 Introduction

The ocean is the central energy source of energy, minerals, food, etc. for human being hence understanding and exploring the ocean area becomes an important task (Council 1996). Such exploration can be carried out by humans themselves using different manned and unmanned vehicles. But sometimes, it is not possible for human being to personally visit some hostile areas like radioactive environments or higher depth; in such cases, autonomous underwater vehicle (AUV) plays a vital role for achieving such tasks. AUVs are unmanned type underwater vehicles which are used in the commercial, military, scientific, and private sectors which are designed to explore underwater areas and perform different missions like pipeline monitoring, etc. (Russell et al. 2014). AUVs are the most suitable candidate for exploration of extreme environments due to their ability to operate autonomously (Fossen 2011). In such operations, AUVs are supposed to maneuver on their own as per programmed mission, but such missions are prone to fail due to unknown obstacles in AUV's programmed path. Hence, obstacle avoidance becomes a necessary task for AUV.

Obstacle detection and avoidance can be carried out by designing proper feedback control for AUV. The controller designing process can be classified into two types, namely model-based control and model-free control. Model-based control requires precise computation; the typical classical controller design procedure requires derivation of an exact mathematical model by

Article Highlights
- In order to complete the given task in unknown environment, AUV must avoid collisions with obstacles.
- A modified Q-learning-based control is proposed to reduce number of collisions and compared with standard one-step Q-learning-based control.
- Function approximation is utilized along with RL to deal with continuous states and large state-space problem.
- Proposed RL-based control is utilized for multiple obstacle avoidance.

✉ Prashant Bhopale
psbhopale_p14@el.vjti.ac.in

[1] Electrical Engineering Department, Veermata Jijabai Technological Institute, Mumbai 400019, India

careful analysis of process dynamics; by using this mathematical model, control law has to be derived to meet certain design criteria (Su et al. 2013; Qu et al. 2017). Sometimes, reduced order models are used to design controller (Bhopale et al. 2017) but it again requires an abstract mathematical model of the plant. Construction of the abstract model may be carried out by system identification approach but it may increase the parameter dependency (Hafner and Riedmiller 2014) and if the behavior of plant in real time is different from the abstract model due to parametric uncertainties then, the controller designed using that model may fail. In such scenario, robust controller is proposed for AUV in Cheng et al. (2010) and Bhopale et al. (2016), but the performance of robust control is again limited due to assumptions of bounded uncertainties. All these methods mentioned above are usually based on specific environment and plant's abstract mathematical model and depend on more prior knowledge like experience and rules. Also, they lack self-learning property to adapt to various unknown environments. Once there is any change in the task or environment, the corresponding designed model-based controller need to be updated manually. Hence, it is better to incorporate model-free self-learning approach in designing feedback control for AUV since dependency on the mathematical model, and uncertainties will vanish and the controller will be developed depending entirely on plant's (AUV in our case) behavior in the unknown environments.

Different model-free control approaches have been proposed in literature; where the controller learns plant behavior using neural network (NN). Reinforcement learning (RL) can be considered as a suitable candidate for both self-learning model-based and model-free control approach. Kober (Kober et al. 2013) contains a detailed survey regarding the application of RL in the area of robotics, where AUV applications are also listed. Model-based RL approach utilizes the kinematic model of AUV or sometimes the behavior can be summarized into Gaussian process (GP) model and this model will be used for long-term prediction. Many researchers have combined other controllers with RL where the predicted control of classical control is used as known policy and this policy is modified using actor-critic approach; this particular approach is known as on-policy approach where predefined policy is available but it increases the dependency on model available or derived using NN or GP (Paula and Acosta 2015). On the other hand, the off-policy approach does not require knowledge of AUV's kinetics or dynamics and the agent learn entire control law by itself.

Q-learning is one of the off-policy RL algorithms which can learn from actual plant behavior and can decide its own control command depending on previous experience (Phanthong et al. 2014). The Q-learning process executes in the following sequence: for present state, agent selects action depending on policy (random policy or greedy policy) from Q-table which is a storage of state-action value pairs as per previous experience, this selected action is then executed on plant and generated output is measured, depending on how good or bad the output is, the agent receives reward or punishment and using this reward or punishment the Q-value for particular state-action pair is updated and stored as an experience in the Q-table. For next step, the same process is executed and Q-table is updated. In this way, the Q-table is updated iteratively with different state-action pair for the entire state and action space. At the end of the process, Q-learning policy learns the entire control law by itself for defined state-action space from the scratch, without knowing the kinetics or dynamics of the plant. This self-learning policy can be applied for AUV set-point tracking problem, where AUV can explore entire state space by trying each and every state-action pair (exploration process) to reach the set-point with the objective of maximizing cumulative reward. But in such exploration process if AUV comes across an obstacle and takes any random action in order to explore, it is possible that collision may occur with obstacle causing damage to AUV. In such case, some researchers have proposed auxiliary controller strategy to switch controller to avoid obstacle but again this controller has to be designed from the mathematical model of AUV which as stated above is prone to uncertainties. Also, curse of dimensionality and continuous state problem are some other drawbacks being faced while standard one-step Q-learning algorithm which is being utilized for AUV.

Hence, as a remedy, a modified Q-algorithm is proposed in this paper which does not require auxiliary control or mathematical model of AUV. In the proposed method force exploitation is carried out to deal with this obstacle avoidance problem when AUV is in the unsafe region. This method is entirely model-free and furthermore, NN-based function approximation is utilized to deal with the curse of dimensionality and continuous state problem. Together proposed method removes the dependency of plant model and deals with the curse of dimensionality and continuous state space problem while designing self-learning controller for AUV set point tracking and obstacle avoidance.

Remaining of the paper is organized as follows: In Section 2, a brief introduction to RL and the standard one-step Q-learning algorithm is presented. Then in Section 3, the main idea of obstacle avoidance for AUV using RL is proposed with a modified Q-learning algorithm. Also for the same problem, issues with continuous state space and curse of dimensionality are highlighted and utilization of function approximation method using the NN is proposed. In Section 4, the proposed approach is illustrated using simulation results. Finally, Section 5 concludes the paper by discussing the overall results of the proposed approach.

2 Reinforcement Learning(RL)

RL is a standard machine learning method which is used to solve sequential decision problems modeled in the form of

Markov decision processes (MDPs) (Powell 2007). The dynamic programming assumes deterministic system; on the other hand, RL is approximate dynamic programming obtaining an optimal control policy when the perfect mathematical model is not available. Hence, we can say that RL can be utilized as a model-free approach.

In the RL problem, the agent which is interacting with the environment has to observe a present state $s \in S$ of the environment and, depending on policy an action, $a \in A$ is selected, where state space S and action space A can be either discrete or continuous set, and can be single or multi-dimensional. It is assumed that state s_t at any time instant t contains all relevant information about the plant's current situation. As shown in Fig. 1 below, at time instant t agent observe the state s_t, selects an action a_t from A using policy decided, this action a_t which is used to control states of the system is executed on the environment, then the environment reacts to the action and next state s_{t+1} is generated by the system using action a_t. Now depending on the state s_{t+1}, a reward r_t is generated for state-action pair (s_t, a_t) this reward can be designed as a scalar value or a function of the error between the present state and destination/target state or combination of both.

The main goal of RL is to find a policy π by maximizing cumulative expected reward for the action a in given state s. The desired policy π can be deterministic or stochastic. We can say that a is a sample over actions distribution over new state encountered as $a \sim \pi(s, a) = P(a|s)$. The reward functions are commonly designed as function of present state, i.e., $r = r(s_t)$; current state and action pair, i.e., $r = r(s_t, a_t)$; or it can be a function of the transitions from one state to new state, i.e., $r = r(s_t, a_t, s_{t+1})$.

The RL agent is expected to find out the relations between available states, available actions, and earned rewards depending on experience or best available choices. Hence, knowledge of exploration and exploitation is necessary to design RL agent.

2.1 Exploration vs Exploitation

From the agent point of view, the environment may be static or dynamic; hence, agent has to try different actions randomly, receive a reward, and keep learning on trial and error basis, this process is known as exploration. When the agent chooses the best action from learned experience and minimizes the cost of learning, it is called exploitation. If the agent has very less experience in large dimensional spaces, then agent selecting best actions based on current learned experience (which is not sufficient) is not preferable, because better alternative actions may be available which were potentially never been explored, hence sufficient exploration has to be done for learning the global optimal solution. Now a question arises that how much and when to explore, and how much and when to exploit. However, too much exploration can cost more in terms of performance and stability when the online implementation is necessary. The greedy policy can be used in such case where the exploration rate is more when the agent starts learning, as experience is gained and exploitation rate increases and exploration rate decreases gradually to reach the optimal solution. This method can be used to deal with exploration and exploitation trade-off.

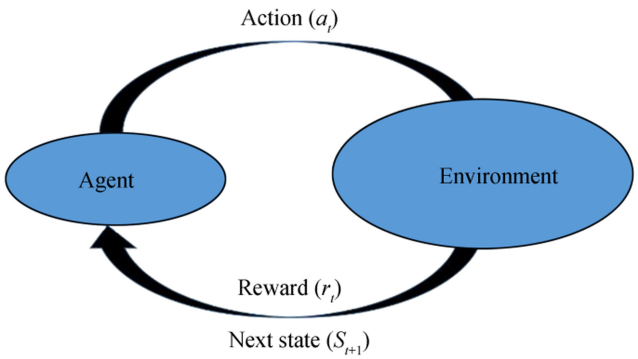

Fig. 1 Reinforcement learning mechanism where learning agent interacts with an environment

2.2 Q-learning

The Q-learning algorithm is model-free, off-policy RL technique, which uses temporal difference learning approach. It is possible to prove that if sufficient training and experience is given to RL agent under any soft-policy, then the algorithm will converge to close approximation of the action-value function for arbitrary target policy with probability 1. Optimal policy can be learned by Q-learning in both more exploration and random policy case.

The state-action value is updated in Q-learning as,

$$Q(s_t, a_t) \coloneqq Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)\right] \quad (1)$$

The following are parameters in the Q-value update process:

- α is a learning rate parameter; it can be set between 0 and 1 value. If α is set to 0, then, no learning process is carried out and Q-values will never be updated. If α is set to 0.9, then, learning can occur very quickly.
- γ is a discount factor; this parameter can take any value between 0 and 1. This factor is used to ensure that the future rewards are not worth.
- ε - if $\varepsilon = 1$ then pure exploration is carried out and if $\varepsilon = 0$ then pure exploitation. Hence ε is normally set to small positive value between 0 and 1. It is a probability for deciding the policy selection factor $\hat{B}$, when $\hat{B} = 1$ exploitation is carried out, else exploration continues.

In a scenario when each action is executed on every state in a huge number of times, and if learning rate α is decayed appropriately with increasing number of trials, the Q-values will converge optimal value Q^* with probability 1 see

(Watkins and Dayan 1992). In this case, Q directly approximates to optimal value Q^*, independent of the policy being followed. This approximation simplifies algorithm analysis and enabled early convergence proofs. But this policy still depends on which state-action pairs are visited and which value functions are updated hence it becomes mandatory to visit all state action pair. If proper exploration-exploitation is carried out then under this assumption Q_t converges to Q^* with probability 1, this one-step Q-learning algorithm (Sutton and Barto 1998) is shown below:

Algorithm 1 One-step Q-learning algorithm

Initialize $Q(s,a)$ arbitrarily
repeat (for each episode):
  Initialize s_t;
  repeat (for each step of episode):
    Choose a_t for s_t using policy derived from Q;
    Take action a_t, observe r_t and s_{t+1};
    update $Q(s_t,a_t) \coloneqq Q(s_t,a_t)$
    $+\alpha[r_{t+1}+\gamma \max_a Q(s_{t+1},a_t) - Q(s_t,a_t)]$
  $s_t \leftarrow s_{t+1}$
  until s is terminal
until all episodes end.

2.3 Q-learning in an Unknown Environment

Since Q-learning is a type of RL, it can directly interact with the unknown environment and develop self-learning control without any prior knowledge of the environment. When the environment is unknown, the obstacle avoidance problem of an AUV can be considered as a behavior selection task. In this task, the AUV can automatically produce a correct action of reaching the destination without collision according to the environment information perceived by the sensors equipped on the AUV. The task of finding the optimal path for AUV in an unknown environment is shown in Fig. 2. It is assumed that AUV's sensor system collects information regarding its own position $[x_t, y_t, \psi_t]$ and information of nearby obstacle's (represented by gray circles) position with its dimension at every time instant t. The black point is initial position of AUV represented by $[x_o, y_o, \psi_o]$ and the green point is the destination point fed to AUV which is represented by $[x_d, y_d, \psi_d]$.

The linear distance between AUV and destination point is calculated by

$$\Delta d = \sqrt{(x_d - x_t)^2 + (y_d - y_t)^2}$$

And the angular difference between AUV's current orientation and the destination expected orientation at time t is given by

$$\Delta\psi = \psi_d - \psi_t, \quad \Delta\psi \in [-\pi, +\pi]$$

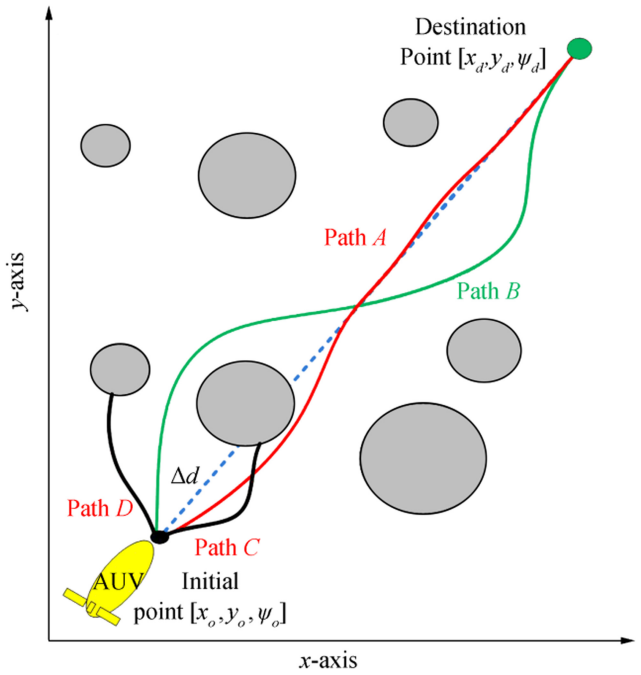

Fig. 2 Q-learning to find optimal path for AUV

In order to navigate the AUV to its destination point, it is assumed that these variables are always known at each time instant t. Therefore, an obstacle avoidance task is to obtain these variables, Δd and $\Delta\psi$ at each time step t, and based on them determine a state-action mapping process until the goal is achieved. As stated in above section Q-learning is a self-learning approach which learns optimal value function (1) with sufficient training using trial and error approach. Q-learning learns control policy for what to do and how to do, to maximize the reward value as stated in Algorithm 1. According to this algorithm, AUV first checks its current state s_t (position and orientation) in current environment, then pick up random action a_t. The random action will result in next state s_{t+1} and, depending on next state, the reward value r_t is generated as reinforcement signal depending on Δd, $\Delta\psi$ and whether target achived or collision occored. This reward value indicates the consequences or advantages of a_t at s_t.

The information s_t, a_t, s_{t+1} and r_t are fed to Q-value function and Q-value, $Q(s_t,a_t)$ is updated. This process is repeated for next state and taking random action at that state until the destination point is reached or AUV collide with some obstacle. If in this process, AUV find out 4 paths namely A, B, C and D as shown in Fig. 2 then the path A is decided as optimal path at the time of exploitation because Q-learning policy attempts to maximize the cumulative reward value which agent receives in progressive transition of states from its present state. But as we can see, there is no provision to avoid or reduce collision with obstacles in this standard Q-

learning algorithm, all it can learn is from experience, also there is no provision to deal with curse of dimensionality, hence a modified Q algorithm is proposed which can reduce the number of collision with obstacles in learning process and can deal with curse of dimensionality along with continuous state space problem.

3 Modified Q-learning

Traditionally, AUV consists 3 subsystems namely guidance system, navigation system, and control system. Guidance system designs an optimal path depending on vehicle dynamics and obstacles, control system executes the path, and navigation system estimates the states/trajectory in presence of noise or disturbances. Together this guidance, navigation, and control (GNC) systems strongly require the mathematical model of AUV which is again parameter-dependent and may fail due to parametric uncertainties, plant-model mismatch or change in the environment. Hence, to replace the GNC system, a behavior adaptive self-learning controller is required to be designed in such scenario which does not depend on plant's mathematical model. The self-learning standard one-step Q-learning algorithm stated in Algorithm 1 can be utilized for this task as stated in Section 2.3 but it has a tendency to explore all possible actions at given state which may be dangerous in presence of obstacle, as controller may try random action as stated in step 4 of Algorithm 1 in order to explore and end up in colliding with obstacle many more times, e.g., as Path C and Path D shown in Fig. 2. This random action exploration at all time is a drawback of present one-step Q-learning algorithm. Also, standard Q-learning requires a large amount of storage to save all discretized state and action space, this particular problem is known as the curse of dimensionality. Hence, to reduce the collision with an obstacle and to deal with the curse of dimensionality, a modified version of Q-learning algorithm is proposed in Section 2.3 by augmenting obstacle with an imaginary unsafe region around the obstacle, force exploitation when the unsafe region is detected and utilizing NN. This modified algorithm ensures that when the obstacle is detected AUV will not perform exploration (will not try new or random action) and force exploitation is carried out to get out of the unsafe region to reduce the number of collisions.

Static obstacles represent hard constraints that must be taken into account in the development of approximately optimal path planner. To facilitate the development of obstacle avoiding modified Q-learning control, we assume that AUV receives full knowledge about nearby obstacle, initial point and current state (position and orientation) using sensor mounted on AUV, then

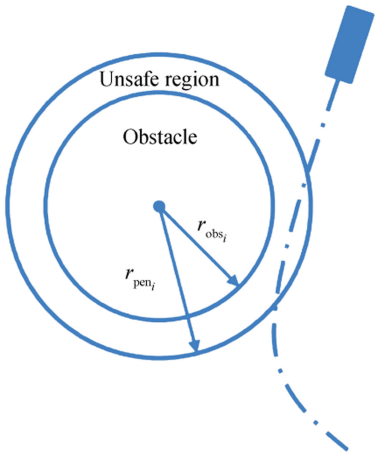

Fig. 3 ith obstacle with (radius r_{obs}) is augmented with the unsafe region (with radius r_{pen})

obstacles are augmented with an imaginary perimeter in the received database that extends from their borders denoting an unsafe region as illustrated in Fig. 3.

It is also possible to use auxiliary controller along with optimal Q control for obstacle avoidance, and switch in between auxiliary and Q control in case of the obstacle detected but it again increases model dependency. Hence, a new approach to deal with this problem by updating next values in Q matrix for upcoming obstacle and force exploitation in the process is proposed in this paper.

Normally, exploration and exploitation is a trade-off between deciding whether to take action which is safe (exploit), try well-known previously updated action which is having high rewards or dare to try new action (explore) in order to discover new strategies with an even higher or lower reward. But in presence of an obstacle, trying exploration is not a good idea since a random new action may result in collision with the obstacle, hence in the modified Q-learning algorithm, force exploitation is carried out whenever the AUV goes into the unsafe region around an obstacle to avoid the possibility of a collision. The ε is probability which is set between 0 and 1 to decide how much to exploit and how much to explore as stated in Section 2.1 above. The policy factor $\hat{B}$ is random value with $(1-\varepsilon)$ exploitation and (ε) exploration probability. When $\hat{B}$ becomes 1 then pure exploitation is carried out else pure exploration (process randomly choosing an action a_t at state s_t) will be continued as,

$$\text{Action } a_t = \begin{cases} \arg^{\max}_{a \in A} Q & \text{if } \hat{B} = 1 \\ \text{rand}(a_t \in A) & \text{otherwise} \end{cases}$$

As stated in Algorithm 2, if the AUV enters in the unsafe region then future Q-values is updated to avoid the collision and $\hat{B}$ will be set to 1 for pure exploitation irrespective of ε value. Until AUV is in the unsafe region this process repeated to move AUV into the safe region and avoid collision with obstacles. Proposed modified Q-learning algorithm is stated in Algorithm 2 below,

Algorithm 2 Modified one-step Q-learning algorithm

Initialize $Q(s,a)$ arbitrarily
Initialize desired state s_d
repeat (for each episode):
Initialize s_t;
Initialize exploration or exploitation policy factor $\hat{B}$ and store it as $\bar{B}$
repeat (for each episode):
if $\hat{B} == 1$
then $a_t = \max_a Q(s_t, a)$: choose action with maximum Q;
else $a_t =$ datasample(action space): randomly choose action;
end
Take action a_t, and observe s_{t+1};
Decide reward $r_t(s_t, a, s_{t+1})$ using Algorithm 3;
update $Q(s_t, a_t) \coloneqq Q(s_t, a_t)$
$+\alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)]$
if unsafe region is detected
$Q(s_{t+1}, a_t) = \text{maximum}(Q)$: give minimum value for next step and same action to not to go closer to the obstacle or to avoid collision.
$\hat{B} = 1$: make pure exploitation to avoid next action in same direction
else
$\hat{B} = \bar{B}$: load saved factor for episode.
end if
$s_t \leftarrow s_{t+1}$
until s_{t+1} is terminal
until all episodes end.

To explain the importance of policy factor $\hat{B}$ in Algorithm 2, the procedure in this algorithm is shown as flowchart in Fig. 4.

The reward r can be designed as a function of the error between desired state s_d and new state s_{t+1} for the transition using action a_t, also the reward value will be depending on the current state of AUV that whether present action a_t made the transition from safe to unssafe resion, unsafe to safe resion or collision occurred. The short schematic for the same is stated in Algorithm 3 below,

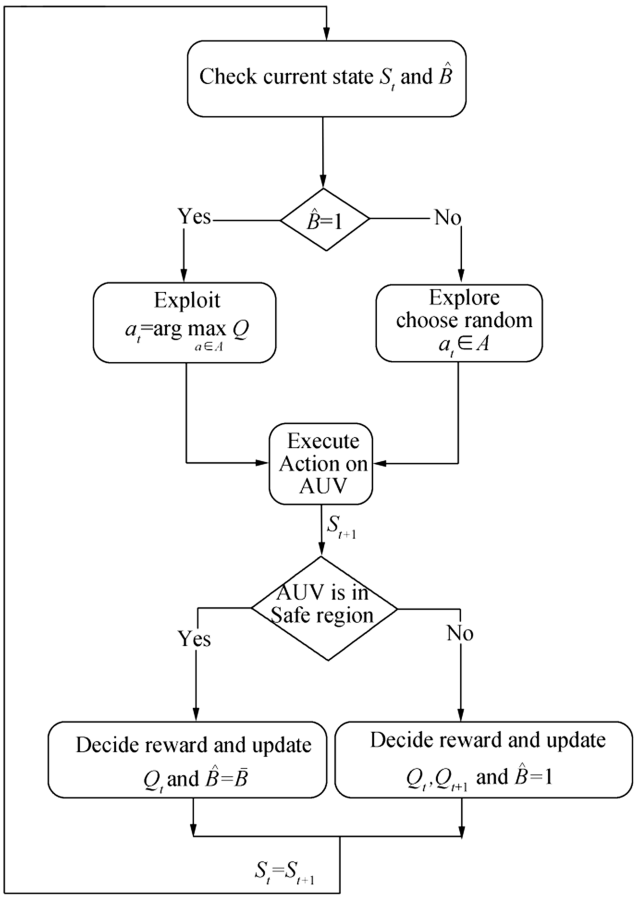

Fig. 4 Flowchart for modified Q algorithm

Algorith 3 Reward function for obstacle avoidance

if transition from safe region to unsafe region,
then $r_t = -10$;
elseif transition from unsafe region to unsafe region,
then $r_t = -20$;
elseif transition from unsafe region to safe region,
then $r_t = +10$;
elseif collision with obstacle,
then $r_t = -100$ and restart the exploration;
else it's transition from safe region to safe region,
then $r_t(x_t, a_t) = \tanh^2(|s\_d - s_{t+1}| * (\tan^{-1}(\sqrt{0.95}/\mu)) * C$
end if

The second last step in algorithm 3 is reward function as a function of error. This algorithm ensures that if the AUV is entered in the unsafe region, it will try not to go closer to the obstacle avoiding the collision and get out of the unsafe region by taking adifferent action. If the AUV in a safe region it will try to take greedy action towards the desired set-point.

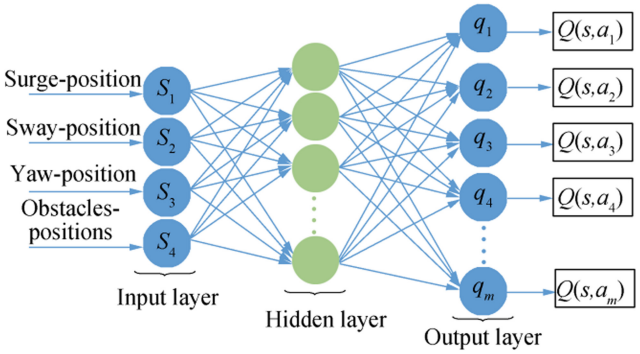

Fig. 5 Neural network (NN) for modified Q algorithm

3.1 Issues with Continuous State Space and Curse of Dimensionality

RL algorithms can be modeled as MDPs; hence, we are required to define state space S and action space A. Q-learning is executed in order to learn the mapping from present state input (s_t) to the highest value of tried action ($\max(Q(s_t, a_t))$. In a navigation problem, the AUV receives the state information from environment using its internal measurement units (IMUs), and this state is used to decide which action is to be taken in order to achieve desired setpoint/goal.

For AUV action space, A is defined by the span of rudder plane [$\min\delta_r$ $\max$ δ_r] and stern plane [$\min\delta_s$ $\max$ δ_s]. Since the maximum span is [−20 + 20] in degrees for AUV the action space for each control plane can have m user-defined discrete values. This will help AUV in taking other decision and avoiding the previous decision in case if AUV is in unsafe region near the obstacle.

It is assumed that, when sensor system detects a nearby obstacle, the AUV will receive this information from vector $U \in R$, as an indicator of upcoming obstacle's position and it is unsafe region. Therefore, a state space is augmented with U to define two groups of features, and is expressed as,

$$S_t = \begin{bmatrix} s_t \\ U \end{bmatrix} \tag{2}$$

But as we discretized the states, it is not always possible to be precise and optimal at the same time. Choosing to be more precise will make increase the computation cost; hence, there is a trade-off between the smoothness of the output trajectory and computational efficiency. In such case output, will not remain smooth. (Yoo and Kim 2016) used path smoothing to deal with the smoothness of the output trajectory but computation cost is still high. Also, if the state space discretized coarsely then the dimension of Q matrix increases to high extend increasing storage space, this particular problem is known as the curse of dimensionality. Hence, we propose to use function approximation to deal with continuous state problem and curse of dimensionality.

3.2 Function Approximation Using Neural Network

Traditional Q-learning can be designed directly for AUV but on the cost of discretization of states and actions. However, in AUV navigation task, the states are continuous due to continuous motion and sensory inputs; hence, it is required to have large memory space to store all the state-action pair value for Q-table and learning speed may decrease as it required to precise state space. This is typically known as the curse of dimensionality. In order to solve this problem, function approximation using the neural network (NN) can be used, since it provides a good generalization as a universal function approximation also it has strong ability to deal with large-scale state spaces.

NN basically have three layers namely input layer where input data is loaded, output layer where output data is loaded for training or output is generated for testing, and the intermediate layer is known as hidden layer where different function (e.g., Sigmoid) is used for function approximation, these hidden layers are connected to input and output layer via links and these links have weights assigned. NN can be classified into two types: feed-forward neural network (FFNN) where the weights are fixed and not changed and back-propagation neural network (BPNN) where weights are updated using various methods like back-propagation to train the NN.

In order to propose the NN based Q-learning, the traditional Q-table is replaced by function approximation using three layers NN as shown in Fig. 5.

The input layer of NN has 4 inputs, where 3 inputs are AUV positions in the surge, sway, and yaw direction, and fourth input is obstacle position. The action space is divided into 21 discrete states for convenience and represented as m number of Q-values at the output layer. A NN with fully trained weights is utilized in AUV's navigation problem. For every state transition from s_t to s_{t+1}, the inputs are passed through input layer as shown in Fig. 5 and predicted output is generated by NN. The weights are updated on the basis of networks error, which is a difference between its predicted

Table 1 Parameters for AUV

Initial poistion	$x = 1, y = 1, \psi = 0^\circ$
Desired set point	$x = 100, y = 100, \psi = 45^\circ$
State space	$x \in 0:110$ meters, $y \in 0:110$ meters, $\psi \in 0:359^\circ$
Action space	$[0, \pm5^\circ, \pm10^\circ, \pm15^\circ, \pm20^\circ]$

Table 2 Parameters for Q-learning

γ	0.5
ϵ	0.9
α	0.5
Iterations	1000

output and target value expected output. The optimal Q-value is treated as the target for the output for the respective selected action and the difference is backpropogated using gradient descent method to optimize the network error.

The algorithm of NN based Q-learning can be divided into two different processes. The first process is the training of NN and the second process is the navigation process to use the trained policy in NN to finish a navigation task. In the training process, current state S_t as shown in (2) is providedas an input to FFNN, then FFNN generates m possible action pair as shown in Fig. 5. Among this m output, one output is selected as per the action selection mechanismis shown in Fig. 4, this action is then executed on AUV and moves it to the new state. Then, immediate reward is given for selected action according to user-defined goal, then the Q-value for the respective state-action pair is updated. Now the error between predicted and updated Q-values is fed back to the NN as a target value and the weights of the NN are updated by using backpropagation algorithm. The AUV is supposed to reach the target/ destination point within the limited time period. Present episode is terminated if it exceeds the time limit, or if collision occurs or if the AUV reaches its destination point. However, due to modified Q-learning algorithm, the chances of AUV's collision with the obstacles will be reduced, avoiding wear and tear of AUV using Algorithm 2 and 3 and flowchart in Fig. 4.

At the start of every episode, AUV detects its own position and orientation. Then AUV tries to explore or exploit to reach the destination depending on previously updated Q-learning based NN output and action selection policy, this process always followed until AUV is in the safe region. When AUV detects unsafe region then the next Q-value is updated to not to follow the same path

Table 3 Parameters for NN

Approximation function	tansig
Back propagation function	Gradient decent 'traingdx'
Number of inputs	4
Number of outputs	6
Hidden layers	100

Table 4 Comparison of standard Q-learning and modified Q-learning

Number of iterations	Number of collisions using standard one-step Q-learning algorithm	Number of collisions using modified Q-learning algorithm
10	5	1
100	37	4
1000	218	11

in order to avoid the obstacle and force exploitation is carried out as shown in the flowchart in Fig. 4 and Algorithm 2. Once AUV gets out of unsafe region then it will again try to follow the optimal path with minimum distance criteria and by maximizing cumulative reward. After training the AUV, the future maneuvering task in unknown environments can be carried out using the resulting policy.

4 Simulation Results

REMUS AUV model mentioned in Appendix A is simulated as a plant in MATLAB software and hydrodynamic coefficients values for simulation are taken from (Prestero 2001). Following parameters are taken into consideration for standard and modified Q-learning algorithm as shown in Tables 1, 2, and 3.

The first simulation is carried out as a learning process for single obstacle avoidance where initial and final points are same as Table 1. It has been observed that the AUV takes the optimal straight line path without obstacle.

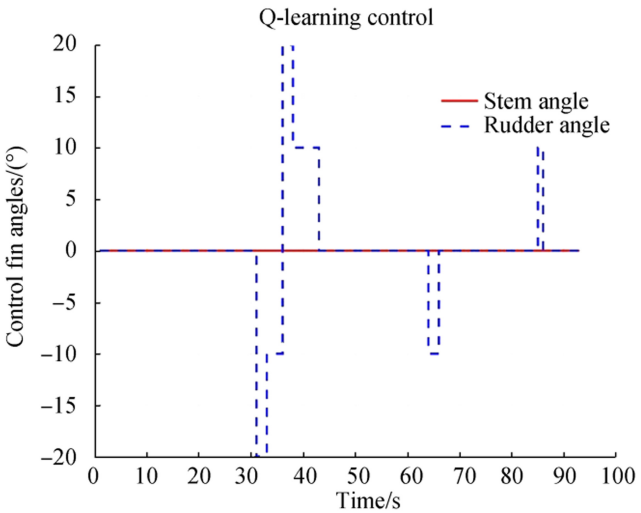

Fig. 6 Control signal for single obstacle avoidance

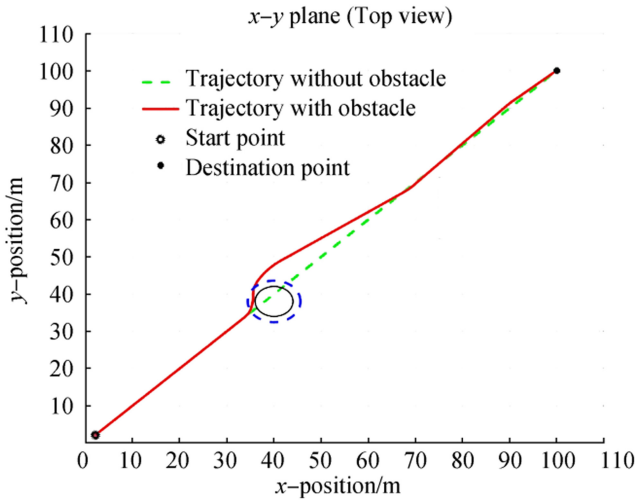

Fig. 7 Single obstacle avoidance using modified Q-learning

When a single obstacle is introduced, the comparison of the modified Q-learning algorithm with the standard one-step Q-learning algorithm is shown in Table 2 where AUV successfully evade unsafe region using the modified Q-learning algorithm with minimum collision and failed attempts as compared with the standard one-step Q-learning algorithm. Standard Q-learning has the strategy of exploration without concerning unsafe region and has more number collision with the obstacle, while on the other hand modified Q-learning algorithm considers unsafe region around the obstacle and reduces the collision with obstacles using force exploitation process as shown in Table 4.

The learned control signal for a single obstacle is as shown in Fig. 6. AUV tries to get out of the unsafe region in its path

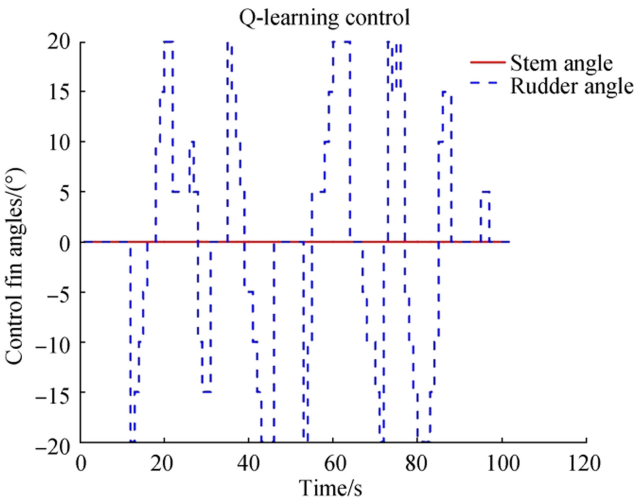

Fig. 8 Control signal for multiple obstacle avoidance

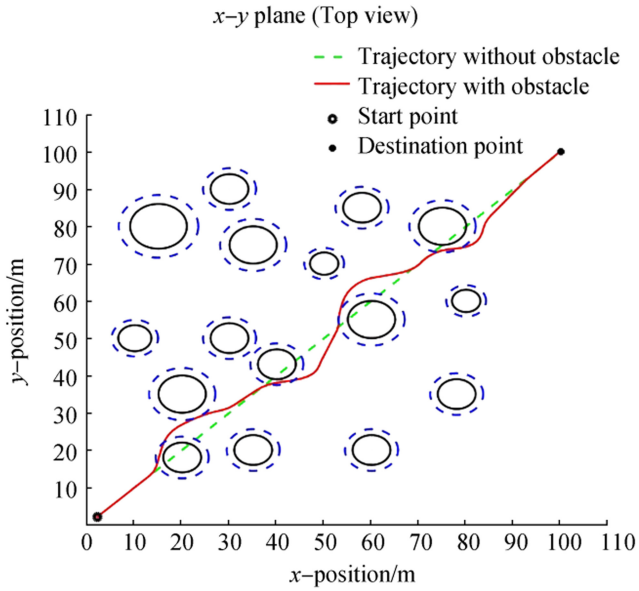

Fig. 9 Multiple obstacle avoidance using modified Q-learning

and after coming out of unsafe region the AUV attempts to follow the optimal path as shown in Fig. 7 with minimum error.

After learning process completed, the AUV is subjected to unknown environment and with multiple obstacles and it is observed that AUV successfully reaches the destination point avoiding the obstacles by taking control decisions as shown in Fig. 8 and the obstacle avoidance is shown in Fig. 9.

5 Conclusions

A modified Q-learning algorithm is utilized in obstacle avoidance and motion planning for set-point tracking problem using reward function and force exploitation in unsafe areas for AUV. The comparison of standard and modified algorithm shows modified algorithm performs better than standard algorithm since it reduces the chances of collision with obstacles. Furthermore, approximating functions are utilized in the form of NN to deal with the curse of dimensionality and continuous state space problem. From the simulation results, it can be concluded that the modified Q-learning algorithm can deal with obstacles in AUV navigation scenario without having any prior knowledge of the AUV dynamics and environment. This can solve the problems arising in AUV exploration in unknown and hostile areas where destination point can be reached with the self-learning, optimal path and obstacle avoidance with minimum wear and tear for AUV.

Acknowledgements The authors would like to acknowledge the support of Centre of Excellence (CoE) in Complex and Nonlinear dynamical system (CNDS), through TEQIP-II, VJTI, Mumbai, India.

Appendix 1

The forward motion of AUV in the horizontal plane is referred to as surge (x) (longitudinal motion), sidewise motion is referred as sway (y) (latitudinal motion) and angular motion about the vertical axis is referred as yaw (ψ). The remaining three DOFs are heave (x) (vertical motion), pitch (θ) (rotation about the transverse axis), and roll (ϕ) (rotation about the longitudinal axis) (Fossen 2011).

The following mathematical model for AUV is adopted as a plant to mimic the behavior of AUV motion for given control command:

Kinematics

The 6 DOF kinematic equations for the body to north-east-down (x-y-z in our case) transformation using Euler's Theorems and SNAME's notation (Fossen 2011) for the position $[x, y, z, \phi, \theta, \psi]^{\mathrm{T}}$ and velocity $[u, v, w, p, q, r]^{\mathrm{T}}$ are as follows:

$$
\begin{aligned}
\dot{x} &= [\cos(\theta)\cos(\psi)]u + [\cos(\psi)\sin(\phi)\sin(\theta) - \cos(\phi)\sin(\psi)]v + [\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta)]w \\
\dot{y} &= [\cos(\theta)\sin(\psi)]u + [\cos(\phi)\cos(\psi) + \sin(\theta)\sin(\phi)\sin(\psi)]v + [\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)]w \\
\dot{z} &= -[\sin(\theta)]u + [\sin(\phi)\cos(\theta)]v + [\cos(\phi)\cos(\theta)]w \\
\dot{\phi} &= p + [\sin(\phi)\tan(\theta)]q + [\cos(\phi)\tan(\theta)]r \\
\dot{\theta} &= \cos(\phi)q - \sin(\phi)r \\
\dot{\psi} &= \left[\frac{\sin(\phi)}{\cos(\theta)}\right]q - \left[\frac{\cos(\phi)}{\cos(\theta)}\right]r, \theta \neq 90^{\circ}
\end{aligned}
\tag{A1}
$$

Dynamics

The dynamic equations of motion can be stated as follows:

$$
\begin{aligned}
& m\left[\dot{u} - vr + wq - x_g\left(q^2 + r^2\right) + y_g\left(pq - \dot{r}\right) + z_g\left(pr + \dot{q}\right)\right] = X_{\text{total}} \\
& m\left[\dot{v} - wp + ur - y_g\left(r^2 + p^2\right) + z_g\left(qr + \dot{p}\right) + x_g\left(qp + \dot{r}\right)\right] = Y_{\text{total}} \\
& m\left[\dot{w} - uq + vp - z_g\left(p^2 + q^2\right) + x_g\left(rq - \dot{q}\right) + y_g\left(rq + \dot{p}\right)\right] = Z_{\text{total}} \\
& I_x\dot{p} + \left(I_z - I_y\right)qr - \left(\dot{r} + pq\right)I_{xz} + \left(r^2 - p^2\right)I_{yz} + \left(pr - \dot{q}\right)I_{xy} + m\left[y_g\left(\dot{w} - uq + vp\right) - z_g\left(\dot{v} - wp + ur\right)\right] = K_{\text{total}} \\
& I_y\dot{p} + \left(I_x - I_z\right)rp - \left(\dot{p} + qr\right)I_{xy} + \left(p^2 - r^2\right)I_{zx} + \left(qp - \dot{r}\right)I_{yz} + m\left[z_g\left(\dot{u} - vr + wq\right) - x_g\left(\dot{w} - uq + vp\right)\right] = M_{\text{total}} \\
& I_z\dot{r} + \left(I_y - I_x\right)pq - \left(\dot{q} + rp\right)I_{yz} + \left(q^2 - p^2\right)I_{xy} + \left(rp - \dot{p}\right)I_{zx} + m\left[x_g\left(\dot{v} - wp + ur\right) - y_g\left(\dot{u} - vr + wq\right)\right] = N_{\text{total}}
\end{aligned}
\tag{A2}
$$

Here, m is the mass of the craft, $[u, v, w]^{\mathrm{T}}$are linear velocities, $[p, q, r]^{\mathrm{T}}$ are angular velocities, $I_{(.)}$ is inertia in particular axial or cross-axial direction and (X_{total}, Y_{total}, Z_{total}) and (K_{total}, M_{total}, N_{total}) are total external forces in linear and angular directions. The first three equations in (A1) and (A2) represents the translational motion, while the last three equation represents the rotational motion.

The external forces in rigid body equation of motion shown in (A2) plays important role in the modeling of an underwater vehicle. In these forces and coefficients are explained including restoring forces, hydrodynamic forces, truster forces and lift forces due rudder (δ_r) and stern (δ_s) elevation. The total external force will be calculated as:

$$
\begin{aligned}
\tau_{\text{total}} = {} & \text{Hydrostatic Force} + \text{Hydrodynamic Force} \\
& + \text{Added Mass Force} + \text{Body Lift Force} \\
& + \text{Fin Lift force} + \text{Propeller Thrust}
\end{aligned}
$$

Therefore, the external forces in rigid body equation of motion shown in Eq. (A2) can be calculated as

$$
\begin{aligned}
X_{\text{total}} &= X_{HS} + X_{u|u|}u|u| + X_{\dot{u}}\dot{u} + X_{wq} + X_{qq}qq + X_{vr}vr + X_{rr}rr + X_{\text{prop}} \\
Y_{\text{total}} &= Y_{HS} + Y_{v|v|}v|v| + Y_{r|r|}r|r| + Y_{\dot{v}}\dot{v} + Y_{\dot{r}}\dot{r} + Y_{ur}ur + Y_{wp}wp + Y_{pq}pq + Y_{uv}uv + Y_{uu\delta_r}u^2\delta_r \\
Z_{\text{total}} &= Z_{HS} + Z_{w|w|}w|w| + Z_{q|q|} + Z_{\dot{w}}\dot{w} + Z_{\dot{q}}\dot{q} + Z_{uq}uq + Z_{vp}vp + Z_{rp}rp + Z_{uw}uw + Z_{uu\delta_r}u^2\delta_s \\
K_{\text{total}} &= K_{HS} + K_{p|p|}p|p| + K_{\dot{p}}\dot{p} + K_{\text{prop}} \\
M_{\text{total}} &= M_{HS} + M_{w|w|}w|w| + M_{q|q|}q|q| + M_{\dot{w}}\dot{w} + M_{\dot{q}}\dot{q} + M_{uq}uq + M_{vp}vp + M_{rp}rp + M_{uw}uw + M_{uu\delta_s}u^2\delta_s \\
N_{\text{total}} &= N_{HS} + N_{v|v|}v|v| + N_{r|r|}r|r| + N_{\dot{v}}\dot{v} + N_{\dot{r}}\dot{r} + N_{ur}ur + N_{wp}wp + N_{pq}pq + N_{uv}uv + N_{uu\delta_r}u^2\delta_r
\end{aligned}
\tag{A3}
$$

Hence, (A1) to (A3) represents complete 12 order 6 degree of freedom equations of motion of AUV which can be used for AUV simulations. The hydrodynamic coefficients, inertial constants, and physical parameters are taken from (Prestero 2001).

References

Bhopale P, Bajaria P, Kazi F, Singh N (2016) LMI based depth control for autonomous underwater vehicle. International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, India, 477–481

Bhopale P, Bajaria P, Kazi F, Singh N (2017) Enhancing reduced order model predictive control for autonomous underwater vehicle. In: Le NT, van Do T, Nguyen N, Thi H (eds) Advanced computational methods for knowledge engineering. ICCSAMA 2017. Advances in intelligent systems and computing, vol 629. Springer, Cham, 60–71

Cheng X, Qu J, Yan Z, Bian X (2010) H∞ robust fault-tolerant controller design for an autonomous underwater vehicle's navigation control system. J Mar Sci Appl 9(1):87–92. https://doi.org/10.1007/s11804-010-8052-x

Council, National Research (1996) Underwater vehicles, and national needs. National Academies Press, Washington, DC, 1–6

Fossen T (2011) Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons Ltd. Publication, 6–78

Hafner R, Riedmiller M (2014) Reinforcement learning in feedback control: challenges and benchmarks from technical process control. Mach Learn 84(1–2):137–169. https://doi.org/10.1007/s10994-011-5235-x

Kober J, Andrew B, Jan P (2013) Reinforcement learning in robotics: a survey. Int J Robotics Res 32(11):1238–1274. https://doi.org/10.1177/0278364913495721

Paula M, Acosta G (2015) Trajectory tracking algorithm for autonomous vehicles using adaptive reinforcement learning. Oceans 2015, Washington, DC, 1–8

Phanthong T, Maki T, Ura T, Sakamaki T, Aiyarak P (2014) Application of A* algorithm for real-time path re-planning of an unmanned surface vehicle avoiding underwater obstacles. J Mar Sci Appl 13(1):105–116. https://doi.org/10.1007/s11804-014-1224-3

Powell W (2007) Approximate dynamic programming: solving the curses of dimensionality. John Wiley and Sons Publication, 1–25

Prestero T (2001) Verification of six-degree of freedom simulation model for the REMUS autonomous underwater vehicle, MSc/ME Thesis. Massachusetts Institute of Technology, Cambridge, 1–78

Qu Y, Xu H, Yu W, Feng H, Han X (2017) Inverse optimal control for speed-varying path following of marine vessels with actuator dynamics. J Mar Sci Appl 16(2):225–236. https://doi.org/10.1007/s11804-017-1410-1

Russell B, Veerle A, Timothy P, Bramley J, Douglas P, Brian J, Henry A, Kirsty J, Jeffrey P, Daniel R, Esther J, Stephen E, Robert M, James E (2014) Autonomous underwater vehicles (AUVs): their past, present and future contributions to the advancement of marine geoscience. Mar Geol 352:451–468. https://doi.org/10.1016/j.margeo.2014.03.012

Su Y, Zhao J, Cao J, Zhang G (2013) Dynamics modeling and simulation of autonomous underwater vehicles with appendages. J Mar Sci Appl 12(1):45–51. https://doi.org/10.1007/s11804-013-1169-6

Sutton R, Barto A (1998) Introduction to reinforcement learning. MIT Press, Cambridge, MA, USA, pp 1–150

Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292. https://doi.org/10.1007/BF00992698

Yoo B, Kim J (2016) Path optimization for marine vehicles in ocean currents using reinforcement learning. J Mar Sci Technol 21(2):334–343. https://doi.org/10.1007/s00773-015-0355-9