DOI: 10.48011/asba.v2i1.1683

# Uma Abordagem de Aprendizado *on-line* para o Seguimento de Trajetórias usando Robôs não Holonômicos

**Mateus Sousa Franco** [*], **Sérgio R. Barros dos Santos** [*], **Fabio Augusto Faria** [*]

[*] *Instituto de Ciência e Tecnologia*
*Universidade Federal de São Paulo*
*São José dos Campos - SP.*
*E-mails: {mateus.franco, sergio.ronaldo, ffaria}@unifesp.br*

**Abstract:** This study investigates a Reinforcement Learning (RL) method to derivate control laws of a non-holonimic robot considering the coupling and nonlinearity of the system. The controller is on-line derivated from the interaction between the agent and an unknown environment through a Q-learning based approach. This approach aims to find the best action that maximizes the rewards along attempts to follow a trajectory. Performed experiments might show that the learned controllers were able of efficiently following diverse trajectories considering different speed variations of the robot translation and rotation as well as maximizing the reward amount over iteractions for two distinct learning process configurations.

**Resumo**: Este artigo investiga a aplicação de um método de Aprendizado por Reforço (RL) para derivar as leis de controle de robôs não holonômico, considerando o acoplamento e a não linearidade do sistema. Os controladores são derivados *on-line* através da interação entre o agente real e o ambiente desconhecido, usando uma abordagem baseada no algoritmo *Q-Learning*, que visa descobrir qual a melhor ação a ser tomada pelo agente, de modo a maximizar as recompensas recebidas em cada tentativa de execução do seguimento da trajetória desejada. Resultados experimentais mostraram que os controladores aprendidos são capazes de realizar o seguimento de diferentes trajetórias, de forma eficiente, levando em conta a variação das velocidades de translação e de rotação do robô e a maximização do valor das recompensas ao longo das iterações, conforme apresentado nos estudos de caso.

*Keywords:* Computer Vision; PID Controller; Q-learning; Reinforcement Learning
*Palavras-chaves:* Aprendizado por Reforço; Controlador PID; Q-learning; Visão Computacional

## 1. INTRODUÇÃO

As técnicas clássicas de projeto de controladores envolvem a obtenção de um modelo matemático do sistema a ser controlado, no qual é representado por expressões que descrevem e modelam o seu comportamento. Esta etapa tende a ter uma complexidade que varia de acordo com o tipo de sistema e com as condições estabelecidas para a operação, podendo levar a modelos matemáticos extremamente complexos, mesmo que lineares, e não necessariamente precisos (Bekey, 2005).

Um tipo de controlador amplamente utilizado devido à sua simplicidade de otimização e implementação é o controlador Proporcional-Integral-Derivativo (PID). Esse controlador gera uma saída de controle proporcional ao erro entre o estado desejado e o estado atual (ganho proporcional), levando em consideração também a taxa de variação do erro (ganho derivativo) e o erro acumulado ao longo do tempo (ganho integral). O PID é um controlador geralmente empregado em sistemas lineares, porém também pode ser utilizado em sistemas não lineares (Wilson et al., 2016), desde que esse sistema opere dentro de uma faixa especifica de operação (de Sousa Neves, 2009; Howell, 2000; Meenakshi, 2008).

O Aprendizado por Reforço (do inglês, *Reinforcement Learning* (RL)), é definido como um método de otimização que pode ser aplicado em problemas de controle de processos, onde os modelos dos sistemas não são conhecidos a priori. A partir da interação com ambiente aleatório e desconhecido, o agente aprende estratégias adaptativas de controle para as diferentes situações, por meio de recompensas que sinalizam se o agente tomou a ação correta ou incorreta. O objetivo do agente é acumular o máximo de reforço positivo possível durante a exploração do ambiente (Carlucho et al., 2017; Sutton, 1999; Kober et al., 2014).

Os métodos baseados em RL possuem grande potencial no âmbito da robótica móvel, principalmente, da perspectiva dos sistemas de controle, uma vez que robôs móveis são classificados como sistemas não lineares, cujo comportamento possui um elevado acoplamento e um alto grau de incerteza associada à determinação dos estados, que são obtidos a partir dos sensores. Como os sistemas não lineares são matematicamente muito complexos, a tarefa de projetar controladores utilizando seus modelos mate-

máticos é extremamente complicada. Neste contexto, uma forma de superar esse problema é realizar o aprendizado dos controladores de forma *on-line*, i.e., diretamente em robôs reais (sistemas não lineares), de modo que os controladores possam ser sintonizados de modo interativo levando em conta o acoplamento e a não linearidade desses sistemas (Carlucho et al., 2017).

O objetivo deste trabalho é implementar um algoritmo de controle não linear baseado na abordagem de RL *on-line*, que seja capaz de controlar o agente em distintas faixas de operação, estabelecidas ao longo do seguimento de diferentes trajetórias. A partir da interação do agente com o ambiente, o algoritmo RL busca encontrar um controlador, de forma interativa, que seja capaz de executar o trajeto desejado com precisão e eficiência, para diferentes velocidades de translação e de rotação. O aprendizado ocorre no próprio robô através do algoritmo de RL embarcado e do *feedback* fornecido pelo sistema de visão, que observa os estados no ambiente. A informação visual obtida é usada para calcular o sinal de reforço aplicado a cada episódio.

O artigo está organizado da seguinte forma: Na Seção 2, é apresentada uma descrição prévia do modelo do robô e do algoritmo de RL usado neste trabalho. A Seção 3 é dedicada à apresentação da estratégia proposta para o processo de aprendizagem *on-line* do sistema de controle. Na Seção 4, nós apresentamos os resultados obtidos para cada estudo de caso. Para finalizar, as conclusões são apresentadas na Seção 5.

## 2. FUNDAMENTAÇÃO

Nesta seção são apresentados os conceitos básicos usados ao longo deste trabalho tal como o Aprendizado por Reforço na subseção 2.1, e a modelagem cinemática do robôs não holonômicos na subseção 2.2.

### *2.1 Aprendizado por Reforço*

O Aprendizado por Reforço também conhecido como *Reinforcement Learning* (RL) é um método de aprendizagem de máquina, que permite ao agente (qualquer sistema cyberfísico) descobrir, sem prévio conhecimento do ambiente, quais as melhores ações a serem tomadas de modo a maximizar a recompensa recebida em cada tentativa de realização da missão desejada (Sutton, 1999). Este paradigma é diferente, por exemplo, do Aprendizado Supervisionado, que conta com conhecimento prévio do ambiente (mundo real) durante etapa de treinamento (Sutton, 1999; Kober et al., 2014; Carlucho et al., 2017; Campbell et al., 2016).

De acordo com Sutton (1999), o RL consiste de uma arquitetura formal que define a interação entre o agente e seu ambiente da seguinte forma:

- $\Phi$ é um conjunto discreto de estados do ambiente;
- $\alpha$ é um conjunto discreto de ações que pode ser tomado pelo agente em cada estado $s \in \Phi$;
- $R$ é o sinal de reforço (recompensas) normalmente entre $\{0,1\}$ que pode ser calculado para uma determinada ação $a \in \alpha$.

O RL pode ser matematicamente formalizado usando o chamado *Processo de Decisão Finito de Markov* (MDP), que é uma formalização clássica dos problemas de tomada de decisão, onde a ação escolhida não influencia apenas as recompensas imediatas, mas também os estados e as recompensas futuras (Sutton, 1999; Pellegrini, 2007; Campbell et al., 2016).

A política de ações é o elemento que estabelece o mapeamento entre um dado estado e a ação executada, baseado em probabilidades (Kober et al., 2014), e define o modelo de comportamento do agente. Se o agente está seguindo a política $\pi$ em um tempo $t$, então $\pi(a|s)$ é a probabilidade que $A_t = a$ se $S_t = s$. Dada uma política $\pi$, é possível definir formalmente a função ação-valor $q_\pi(a,s)$ como o valor de tomar uma ação $a$ em um estado $s$:

$$q_\pi(a,s) = \mathbb{E}[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a], \forall s \in \Phi, \forall a \in \alpha \quad (1)$$

Resolver a tarefa do RL significa encontrar a política ótima para o MDP que maximize a recompensa total ao longo do tempo. Considere-se uma política $\pi$ definida como melhor ou igual à política $\pi'$ para todos os estados. Sendo assim, pode-se inferir que $\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v_{\pi'}(s) \forall s \in \Phi$. Sempre há uma política que é melhor que as outras, essa é a política ótima $\pi_*$; no entanto nem sempre é única. Todas as políticas ótimas compartilham as mesmas funções de estado-valor ótima, $v_*(s)$, e de ação-valor ótima, $q_*(s,a)$ (Sutton, 1999).

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in \Phi \quad (2)$$

$$q_*(s,a) = \max_\pi q_\pi(s,a), \forall s \in \Phi, \forall a \in \alpha \quad (3)$$

Ainda é possível escrever $q_*(s,a)$ em termos de $v_*(s)$, de modo a obter uma função que fornece o retorno esperado para um par estado-ação seguindo a política ótima (Sutton, 1999).

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma . v_*(S_{t+1})|S_t = s, A_t = a] \quad (4)$$

Especificamente, o *Q-learning* pode ser usado para encontrar uma política de ações para qualquer MDP. Nessa abordagem, o agente (robô) aprende uma função ação-valor (*Q-function*) ótima $q_*$ de forma direta, independente da política vigente, que define a utilidade esperada de tomar uma determinada ação ($a$) em um dado estado ($s$) (Watkins, 1992). Assim, em cada iteração ou episódio, o algoritmo escolhe uma ação $a$ tal que a função $Q(s,a)$ seja maximizada.

$$Q(s,a) \leftarrow Q(s,a) + \beta[R(s) + \gamma \max_a Q(s_1,a_1) - Q(s,a)] \quad (5)$$

onde $R(s)$ o sinal de reforço imediato, $Q(s_1,a_1)$ é o valor da função $Q$ referente ao próximo estado $s_1$ resultante da ação $a_1$, $\beta$ é a taxa de aprendizagem e $\gamma$ refere-se ao parâmetro de desconto para as futuras recompensas no intervalo real $[0,1)$.

### *2.2 Modelagem Cinemática de Robôs não Holonômicos*

Considere o ponto $P$ centrado entre as rodas 1 (esquerda) e 2 (direita) que possuem um raio $r$ e estão localizadas à uma distância $l$ de $P$, com velocidades angulares $\dot{\varphi}_1$ e $\dot{\varphi}_2$ associadas. Uma vez que estes valores são conhecidos, é possível estabelecer o modelo matemático que define a

cinemática dos movimentos de translação e rotação do robô em relação ao referencial inercial, que pode ser visto na Equação 6 (Siegwart, 2004).

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\varphi}_1, \dot{\varphi}_2) \tag{6}$$

Supõe-se que o robô esteja alinhado ao eixo $X_R$ movendo-se no sentido positivo, ou seja, com a orientação $\theta = 0°$ e considere que a roda 1 está girando com velocidade angular $\dot{\varphi}_1$, enquanto a roda 2 permanece parada. Então, a roda 1 irá transladar com uma velocidade tangencial $\dot{x}_{r1} = \dot{\varphi}_1 r$ e fará com que o robô faça um movimento circular de raio $2l$ cujo centro de rotação é o ponto de contato da roda 2 com a superfície.

Como o ponto $P$ está exatamente na metade da distância entre a roda 1 e o centro de rotação, a sua velocidade de translação $\dot{x}_r$ é a metade da velocidade de translação da roda, conforme pode ser visto na Equação 7. Analogamente, pode-se adotar o mesmo raciocínio para o caso em que a roda 1 está parada e a roda 2 em movimento, dando origem à Equação 8.

$$\dot{x}_{r1} = \frac{r\dot{\varphi}_1}{2} \tag{7}$$

$$\dot{x}_{r2} = \frac{r\dot{\varphi}_2}{2} \tag{8}$$

A expressão que determina $\dot{x}_r$ é uma combinação dos efeitos causados pela rotação das duas rodas e pode ser obtida através da soma das duas expressões, conforme pode ser visto na Equação 9.

$$\dot{x}_r = \dot{x}_{r1} + \dot{x}_{r2} = \frac{r\dot{\varphi}_1}{2} + \frac{r\dot{\varphi}_2}{2} \tag{9}$$

Considerando-se que as rodas possuem a mesma velocidade angular, porém sentidos de rotação opostos. Neste caso, o robô gira sobre ponto $P$, de forma estacionaria, de modo que $\dot{x}_r$ é nulo. O valor de $\dot{y}_r$ será sempre nulo, uma vez que nenhuma das rodas contribuiu com movimentos laterais.

Se a roda 1 (direita) gira num sentido que move o robô para frente enquanto a outra está parada, o ponto $P$ irá girar no sentido horário cujo centro de rotação é a roda parada. A velocidade angular de $P$ pode ser calculada pela Equação 10.

$$\omega_1 = \frac{r\dot{\varphi}_1}{2l} \tag{10}$$

Uma situação análoga pode ser utilizada para derivar a expressão da velocidade angular de $P$ a partir da velocidade angular da roda 2 (esquerda). Basicamente, apenas o sinal muda já que o sentido de rotação se inverte, conforme pode ser observado na Equação 11.

$$\omega_2 = -\frac{r\dot{\varphi}_2}{2l} \tag{11}$$

A combinação destas expressões individuais dá origem ao modelo cinemático do robô de duas rodas, dado pela equação 12.

$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} \frac{r\dot{\varphi}_1}{2} + \frac{r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_1}{2l} - \frac{r\dot{\varphi}_2}{2l} \end{bmatrix} \tag{12}$$

onde $R(\theta)^{-1}$ refere-se a matriz de rotação ortogonal usada para transformação do sistema de coordenadas do robô para o sistema de coordenadas inercial.

## 3. DESENVOLVIMENTO DO SISTEMA

Nesta seção, nós discutimos a implementação do sistema proposto para realizar o aprendizado do seguimento de trajetórias diretamente no robô real.

### *3.1 Arquitetura Proposta*

A Figura 1 mostra o diagrama de bloco completo do sistema proposto, incluindo a interconexão de cada bloco funcional. O módulo denominado de Unidade de Visão Computactional e Aprendizado (CVLU) é formado pelos algoritmos de visão computacional e Q-learning, e também pela interface de comunicação serial usada para realizar a troca de informações com a Unidade de Controle de Motores (MCU) e com o módulo de Servidor Web (WU).

A MCU é usada para parametrizar o funcionamento dos dois motores independentes do robô através das instruções enviadas pela CVLU. Por sua vez, o WU é responsável pelo armazenamento de todos os dados do processo de aprendizagem, implementado em uma interface Web que permite consultar os dados dos treinamentos anteriores e acompanhar em tempo real o progresso do novo treinamento.

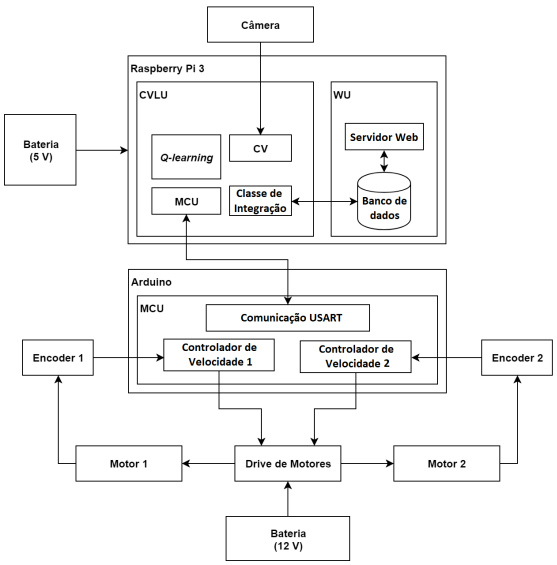


Figura 1. Diagrama de blocos dos subsistemas implementado.

Neste trabalho, por se tratar de um estudo prático (em um cenário real), é praticamente impossível assumir que as observações dos estados do agente são absolutas e

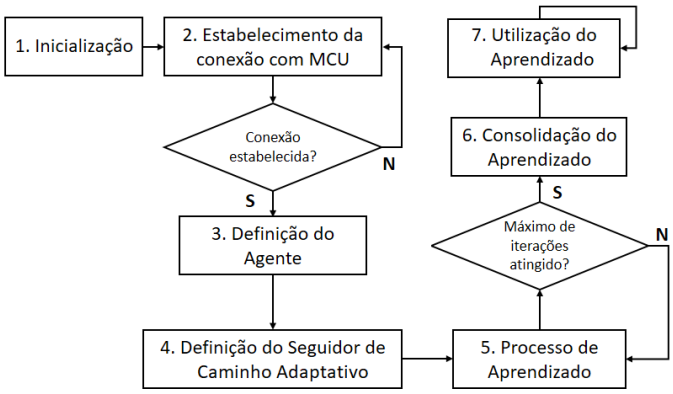


Figura 2. Ciclo de funcionamento implementado para o robô seguidor de caminho adaptativo.

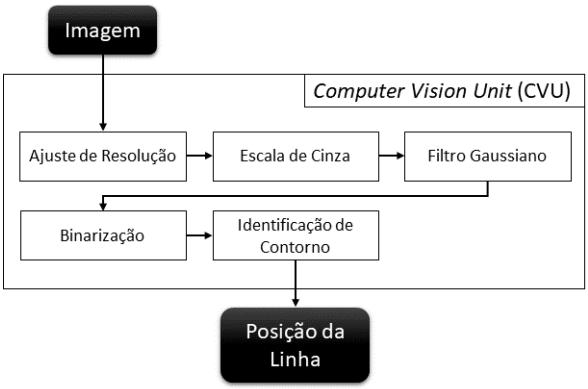


Figura 3. Fluxograma do processamento realizado para cada imagem capturada.

livres de ruídos, uma vez que esses estados são obtidos através de dois encoders com resolução de 341 PPR e um sensor óptico (câmera) IMX219 8-megapixel. Neste contexto, os filtros digitais (Butterworth e média móvel) foram utilizados para reduzir os ruídos e evitar leituras incorretas. A Figura 2 mostra o fluxograma das principais operações realizadas durante o processo de aprendizado.

Inicialmente, as variáveis que definem os estados, as ações e as recompensas são inicializadas conforme a configuração adotada para o treinamento. Em seguida, é iniciada a comunicação serial entre a CVLU e a MCU. Quando a comunicação é estabelecida, o programa segue para o passo seguinte, que consiste na definição das estruturas de dados necessárias para execução do algoritmo *Q-learning*. Uma vez definido, o processo de aprendizado é iniciado e permanece em execução até que todas as iterações sejam executadas, treinando o controlador de translação e orientação do robô. Por fim, ocorre a consolidação do aprendizado, fazendo com que o robô execute a política de ações ótima aprendida para realizar o seguimento das trajetórias.

A CVLU e MCU foram implementados em hardware distintos a fim de paralelizar a execução dos programas principais e garantir um melhor desempenho do processo de aprendizado. Para tal, foi necessário criar um protocolo de comunicação padronizado entre os dois dispositivos para possibilitar que as ações selecionadas pela CVLU fossem enviadas e executadas pela MCU.

### *3.2 Sistema de Visão Computacional*

Neste trabalho, o sistema de visão computacional é utilizado para identificar o caminho a ser seguido (definido por meio de faixas no ambiente) e estimar o desvio de posição através da contagem dos pixels entre centro da faixa e o centro da imagem. Se o centro da faixa aparece à esquerda com relação ao centro da imagem, o robô encontra-se à direita da faixa, caso contrário, o robô está a esquerda da faixa. Para seguir corretamente o caminho, o robô deve manter seu centro com o menor desvio possível em relação ao centro da faixa detectada pela sistema de visão.

A Figura 3 mostra o procedimento implementado para realizar o processamento da imagem capturada durante o movimento de translação do robô.

A imagem obtida é cortada em uma região específica para otimizar o processamento. Em seguida, esta imagem é convertida para escala de cinza e um filtro Gaussiano é aplicado para diminuir o ruído após a conversão. Por fim, faz-se uma binarização da imagem, que transforma *pixels* com valores acima de um limiar em branco e abaixo deste mesmo limiar em preto. Desta forma, basta identificar o polígono formado pela faixa e o seu centro, que possui uma coordenada na horizontal e na vertical. Como o propósito é ver o quão alinhado o robô está em relação à faixa demarcada, a coordenada horizontal é a mais relevante. Neste caso, o desvio pode ser determinado pela diferença entre as coordenadas horizontais do centro da faixa e do centro da imagem.

A Figura 4 ilustra a forma que o caminho a ser seguido é identificado e como o desvio é obtido a partir da imagem amostrada.


Figura 4. Processamento da imagem amostrada pela câmera da CVLU. (a) Amostragem, filtragem e binarização. (b) Cálculo do desvio.

### *3.3 Q-Learning On-line*

A estratégia de RL, baseada no algoritmo *Q-learning*, é proposta para derivar de forma *on-line* os controladores de translação e orientação para controlar o seguimento de caminho realizado pelo robô não holonômico, considerando o acoplamento e a não linearidade existente no sistema. Em cada iteração do algoritmo, o sistema de visão estima o desvio do robô em relação ao caminho a ser percorrido e, em seguida, calcula o estado atual da plataforma. Logo depois, as ações (velocidades dos motores) são escolhidas com base nos valores da tabela $Q$, e essas ações (comandos) são transmitidas para a Unidade de Controle de Motores (MCU). Na sequência, um novo estado é observado, a recompensa é calculada e a posição da tabela $Q$ correspondente ao estado anterior e as ações escolhidas são atualizadas (Hakim et al., 2013).

A Figura 5 contém um fluxograma das operações básicas executadas no contexto do processo do aprendizado. O processo acontece de forma cíclica sempre recomeçando pela determinação do estado atual.

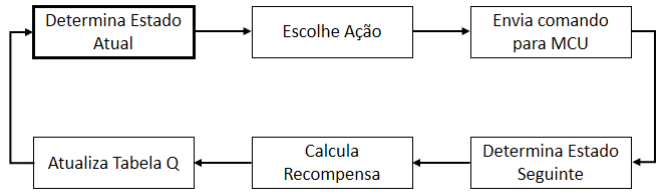


Figura 5. Fluxograma geral do algoritmo *Q-learning* implementado no computador embarcado.

A determinação dos estados é feita com base na posição horizontal do centro da faixa. Considere-se uma imagem com resolução $(w,h)$ *pixels*, sendo $w$ a quantidade de *pixels* na horizontal e $h$ a quantidade na vertical. Então, os possíveis valores de posição horizontal $x$ estão dentro do intervalo $[1,w]$, portanto $x = w/2$ é o centro da imagem.

O intervalo $[1,w]$ é dividido em $n-1$ partes, onde $n-1$ é a quantidade de estados que é possível identificar a faixa no campo de visão do robô. Cada estado é definido por um intervalo fechado $[s,f]$, onde $s$ indica o *pixel* de início da imagem e $f$ o *pixel* de término da imagem.

A Figura 6 contém uma ilustração que descreve a lógica de definição dos estados do agente para um caso em que $w = 300$ e $n = 4$.

Quando a faixa encontra-se fora do campo de visão do robô, o algoritmo de processamento da imagem retorna o valor zero, caracterizando em um estado onde o robô encontra-se sem referência de trajetória. Neste caso, ao fim da iteração, o *Q-learning* deverá selecionar outras ações que fará com que o robô encontre novamente o caminho.

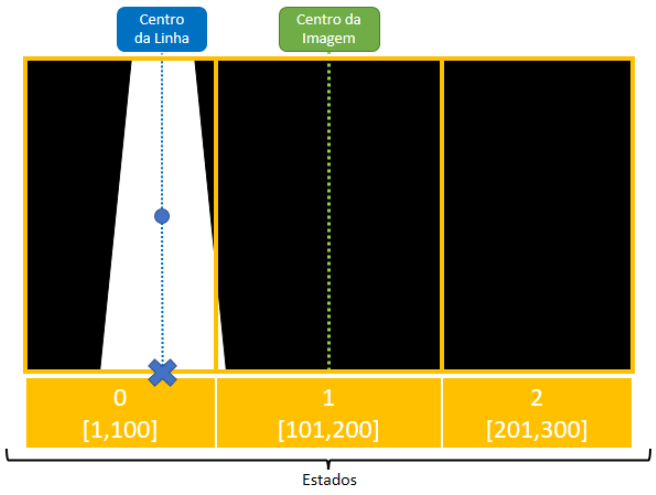


Figura 6. Delimitação dos estados para $w = 300$ e $n = 4$. O centro da faixa encontra-se no intervalo correspondente ao estado 0.

Os pares $[s,f]$ são armazenados em uma lista no programa principal da CVLU, conforme pode ser observado na Tabela 1 que ilustra um exemplo em que $w = 300$ e $n = 4$.

Tabela 1. Intervalos definidos para o caso com quatro estados.

| Índice | Intervalos |
|---|---|
| 0 | [1,100] |
| 1 | [101,200] |
| 2 | [201,300] |
| 3 | [0,0] |

Uma vez que o estado atual foi determinado, o programa escolhe as próximas ações a serem executadas tomando como base a tabela Q. Uma probabilidade no intervalo fechado [0,1] é sorteada e a linha da tabela $Q$ correspondente ao estado atual é varrida enquanto o valor acumulado de probabilidade das possíveis ações é menor do que a probabilidade sorteada. A escolha das ações na varredura realizada ocorre quando é encontrada uma probabilidade associada as ações que é maior que a probabilidade sorteada.

As ações são definidas como pares de valores $(rpm_1, rpm_2)$, onde o primeiro valor representa a velocidade angular do motor 1 (esquerdo) e o segundo a velocidade do motor 2 (direito), ambas em rotações por minuto. No caso do robô seguidor de caminho, é necessário haver pelo menos três ações: curva para a esquerda, seguir em frente, curva para a direita. A quantidade de ações impacta diretamente na forma de movimentação do robô nas curvas, pois permite realizar curvas com diferentes graus de intensidade.

A Tabela 2 contém um exemplo de combinações de velocidades para as três ações básicas do robô seguidor de caminho.

Tabela 2. Conjunto básico de ações do agente.

| Descrição | Ação | Motor 1 | Motor 2 |
|---|---|---|---|
| Curva (Esquerda) | 0 | 0 rpm | 100 rpm |
| Seguir em Frente | 1 | 100 rpm | 100 rpm |
| Curva (Direita) | 2 | 100 rpm | 0 rpm |

Um outro caso onde assume-se cinco ações é mostrada na Tabela 3. Cada uma das ações são armazenadas em uma lista onde o índice estabelecido equivale ao par de velocidades para os motores do robô.

Tabela 3. Conjunto estendido de ações do agente.

| Ação | Descrição | Motor 1 | Motor 2 |
|---|---|---|---|
| 0 | Giro anti-horário | -100 rpm | 100 rpm |
| 1 | Curva à Esquerda | 0 | 100 rpm |
| 2 | Seguir em frente | 100 rpm | 100 rpm |
| 3 | Curva à Direita | 100 rpm | 0 |
| 4 | Giro horário | 100 rpm | -100 rpm |

Para isso, o cálculo das recompensas deve ser feito considerando que o estado que fornece a maior recompensa encontra-se no centro da imagem. Quanto mais afastado do centro, menor a recompensa recebida pelo agente. A pior recompensa é recebida quando o robô acessa um estado onde a faixa está fora do campo de visão. Neste trabalho, optou-se por implementar o cálculo da recompensa através da associação direta entre o estado e o seu respectivo valor de recompensa, semelhante à lista de estados e ações. Entretanto, esse cálculo pode ser feito de outras formas, como por exemplo, utilizando uma função para calcular a recompensa de acordo com a distância entre o centro da

faixa e centro da imagem. A Figura 7 mostra como foi implementada a relação entre os estados e as recompensas para o caso com três ações.

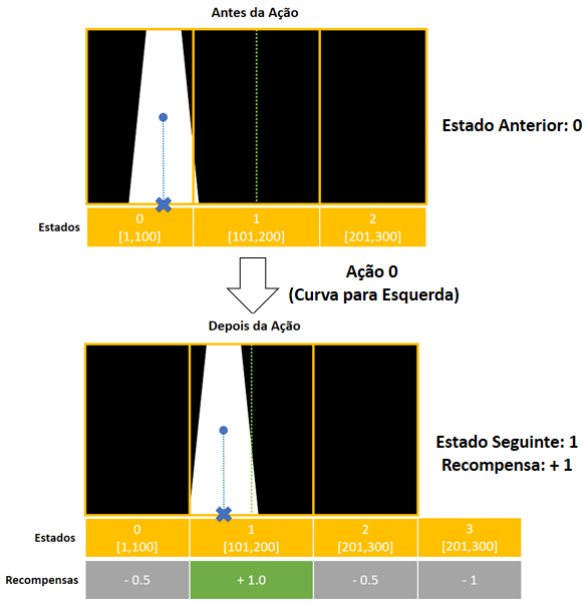


Figura 7. Determinação do estado atual, escolha e execução da ação e determinação do estado seguinte.

Após o término das iterações, a CVLU faz um processo de consolidação da política obtida. A tabela $Q$ do agente é varrida e um mapa é construído usando todos os estados e as ações que possuir o maior valor acumulado ao término do treinamento. Desta forma, ao invés de sempre realizar varreduras sobre a tabela $Q$, o robô pode utilizar a política aprendida num processo de aprendizado, consolidada na forma de um mapa estado-ação (controlador de translação e orientação), de modo mais eficiente e conciso, eliminando varreduras desnecessárias.

## 4. RESULTADOS

Os testes experimentais apresentados a seguir foram realizados em diferentes trajetos estabelecidos, formados por uma combinação de curvas com distintos graus de curvatura, e retas curtas e longas. A medida que o robô percorre cada um dos trajetos, o mesmo adquire conhecimento acumulativo a respeito do caminho percorrido, aprimorando a eficiência de execução do seguimento da trajetória e adaptando-se as diferentes situações. A Figura 8 mostra o agente robótico desenvolvido neste estudo.

### *4.1 Estudo de Caso I*

O primeiro estudo de caso utiliza a quantidade mínima de estados possíveis e de ações necessárias para realizar o seguimento das trajetórias. Neste caso, o processo de aprendizado possui quatro estados, com os quais é possível distinguir a posição do agente, como à esquerda da faixa, à direita da faixa, sobre a faixa e sem visão da faixa.

Definiu-se um conjunto básico de três ações que incluem os movimentos de curva à esquerda, curva à direita e seguir em frente. A ação de ficar parado foi desconsiderada, pois, além de não promover uma mudança de estados, não faria

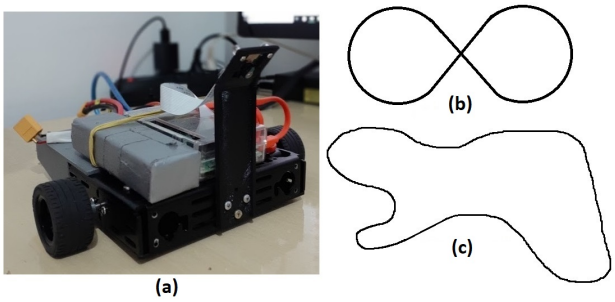


Figura 8. Composição do ambiente Experimental: a) robô utilizado neste estudo, b) circuito com cruzamento e c) circuito misto.

com que o robô avançasse no circuito. Os estados e ações definidos para esse estudo de caso encontram-se descritos na Tabela 4 e na Tabela 5, respectivamente. O processo de aprendizado foi configurado com $\beta = 0,1$, $\lambda = 0,9$ e 2200 episódios.

Tabela 4. Estados do Estudo de Caso I.

| Estado | Descrição | Início | Fim | Recompensa |
|---|---|---|---|---|
| 0 | Linha à Esquerda | 1 | 101 | -0,07 |
| 1 | Sobre a linha | 102 | 139 | 1,0 |
| 2 | Linha à Direita | 140 | 240 | -0,07 |
| 3 | Sem visão da linha | 0 | 0 | -1,0 |

Tabela 5. Ações do Estudo de Caso I.

| Ação | Descrição | Motor 1 | Motor 2 |
|---|---|---|---|
| 0 | Curva à Esquerda | 0 | 100 rpm |
| 1 | Seguir em Frente | 100 rpm | 100 rpm |
| 2 | Curva à Direita | 100 rpm | 0 |

A Figura 9 ilustra o gráfico dos valores finais dos pares estado-ação para cada um dos estados. Considerando o Estado 0, em que a faixa encontra-se a esquerda do robô, o par de maior valor foi (Estado 0, Ação 0), o que significa que o robô aprendeu que, se a faixa encontra-se a sua esquerda, ele deve fazer uma curva à esquerda. Também pode ser observado que os pares de maiores valores são (Estado 1, Ação 1) e (Estado 2, Ação 2), respectivamente. Considerando o Estado 3, o maior valor está associado ao par (Estado 3, Ação 2), significando que o robô aprendeu que a melhor ação é executar uma curva à direita no caso de perder a referência da faixa. Também é possível inferir que a ação de seguir em frente seria a pior opção, pois a ação possui um menor valor acumulado.

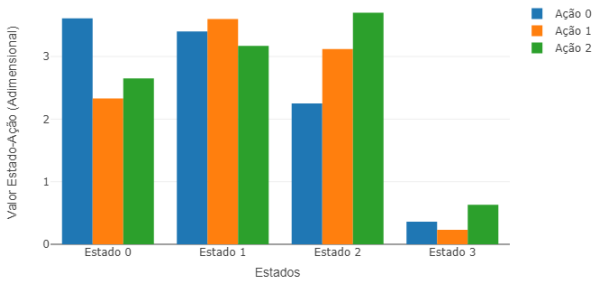


Figura 9. Valores finais (adimensionais) dos pares estado-ação para cada estado em função da ação no Estudo de Caso I.

A evolução da recompensa média acumulada ao longo das iterações encontra-se na Figura 10. É possível notar nesse

gráfico que a partir do episódio 1600, a recompensa média atinge um valor aproximadamente constante, indicando que o algoritmo Q-learning aprendeu uma política de ações ótimo capaz de seguir os trajetos estabelecidos.

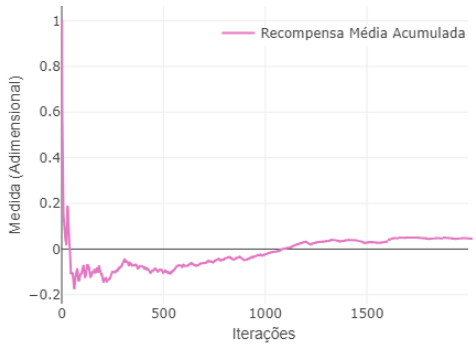


Figura 10. Recompensa Média Acumulada ao longo das iterações no Estudo de Caso I.

A Figura 11 mostra um mapa de calor que indica a quantidade de vezes que cada par estado-ação foi utilizado ao longo do processo. Nele é possível observar que os pares que tem o maior valor associado possuem mais acessos, exceto no caso do Estado 3. Com este mapa de calor, obteve-se a Tabela 6, que mostra o percentual de acesso para cada par de maior valor associado.

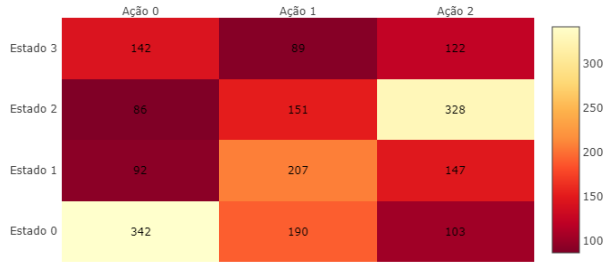


Figura 11. Mapa de Calor da quantidade de acesso aos pares estado-ação relativo ao Estudo de Caso I.

Tabela 6. Percentual de acessos dos pares estado-ação de maior valor no Estudo de Caso I.

| Estado | Total de Acessos | Ação com Maior Valor | Acessos da Ação | Percentual |
|---|---|---|---|---|
| 0 | 635 | Ação 0 | 342 | 53,8 % |
| 1 | 446 | Ação 1 | 207 | 46,4 % |
| 2 | 565 | Ação 2 | 328 | 58,1 % |
| 3 | 353 | Ação 2 | 122 | 34,6 % |

### *4.2 Estudo de Caso II*

O propósito deste estudo de caso é avaliar a capacidade do sistema para aprender um controlador adequado para um maior número de estados e ações, bem como observar os efeitos causados por essa mudança. O processo de aprendizado foi realizado usando seis estados e cinco ações. Assim como no estudo de caso anterior, a ação na qual mantém o robô parado é desconsiderada.

Os estados e suas respectivas recompensas encontra-se na Tabela 7, enquanto as ações estão descritas na Tabela 3.

Este processo foi configurado utilizando $\beta = 0{,}1$, $\lambda = 0{,}9$, 10000 episódios e durou cerca de 65 minutos.

Tabela 7. Estados do Estudo de Caso

| Estado | Posição da Linha | Início | Fim | Recompensa |
|---|---|---|---|---|
| 0 | Extrema Esquerda | 1 | 61 | -0,05 |
| 1 | Esquerda | 62 | 100 | -0,02 |
| 2 | Sobre a linha | 101 | 140 | 1,0 |
| 3 | Direita | 141 | 179 | -0,02 |
| 4 | Extrema Direita | 180 | 240 | -0,05 |
| 5 | Sem visão (perdido) | 0 | 0 | -1,0 |

Os valores finais dos pares estado-ação para cada um dos estados é ilustrado na Figura 12. Considerando o Estado 0 e o Estado 4, é possível observar que os pares de maior valor são (Estado 0, Ação 0) e (Estado 4, Ação 4), indicando que após o processo de aprendizado, o robô aprendeu que para os desvios mais acentuados deve tomar as ações correspondentes às curvas mais bruscas.

Analogamente, o mesmo pode ser observado para o Estados 1 e Estado 3, em que o robô apresenta um leve desvio de posição. Nestes casos, os pares de maior valor são (Estado 1, Ação 1) e (Estado 3, Ação 3), indicando que após o processo de aprendizado, o robô aprendeu que as ações correspondentes às curvas mais suaves são mais adequadas para desvios menos acentuados.

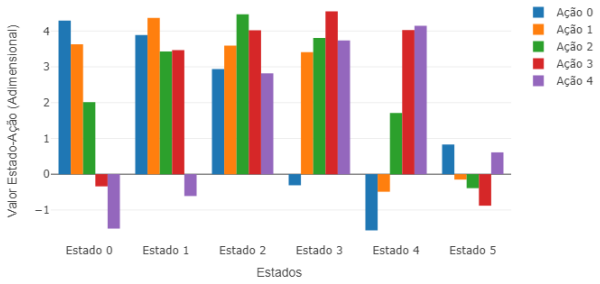


Figura 12. Valores estado-ação (adimensionais) para cada estado em função da ação no Estudo de Caso II.

Um fato interessante a se destacar é o resultado obtido para o Estado 5 em que os dois pares de maior valor obtidos foram (Estado 5, Ação 0) e (Estado 5, Ação 4), tornando possível inferir que o robô aprendeu que curvas bruscas faz com que ele retorne à faixa de forma mais eficiente do que curvas suaves, quando encontra-se perdido.

A recompensa média acumulada ao longo das iterações encontra-se no gráfico da Figura 13. Essa curva mostra que, inicialmente (até a episódio 400), o robô recebeu uma grande quantidade de penalidades, fazendo a média acumulada ser negativa. Após a iteração 900, o robô passou a receber mais gratificações do que penalidade, fazendo a curva de recompensa média aumentar ao longo do treinamento. Somente após 6500 iterações, o reforço médio recebido pelo agente passa a ser positivo. Então, próximo do episódio 10000, o reforço atinge um valor aproximadamente constante sinalizando que o algoritmo convergiu para uma política de ações ótimo. Note também que, quando comparado ao Estudo de Caso I, a taxa de convergência do estudo de caso II é menor, isso implica em um aumento exponencial do número de iterações necessárias para que o algoritmo comece a convergir.

Ao aumentar o número de estados e ações é possível obter um controlador mais preciso, pois possibilita aumentar a variedade de tipos de curvas e distinção de estados. Entretanto, isso faz com que a matriz Q aumente de tamanho, o que aumenta a complexidade computacional do algoritmo. Isso se evidencia pela quantidade de iterações observadas no Estudo de Caso II.

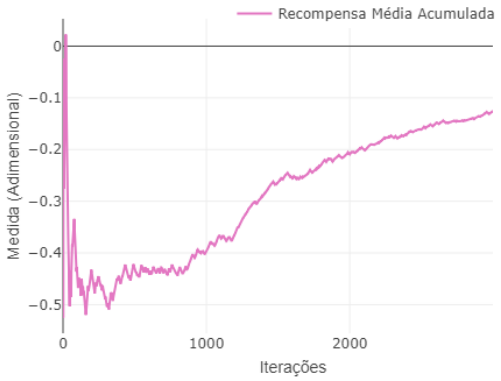


Figura 13. Recompensa média acumulada ao longo das iterações no Estudo de Caso 2.

O mapa de calor da Figura 14 mostra o número de acessos de cada um dos pares estado-ação. Neste mapa é possível observar que os pares de maior valor associado foram os mais acessados. A partir desse resultado, pode-se obter a Figura 8, que apresenta o percentual de acesso aos pares de maior valor para cada estado. Outro fato que se repetiu diz respeito ao Estado 5, em que a faixa estava fora do campo de visão, assim o par com maior valor associado não foi o mais acessado.

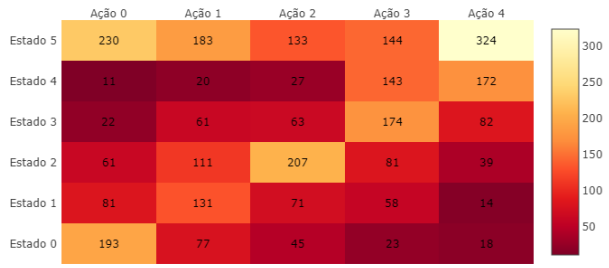


Figura 14. Mapa de Calor da quantidade de acesso aos pares estado-ação relativo ao Estudo de Caso 2.

Tabela 8. Percentual de acessos dos pares estado-ação de maior valor no Estudo de Caso.

| Estado | Total de Acessos | Ação com Maior Valor | Acessos da Ação | Percentual |
|---|---|---|---|---|
| 0 | 356 | Ação 0 | 193 | 54,2 % |
| 1 | 355 | Ação 1 | 131 | 36,9 % |
| 2 | 499 | Ação 2 | 207 | 41,5 % |
| 3 | 402 | Ação 3 | 174 | 43,3 % |
| 4 | 373 | Ação 4 | 172 | 46,1 % |
| 5 | 1014 | Ação 0 | 230 | 22,7 % |

## 5. CONCLUSÃO

A partir dos estudos de casos apresentados, foi possível observar que a abordagem utilizada conseguiu derivar controladores capazes de fazer o robô seguir diferentes trajetórias de forma eficiente. Estes estudos também mostraram que existe uma relação entre a quantidade de estados, ações e o número de iterações necessárias para que o sistema convirja para uma política de ações ótima. Quanto maior o número de estados e ações, maior é a quantidade de iterações necessárias. Este resultado faz sentido, pois o tamanho da matriz $Q$ aumenta, o que demanda uma maior quantidade de iterações para que todas as possibilidades sejam exploradas durante o treinamento. Outro resultado interessante é que a política aprendida no Estudo de Caso II mostrou que o robô aprendeu a executar curvas mais bruscas quando estava sem visão da trajetória. Na prática, esta ação garante que o robô aprenda a encontrar novamente o trajeto em poucas iterações.

## REFERÊNCIAS


Bekey, G.A. (2005). *Automonous Robots: From Biological Inspiration to implementation and Control*. Massachusetts: MIT Press Cambridge.

Campbell, J.S., Givigi, S.N., and Schwartz, H.M. (2016). Multiple Model Q-Learning for Stochastic Asynchronous Rewards. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 81(3-4), 407–422.

Carlucho, I., Paula, M.D., Villar, S.A., and Acosta, G.G. (2017). Incremental Q-learning strategy for adaptive PID control of mobile robots. *Expert Systems with Applications*, 80, 183–199.

de Sousa Neves, M.G. (2009). Auto-tuning de controladores pid pelo método relay: Optimização de controlo em automação industrial. *Dissertação de Mestrado, Universidade Técnica de Lisboa*.

Hakim, A., Hindersah, H., and Rijanto, E. (2013). Application of reinforcement learning on self-tuning pid controller for soccer robot multi-agent system. *IEEE Joint Intern. Conf. on Rural Information Communication Technology and Electric-Vehicle Technology*, 1–6.

Howell, M.N.; Best, M. (2000). On-line pid tuning for engine idle-speed control using continuous action reinforcement learning automata. *Control Engineering Practice*, 8(2), 147–154.

Kober, J., Bagnell, J.A., and Peters, J. (2014). Reinforcement learning in robotics: A survey. *Springer Tracts in Advanced Robotics*, 97, 9–67.

Meenakshi, M. (2008). Microprocessor based digital pid controller for speed control of d.c. motor. *International Conference on Emerging Trends in Engineering and Technology*, 1–6.

Pellegrini, J.; Wainer, J. (2007). Processos de decisão de markov: Um tutorial. *Revista de Informática Teórica e Aplicada*, 14(2), 133–179.

Siegwart, R.; Nourbakhsh, I.R. (2004). *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA.

Sutton, R.S.; Barto, A.G. (1999). Reinforcement Learning: An Introduction. *Trends in Cognitive Sciences*, 3(9), 360.

Watkins, C. H.; Dayan, P. (1992). Q-learning. In *Machine Learning*, 279–292.

Wilson, J., Charest, M., and Dubay, R. (2016). Non-linear model predictive control schemes with application on a 2 link vertical robot manipulator. *Robotics and Computer-Integrated Manufacturing*, 41, 23–30.