Driving Mobile Robots using a Deep LSTM Architecture: An Experimental Approach

Thúlio Noslen S. Santos, Marcus Vinicius Lamar, Carla Cavalcante Koike and Flávio de Barros Vidal

***Abstract*—Driving a mobile robot consists in moving the robot to its goal position quickly, without colliding with obstacles, and with no deviations from the planned path. This process requires a skilled pilot, with expertise and well trained. Among the various techniques available to help pilots to drive mobile robots, many of them are based on automatic learning approaches, as deep learning. This work presents the proposal of a new approaching for applications using a deep Long-Short Term Memory (LSTM) architecture, to assist the driving activities in mobile robots, using data extracted from an expert pilot as the main source of learning data. The main contributions of this proposal are i) a new application of the architecture for deep LSTM; ii) a new information data fusion strategy in the guidance command stage; and iii) a large number of tests in scenarios using a real mobile robot.**

***Index Terms*—Mobile robots, LSTM, Deep Learning.**

I. Introdução

O processo de pilotagem de robôs móveis consiste em levar o robô até o seu objetivo final sem colidir com obstáculos, da forma mais rápida possível e sem desvios, dividido em duas categorias: teleoperados e autônomos [1]. A pilotagem por teleoperação geralmente é custosa pois necessita de um operador capacitado para sua correta operação. Veículos robóticos teleoperados geralmente são equipamentos caros e pesados que podem ferir gravemente um ser humano, necessitando muito cuidado em seu manuseio. Para reduzir o tempo investido na capacitação de operadores, fabricantes de robôs contemplam a utilização de teleoperação com elementos de autonomia. Neste caso, as decisões têm tanto influência interna dos algoritmos do robô quanto do operador, de forma a aproveitar o melhor dos dois mundos.

Técnicas tradicionalmente aplicadas ao aprendizado de máquinas permitem aos robôs móveis terem a autonomia necessária no processo de pilotagem [2]. Dentre estas técnicas, o aprendizado profundo (do inglês *Deep Learning* - DL) é cada vez mais usado. Entre as técnicas de DL, tem se destacado aquelas que permitem a extração automática de características para sua utilização. Lecun *et al.* [3] e Goodfellow *et al.* [4] definem como DL o conjunto de métodos com múltiplos níveis de aprendizagem, obtidos pela composição a partir de módulos simples, não lineares, que alteram progressivamente a representação dos dados de entrada em sua forma bruta, para um nível abstrato, permitindo sua classificação. Dentre estas técnicas, as Redes Neurais Recorrentes são um tipo específico de rede neural artificial em que existe uma memória nas conexões entre os neurônios, permitindo que a rede tenha comportamento dinâmico no tempo. Dentro desta categoria, o modelo de redes LSTM, (do inglês *Long Short-Term Memory*), faz a utilização de um estado interno em sua arquitetura para resolver tarefas como o reconhecimento de escrita e fala, que necessitam de um contexto sequencial para que os termos sejam corretamente reconhecidos [5].

Neste artigo foi feita uma investigação sobre a capacidade de redes LSTM auxiliarem um teleoperador inexperiente na pilotagem de um robô móvel utilizando aprendizado supervisionado a partir de um teleoperador experiente durante a navegação. Deve-se ressaltar que a abordagem proposta neste manuscrito, não se trata de suavização de trajetórias, mas permitir a integração de aprendizagem de máquina no auxílio da guiagem do robô durante sua teleoperação. Para tal, na Seção II os principais trabalhos sobre o tema são apresentados, seguido pela Seção III com a metodologia proposta. Os resultados são apresentados e discutidos na Seção IV. As conclusões e trabalhos futuros na Seção V.

II. Trabalhos Relacionados

Atualmente o uso de técnicas de aprendizado profundo (do inglês *Deep Learning* - DL) encontra-se em aplicações do dia a dia [6] com as mais diversas atividades e aplicações([7], [8], [9], [10], [11], [12]). Nesta ampla gama de aplicações, é natural a utilização de técnicas de DL para a realização de tarefas destinadas à robótica móvel. Recentemente, trabalhos aplicados ao domínio da robótica móvel estão em desenvolvimento utilizando DL, tais como monitoramento de loja de varejo [13], simuladores de código aberto [14] e até mesmo em jogos de futebol [15].

O uso das técnicas de DL estão concentradas nas atividades de percepção (reconhecimento) do ambiente [16], [17] e no auxílio de tarefas [18], [19]. Em tarefas que envolvem a guiagem de robôs móveis, as principais pesquisas descrevem sua utilização a partir de técnicas em que o uso de Redes Neurais Convolucionais usam as informações advindas de sinais multidimensionais [20], [21]. Outros modelos também são aplicados neste tipo de tarefa, como as LSTMs [22] em robótica móvel. Algumas pesquisas de [23] e [24] utilizam a capacidade de memória deste tipo de rede para o planejamento de trajetórias, enquanto que o uso de LSTMs [25] é realizado para o planejamento de tarefas. Para atividades mais complexas, o uso de LSTMs podem ser aplicados em modelos de cognição matemática em robôs [26], incluindo aplicações de geração de movimento humano-robô colaborativo [27] e chegando ao aprendizado de predição de trajetória de pedestres [28].

Thúlio Noslen S. Santos, Marcus Vinicius Lamar, Carla Maria C. C. Koike and Flávio de Barros Vidal* are with Department of Computer Science in the University of Brasilia, Brasilia-DF, Brazil. *Correspondence author - e-mail: fbvidal@unb.br.

III. Metodologia Proposta

A primeira etapa consiste no planejamento da coleta de dados, devido ao grande volume de dados necessário ao treinamento da rede neural profunda escolhida (LSTM). Serão analisadas quais variáveis são relevantes para o processo de treinamento da rede, bem como definir o percurso do robô. A próxima etapa consiste na escolha das arquiteturas de rede neural LSTM adequadas ao problema. Após a escolha das arquiteturas e com a base de dados definida, as redes neurais são treinadas e a raiz do erro quadrático médio normalizado (do inglês *Normalized Root Mean Square Error* - NRMSE) sendo utilizado como critério para avaliar o aprendizado.

Nas próximas seções serão apresentados maiores detalhes dos equipamentos, ferramentas e métodos utilizados na elaboração da base de dados e das redes LSTM.

A. Materiais Utilizados

Neste trabalho, foi utilizado o robô Pioneer 3-AT (Figura 1). Este robô possui um computador embarcado e os dados de odometria são obtidos a partir dos *encoders* das rodas [29]. A programação do robô foi realizada com o auxílio do ROS (do inglês *Robotic Operating System*) [30]. As APIs e *drivers* que permitem a comunicação entre os algoritmos e os sensores e atuadores do robô estão implementados no ROS instalado no computador embarcado. A linguagem de programação utilizada foi *Python* 3 e como *framework* de aprendizado de máquina foi utilizado *Keras*, com *TensorFlow* de *backend*. Para o treinamento das redes neurais, foram usados um computador de mesa com GPU dedicada e o *cluster* de servidores *DevCloud* da Intel. Para os testes, foi usado um notebook com GPU dedicada conectado ao Pioneer. Para a guiagem do robô, foi usado um controle analógico de videogame conectado por cabo ao notebook. O ambiente escolhido, para a criação da base de dados e para os testes reais, é apresentado na Figura 2.

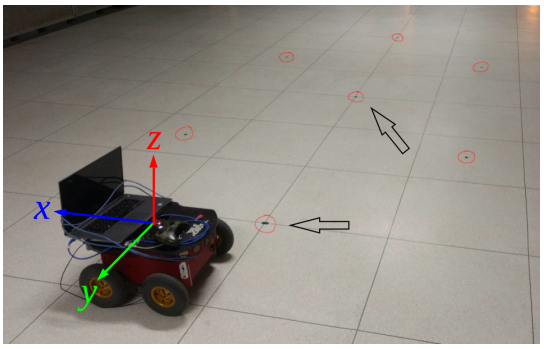

Fig. 1. Marcações do percurso em oito (setas) e definição dos eixos do sistema de coordenadas relativo ao robô. A frente do robô aponta para o eixo x positivo.

B. Criação da Base de Dados

A rede neural necessita aprender a pilotar o robô a partir de uma base de dados construída com os dados obtidos da pilotagem de um piloto experiente. Para criação dessa base de dados, foi definido um cenário de testes focado em uma navegação suave, sem manobras bruscas: independente da velocidade escolhida, a aceleração nas retas é zero e a aceleração nas curvas é muito baixa. O percurso de teste, mostrado em azul na Figura 2 consiste em fazer voltas em forma de 8, em ambos os sentidos de movimento. A Figura 1 mostra, com círculos vermelhos, as marcações feitas no chão para assegurar que o percurso seja executado na mesma escala por todos os pilotos.

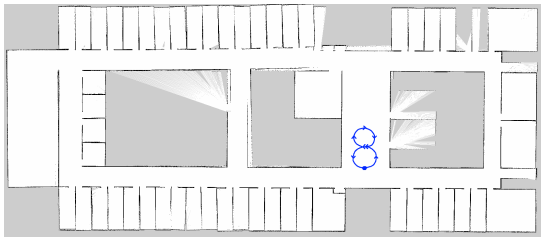

Fig. 2. Ambiente de realização dos testes. Tem-se o percurso de teste em azul, o ponto de início coincide com o de chegada, as setas indicam o sentido de movimento durante os testes. Fonte: https://github.com/Gastd/fcr2018

A entrada da rede neural é composta pelo comando vindo do usuário e de um vetor de contexto do robô. A saída é a predição do próximo comando, como se um piloto experiente estivesse guiando o robô. A entrada de comando do usuário é dada pela posição da alavanca analógica do controle remoto nos eixos x e y, representada pelo vetor $jp = (jp_x, jp_y)$. A saída de predição do próximo comando é representada por um vetor similar, dado por $jn = (jn_x, jn_y)$. Assim, na construção do conjunto de dados para o treinamento, jn em um instante de tempo k é dado por $jn[k] = jp[k+1]$. O vetor de contexto do robô inclui as velocidades linear v_x e angular ω_z do robô, de acordo com os eixos apresentados na Figura 1. Além dos valores de comandos e velocidades, as variações no tempo dessas variáveis também são utilizadas como entradas, de forma a capturar a dinâmica de primeira ordem do sistema de pilotagem, de acordo com $\Delta u[k] = u[k] - u[k-1]$, em que u é uma das variáveis e k o índice relacionado aos intervalos de tempo. Os intervalos de tempo decorridos entre medidas não são constantes, pois dependem do tempo entre publicações da odometria feitas pelo nó do ROS. Apesar da frequência esperada ser $10Hz$, o tempo entre publicações pode variar, então esse valor em nanosegundos, aqui chamado de Δt_{ns}, é utilizado como entrada na rede para auxiliar seu aprendizado. Para facilitar a notação, o vetor s é utilizado para representar as informações de contexto (ou estado) do robô e é definido como $s = (\Delta t_{ns}, v_x, \Delta v_x, \omega_z, \Delta \omega_z)$.

Para realizar a coleta de dados, o robô foi pilotado via controle por um usuário experiente utilizando a navegação considerada ideal. Os deslocamentos da alavanca analógica do controle remoto são mapeados em comandos de velocidade nos eixos do robô como: $v_x = 0.5 j_y$ e $\omega_z = 0.5 j_x$, em que v_x é a velocidade ao longo do eixo x, ω_z é a velocidade angular em torno do eixo z e j_x e j_y são os deslocamentos nos eixos x e y da alavanca do controle. Durante a coleta de dados, o robô Pioneer foi guiado no percurso descrito anteriormente por um piloto experiente por 10 minutos (10 vezes o percurso, sendo 5 voltas em um sentido da trajetória e mais 5 voltas no sentido

contrário, totalizando 3945 amostras coletadas em uma taxa de amostragem média de 116ms), e os dados da pilotagem (vetores s, jn,Δjn, jp e Δjp) foram obtidos e armazenados.

C. Arquitetura da Rede Neural

Para que correções sejam feitas na pilotagem de um usuário inexperiente, é necessária uma rede neural que consiga estimar qual deve ser o próximo comando de velocidade com base nos comandos anteriores do usuário e na pilotagem ideal. Este tipo de predição requer contexto, uma vez que é preciso estimar o que o usuário está tentando fazer, seja fazer uma curva ou seja seguir uma reta, para que se possa determinar a sugestão ideal. Nesse tipo de problema, em que as entradas anteriores da rede são importantes para dar contexto à entrada atual, as redes neurais recorrentes do tipo LSTM atendem aos requisitos. Na definição da arquitetura do modelo de uma LSTM, é necessário que seus hiperparâmetros de treinamento sejam definidos. Neste trabalho, são utilizados 8 diferentes combinações de parâmetros com arquiteturas compostas de duas até quatro camadas LSTM com número variável de neurônios. A quantidade de camadas e o número de neurônios foram parametrizados e os treinamentos foram instanciados, caracterizados pelo tamanho do *batch*. Na Tabela I são apresentadas as combinações de parâmetros de cada conjunto de parâmetros propostos. A fim de simplificar a referência a cada uma das combinações, cada arquitetura da tabela será citada seguindo a notação $L-B$, em que L é uma tupla representando o número de neurônios por camada e B é o tamanho do *batch*.

TABELA I
ARQUITETURAS PROPOSTAS E HIPERPARÂMETROS DE TREINAMENTO.

Épocas	Paciência	Camadas	Neurônios	Tam. do *batch*
200	50	2	(32, 16)	32
200	70	2	(32, 16)	64
200	50	2	(64, 32)	32
200	70	2	(64, 32)	64
200	50	3	(96, 64, 32)	32
200	70	3	(96, 64, 32)	64
200	50	4	(128, 96, 64, 32)	32
200	70	4	(128, 96, 64, 32)	64

O diagrama de entradas e saídas da rede LSTM é apresentado na Figura 3. A rede possui como entradas o vetor de contexto $s[k]$, o comando do usuário $jp[k]$ e sua variação $\Delta jp[k]$ no instante k, e fornece como saída a sugestão de comando $jn[k+1]$ para o instante $k+1$.

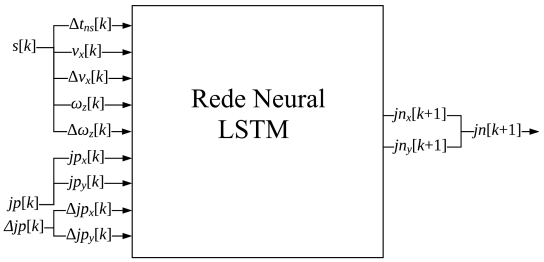

Fig. 3. Diagrama de entradas e saídas da rede neural LSTM

.

1) Treinamento da rede neural: No processo de treinamento, utilizou-se o Erro Médio Quadrático (MSE, do inglês *Mean Squared Error*) como função de perda, com os dados de entrada e saída normalizados no intervalo $[-1, 1]$. O otimizador *Adam* [31] foi utilizado com os parâmetros $\eta = 0.001$, $\beta_1 = 0.9$ e $\beta_2 = 0.999$, em que η é a taxa de aprendizado e os parâmetros β_1 e β_2 são as taxas de decaimento exponencial. Com o objetivo de obter maior generalização, foram utilizados dois tipos de *dropouts* em cada camada LSTM, com exceção da última: o *dropout* comum, que atua entre as entradas e saídas das camadas da arquitetura LSTM, e o *dropout* recorrente, que atua nas portas internas da arquitetura LSTM. Ambos os tipos foram configurados para 20%. Foi utilizado também a regularização a partir do critério de *early stopping* [4] para reduzir os tempos de treinamento utilizando como critério a **Paciência**. Todos os modelos são treinados com no máximo 200 épocas e o parâmetro de **Paciência**, p, ajustado empiricamente em $p = 50$ épocas para um *batch* de 32 e $p = 70$ épocas para um *batch* de 64. O modelo salvo é aquele com a menor perda no conjunto de validação.

Para avaliar o processo de treinamento e teste, foi utilizada a validação cruzada com 10-*folds* para cada arquitetura. Para cada arquitetura foram calculadas a média e o desvio padrão do NRMSE. Após validar as arquiteturas, os modelos escolhidos foram treinados. Para isso, o conjunto de dados foi separado em conjuntos de treino, validação e teste, com divisão de 70%, 15% e 15%, respectivamente. Com isto, 5 iterações de cada uma das arquiteturas propostas foram treinadas e o melhor modelo dentre os 40 modelos treinados no percurso foi escolhido para ser usado no algoritmo de correção para validação qualitativa.

D. Validação Qualitativa dos Modelos

A validação qualitativa dos modelos foi realizada por meio de sessões de operação de pilotagem no robô, no qual pessoas sem experiência em pilotar o robô executam o percurso em oito (Figura 1). Durante a pilotagem, cada modelo utilizado nos testes corrige seus comandos em tempo real, de forma a auxiliar a navegação do piloto inexperiente a partir do aprendizado da navegação ideal do piloto experiente. Esta correção é realizada somente quando a diferença entre o comando do usuário e a saída do algoritmo do modelo de rede neural utilizado ultrapassa um limiar de seleção T. A diferença dos comandos é calculada por:

$$\mathit{diff}(ja, jb) = \frac{||ja - jb||}{2\sqrt{2}} = \frac{\sqrt{(ja_x - jb_x)^2 + (ja_y - jb_y)^2}}{2\sqrt{2}}, \tag{1}$$

em que ja e jb são as entradas de comando do robô, geradas pelo piloto e pelo modelo treinado. A correção do comando do usuário pode ser realizada conforme diagrama da Figura 4.

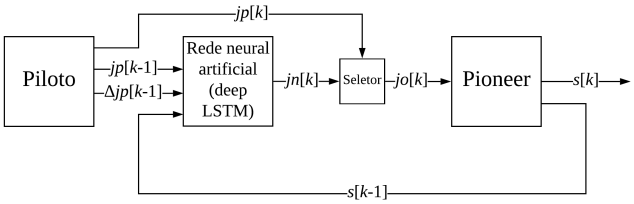

Fig. 4. Diagrama do sistema com correção por rede neural.

O bloco Seletor realiza a decisão sobre a correção de acordo com o Algoritmo 1.

Algorithm 1 Algoritmo do Seletor

função SELETOR($jp, jn, metodo, T$)
  se Método = Desligado **então**
    $jo \leftarrow jp$
  senão se Método = Rede **então**
    se *diff*$(jp, jn) \geq T$ **então**
      $jo \leftarrow jn$
    senão
      $jo \leftarrow jp$
    fim se
  senão se Método = Média **então**
    se *diff*$(jp, \frac{jp+jn}{2}) \geq T$ **então**
      $jo \leftarrow \frac{jp+jn}{2}$
    senão
      $jo \leftarrow jp$
    fim se
  fim se
  retorne jo
fim função

Os dois parâmetros no algoritmo de correção são o limiar de seleção T e o método de correção. Para os testes, foram selecionados dois valores de limiar, $T = 5\%$ ou 20%. Esses valores são empregados para permitir uma situação em que o sistema atua mais frequentemente (5%) e outra no qual o piloto tem mais autonomia (20%). Os valores específicos foram obtidos por meio da análise dos dados de pilotagem: iniciando em 5%, o valor foi incrementado até que a atuação do sistema fosse reduzida. O método de correção pode ser Desligado, Rede ou Média, de acordo com o Algoritmo 1. Esses parâmetros resultam em 5 combinações descritas na Tabela II. Os testes foram realizados no percurso em oito e foram comparadas 2 ou 3 combinações da Tabela II. Para tentar eliminar o viés na validação, as combinações escolhidas e a ordem dos testes foram aleatórias para cada pessoa. Além disso, não foram informadas quais as combinações estavam sendo testadas. A Tabela III apresenta os testes realizados.

TABELA II
COMBINAÇÕES DE PARÂMETROS PARA TESTES DE VALIDAÇÃO.

Método de correção	Limiar de seleção (%)	Notação
Desligado	-	D
Rede	5	R5
Rede	20	R20
Média	5	M5
Média	20	M20

Ao total, foram realizados 14 testes, cada um com uma pessoa que nunca havia pilotado um robô móvel. Dois dos testes envolveram 3 métodos de correção diferentes, o restante envolveu somente 2 métodos. Para a análise quantitativa dos métodos de correção, é importante agrupar quantas pessoas testaram cada método para que possam ser extraídas métricas específicas de cada método, mostrados na Tabela IV.

TABELA III
TESTES REALIZADOS PARA VALIDAÇÃO.

Nº do teste	Método 1	Método 2	Método 3
1	M5	R5	-
2	R5	M5	-
3	R20	M20	-
4	R5	M5	-
5	R5	M5	-
6	M20	R20	-
7	R20	R5	-
8	M20	R20	-
9	M5	M20	-
10	M5	R5	-
11	R5	R20	-
12	R20	M20	-
13	M20	M5	D
14	D	M5	R5

TABELA IV
NÚMERO DE PESSOAS QUE TESTARAM CADA MÉTODO DE CORREÇÃO.

Método	Número de pessoas
D	2
R5	8
M5	8
R20	6
M20	6

Durante os testes, a cada ciclo de correção, uma amostra das informações do sistema foi coletada. A partir dessas informações foram extraídas as métricas usadas para a avaliação quantitativa do desempenho da rede. Nas equações a seguir, N é o número total de amostras coletadas durante o teste, isto é, o número de ciclos executados pelo algoritmo até o momento do cálculo da métrica.

- Tempo médio de predição do algoritmo ($\bar{t}_{pred}$): permite avaliar o atraso devido à rede neural, é dado por

$$\bar{t}_{pred} = \frac{1}{N}\sum_{k=0}^{N} t_{pred}[k], \tag{2}$$

  onde $t_{pred}[k]$ é o tempo necessário para a rede neural realizar a predição no k-ésimo ciclo de correção.
- Percentual de atuação do algoritmo (pa): permite determinar o quanto o algoritmo corrigiu o usuário, dado por

$$pa = \frac{\sum_{k=0}^{N} n[k]}{\sum_{k=0}^{N}(n[k] \vee m[k])}, \tag{3}$$

  em que $n[k]$ e $m[k]$ são variáveis lógicas, e $\vee$ é o operador OU lógico. Os valores de $n[k]$ e $m[k]$ definidos como

$$n[k] = \begin{cases} 1, & \text{se o algoritmo estava atuando no ciclo } k; \\ 0, & \text{caso contrário; } e \end{cases}$$

$$m[k] = \begin{cases} 1, & \text{se o robô estava se movendo no ciclo } k; \\ 0, & \text{caso contrário.} \end{cases}$$

- Aceleração linear média ($\overline{a}_x$): permite determinar a suavidade das mudanças de velocidade linear, dada por

$$\overline{a}_x = \frac{1}{N}\sum_{k=0}^{N}\left|\frac{\Delta v_x[k]}{\Delta t[k]}\right|. \quad (4)$$

- Aceleração angular média ($\overline{\alpha}_z$): permite determinar a suavidade das mudanças de velocidade angular, dada por

$$\overline{\alpha}_z = \frac{1}{N}\sum_{k=0}^{N}\left|\frac{\Delta \omega_z[k]}{\Delta t[k]}\right|. \quad (5)$$

De forma a permitir que $\overline{a}_x$ e $\overline{\alpha}_z$ sejam indicativos de suavidade, essas acelerações médias foram calculadas usando o valor absoluto da diferença das velocidades, $|\Delta v_x|$ e $|\Delta \omega_z|$.

IV. Resultados

A. Criação da Base de Dados

A base de dados com o perfil de um piloto experiente no percurso em oito foi criada com os dados capturados de acordo com a proposta de baixa variação de velocidade.

A Figura 5 mostra exemplos de dados coletados no percurso.

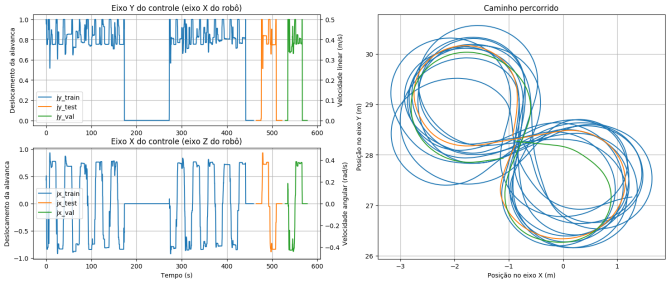

Fig. 5. Dados coletados de um usuário experiente durante o percurso.

Na Figura 5, os gráficos à esquerda mostram a velocidade linear (superior), em m/s, e a velocidade angular (inferior), em rad/s, em função do tempo em segundos. O gráfico à direita mostra a trajetória percorrida pelo robô. São mostrados ainda os conjuntos de treino (em azul), teste (em laranja) e validação (em verde). É possível observar que a trajetória é uniforme e as curvas foram feitas com bastante suavidade, apesar da variação de velocidade linear ser maior que o desejado. Os trechos com velocidade zero foram pausas na captura dos dados. As oscilações na velocidade angular correspondem às mudanças de direção que acontecem quando se cruza o meio do oito.

B. Validação Quantitativa dos Modelos

Os resultados da validação cruzada do aprendizado em 200 épocas são apresentados na Tabela V na coluna *NRMSE(%) VC*, enquanto os resultados do treinamento estão na coluna *NRMSE(%) TR*, com as divisões de treino, teste e validação definidas anteriormente.

Verifica-se pela Tabela V que o melhor modelo obtido foi com a arquitetura $(96, 64, 32) - 32$, com erro NRMSE $= 4.36\%$. Observa-se ainda que todas as arquiteturas propostas tem potencial de aprendizado com os dados de treinamento, uma vez que a média do NRMSE ficou abaixo de 5% para todos os modelos. A discrepância entre os NRMSEs da validação cruzada e os do treinamento se deve ao método pelo qual o *dataset* foi separado em treino, validação e teste. Na validação cruzada, o *dataset* foi dividido aleatoriamente em 10 *folds*, enquanto que no treinamento a divisão foi feita conforme explicado na Seção III-C1.

TABELA V
Resultados da validação cruzada (VC) e do treinamento (TR).

Paciência	Arquitetura	NRMSE(%) VC	NRMSE(%) TR
50	(32, 16) – 32	4.71 ± 0.56	5.10 ± 0.23
70	(32, 16) – 64	4.59 ± 0.68	5.36 ± 0.22
50	(64, 32) – 32	4.76 ± 0.63	5.13 ± 0.12
70	(64, 32) – 64	4.77 ± 0.79	5.18 ± 0.04
50	(96, 64, 32) – 32	4.36 ± 0.83	4.90 ± 0.26
70	(96, 64, 32) – 64	4.46 ± 0.75	4.84 ± 0.22
50	(128, 96, 64, 32) – 32	4.41 ± 0.68	5.08 ± 0.18
70	(128, 96, 64, 32) – 64	4.36 ± 0.73	5.12 ± 0.14

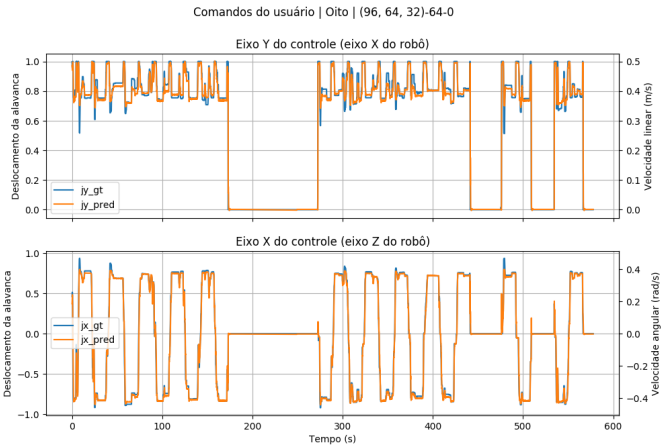

Fig. 6. Predição dos comandos para a arquitetura $(96, 64, 32) - 32$.

Os gráficos das predições reais estão apresentados na Figura 6 e indicam a convergência no aprendizado.

C. Validação Qualitativa dos Modelos

A Tabela VI apresenta o tempo de predição médio da rede neural calculada ao final dos testes. Podemos observar que o tempo de predição da rede ficou próximo de $10ms$, indicando que o modelo é adequado para correção em tempo real dos comandos, desde que a velocidade de deslocamento do robô seja limitada a 1 m/s. A Tabela VII contém as métricas do algoritmo completo para o percurso de teste.

TABELA VI
Tempo de predição médio do modelo.

Método	$\bar{t}_{pred}$ (ms)
D	-
R5	9.2 ± 0.2
M5	9.2 ± 0.2
R20	9.1 ± 0.2
M20	9.2 ± 0.2

Pode-se observar pelo percentual de atuação (pa) que o método de correção por rede neural atua mais frequentemente do que a correção por média. Isto acontece porque $||jp - \frac{jp+jn}{2}|| = ||\frac{jp-jn}{2}|| = \frac{1}{2}||jp - jn||$, então

TABELA VII
RESULTADOS DO ALGORITMO NA VALIDAÇÃO QUALITATIVA.

Método	pa (%)	$\overline{a}_x$ (m/s^2)	$\overline{\alpha}_z$ (rad/s^2)
D	-	0.1381 ± 0.0524	0.2428 ± 0.0474
R5	67.52 ± 10.10	0.0928 ± 0.0253	0.2133 ± 0.0480
M5	14.34 ± 11.86	0.1222 ± 0.0418	0.2534 ± 0.0769
R20	12.83 ± 5.30	0.1234 ± 0.0243	0.2610 ± 0.0679
M20	0.09 ± 0.17	0.1131 ± 0.0271	0.2142 ± 0.0618
R	47.99 ± 27.61	0.1037 ± 0.0290	0.2303 ± 0.0604
M	9.25 ± 11.71	0.1189 ± 0.0375	0.2394 ± 0.0743

$diff(jp, \frac{jp+jn}{2}) = \frac{1}{2}diff(jp, jn)$, fazendo com que a correção por média sempre atue menos frequentemente do que a correção por rede para um mesmo limiar.

Observando as colunas das acelerações linear média $\overline{a}_x$ e angular média $\overline{\alpha}_z$, verifica-se que a correção por rede com limiar de 5% tende a suavizar melhor a direção do piloto, em comparação com a correção por média. Com $pa = 67.52\%$, a correção por rede está na maior parte do tempo suavizando a trajetória e impedindo movimentos bruscos, e o limiar de 5% permite que as transições sejam suaves. Para o limiar de 20%, a suavização foi maior para a correção por média, devido ao chaveamento entre o comando do piloto e do algoritmo ser uma transição abrupta para esse limiar. Com $pa = 0.09\%$, a correção por média interfere pouco na direção do usuário. A correção por rede, com $pa = 12.83\%$, torna o movimento do usuário menos suave, uma vez que para o limiar de 20%, as transições entre períodos de correção e não-correção são menos suaves do que as do limiar de 5%.

A análise da aceleração média com método de correção Desligado se torna não confiável devido ao baixo número de pessoas que fizeram testes com esse método. Com somente 2 pessoas, as variações no perfil de pilotagem se tornam muito mais significativas do que para os outros métodos, que tiveram testes com 6 ou 8 pessoas.

D. Perfis de Pilotagem

Durante os testes, observou-se que o perfil de pilotagem dos usuários variou bastante. Por perfil de pilotagem entende-se a maneira como são feitas as curvas. Os gráficos das seções subsequentes consistem nos comandos de velocidades e a trajetória, no mesmo formato da Figura 5. Os gráficos estão codificados por cores. Os comandos do piloto estão com cores frias (tons de azul e verde) e os comandos corrigidos estão com cores quentes (amarelo, laranja e vermelho). O percurso também está com cores quentes, indicando que é a trajetória com os comandos corrigidos. O ponto roxo indica a posição de início e o ponto azul indica a posição final do teste.

1) Usuário A: No primeiro caso, tem-se a realização dos testes indicados na Tabela VIII, na ordem D → M5 → R5. O usuário A teve dificuldades em seguir a trajetória em oito sem correção e dos três métodos testados, preferiu dirigir com correção por rede neural. A pilotagem sem correção, mostrada na Figura 7, resulta em maiores acelerações $\overline{a}_x$ e $\overline{\alpha}_z$, ou seja, menor suavização. A correção por média M5, mostrada na Figura 8, resulta em suavização intermediária. A correção por rede neural R5, mostrada na Figura 9, a preferida do usuário A, resulta na maior suavização. Pode-se observar nos gráficos de trajetória que as curvas se tornam mais bem definidas conforme a interferência da rede aumenta. O algoritmo tenta manter os comandos do usuário constantes, principalmente os de velocidade linear. Isto indica que a rede neural conseguiu aprender e cumpriu bem o objetivo de auxiliar na pilotagem.

TABELA VIII
RESULTADOS DOS TESTES PARA O USUÁRIO A.

Método	pa (%)	$\overline{a}_x$ (m/s^2)	$\overline{\alpha}_z$ (rad/s^2)
D	-	0.1904	0.2902
M5	27.77	0.1582	0.2791
R5	78.47	0.0638	0.1707

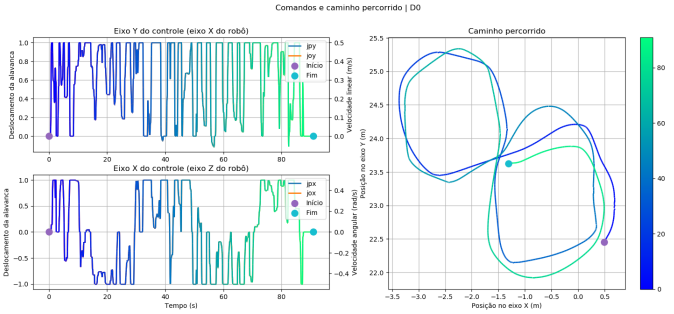

Fig. 7. Gráficos do usuário A para testes sem correção.

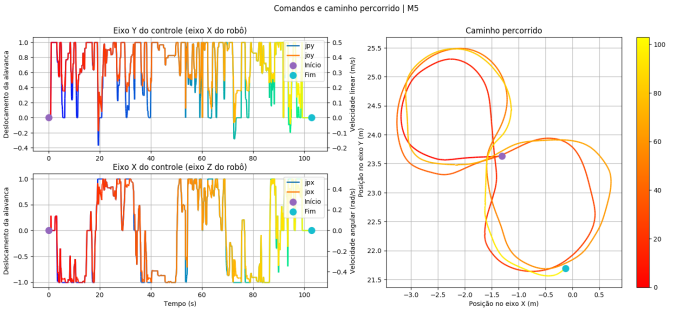

Fig. 8. Gráficos do usuário A para testes com M5.

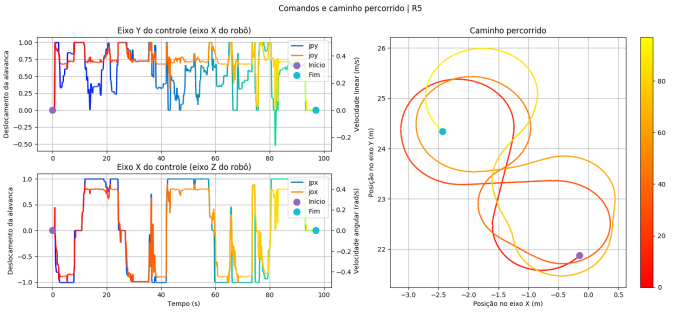

Fig. 9. Gráficos do usuário A para testes com R5.

2) Usuário B: O algoritmo foi colocado em uso com usuário B nos testes indicados na Tabela IX, na ordem M5 → R5. O usuário B realizou o percurso com maior facilidade na correção por média, conforme visto na Figura 10, sendo indicado como seu modo de correção preferido. Apesar da velocidade linear estar mais instável, a trajetória do usuário está bastante uniforme, com as voltas bem alinhadas. A correção por rede neural resulta em maior suavização que por média. Entretanto, o usuário saiu algumas vezes do trajeto, que pode ser notado

na Figura 11. O algoritmo tenta impor suavidade pela rede, porém encontra alguma resistência da parte do usuário. A velocidade linear neste caso está mais estável, mas como o perfil de pilotagem do usuário é diferente do perfil da rede, a trajetória está menos uniforme.

Observando a Tabela IX do usuário B, tem-se que a mesma correção por rede acarretou em menos suavização do que no caso do usuário A. Isto possivelmente se deve à dissimilaridade do perfil do usuário B para com o perfil da rede. O usuário teve que constantemente ajustar sua pilotagem compensando as sugestões do algoritmo, e isso acabou aumentando a aceleração média. O método preferido foi o M5, que confere a esse usuário maior liberdade na pilotagem do robô.

TABELA IX
RESULTADOS DOS TESTES PARA O USUÁRIO B.

Método	pa (%)	am_x (m/s^2)	αm_z (m/s^2)
M5	21.66	0.1477	0.2858
R5	62.71	0.0958	0.2311

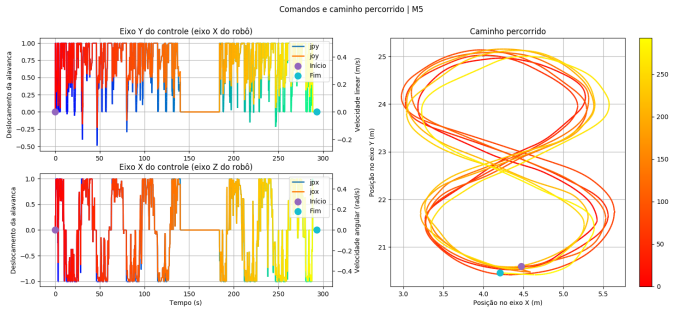

Fig. 10. Gráficos do usuário B para testes com M5. Durante o teste, o sentido das voltas foi invertido, evidenciado pela pausa nos comandos.

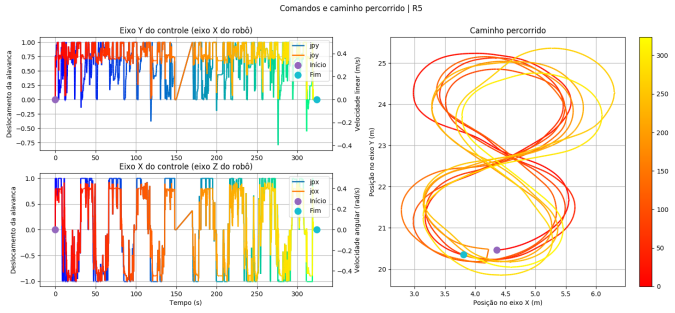

Fig. 11. Gráficos do usuário B para testes com R5.

3) Usuário C: Este usuário possui um perfil de pilotagem bastante diferente do esperado. Este usuário tentou fazer o oito com linhas retas e curvas em torno do eixo do robô, conforme pode ser visto na trajetória da Figura 12. A imposição de suavidade também não agradou o usuário C, que preferiu a correção por média (M20). Os testes foram realizados como indicados na Tabela X, na ordem M20 → R20. Já no caso do método R20, mostrado na Figura 13, o algoritmo tenta conformar o usuário ao perfil da rede, evidenciado pelo formato mais arredondado da trajetória. Isto, no entanto, fez com que o usuário não gostasse deste método. Para o usuário C, as acelerações $\overline{a}_z$ e $\overline{\alpha}_z$ possuem grandezas ligeiramente maiores do que para os usuários A e B. Além disso, $\overline{\alpha}_z$ para os dois métodos tem valores próximos. Durante os testes, percebeu-se que o usuário teve muita dificuldade em pilotar como gostaria. Por exemplo, o usuário tentou repetidas vezes girar o robô em torno do próprio eixo, mas o algoritmo impedia, e só permitia que as curvas fossem feitas se existisse velocidade linear.

Isto evidencia que a coleta de dados não incluiu situações onde o robô é comandado para girar em torno do próprio eixo, com $v_x \approx 0$ e $|\omega_z| > 0$, ou comandado de ré, com velocidade linear negativa $v_x < 0$. Como a rede neural não aprendeu este tipo de movimento, o algoritmo de correção por rede não sugere este tipo de movimentação, e tenta corrigi-lo quando o usuário o utiliza.

TABELA X
RESULTADOS DOS TESTES PARA O USUÁRIO C.

Método	pa (%)	$\overline{a}_x$ (m/s^2)	$\overline{\alpha}_z$ (rad/s^2)
M20	0.44	0.1606	0.3335
R20	18.15	0.1436	0.3405

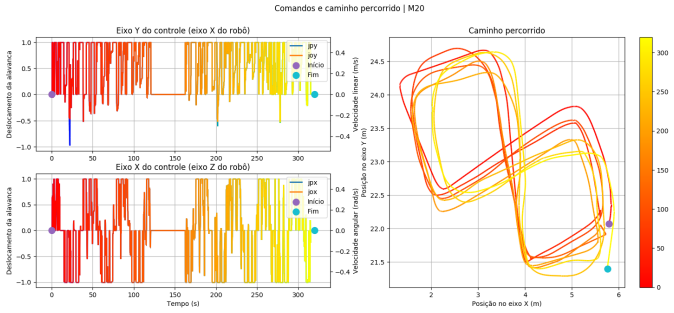

Fig. 12. Gráficos do usuário C para testes com M20.

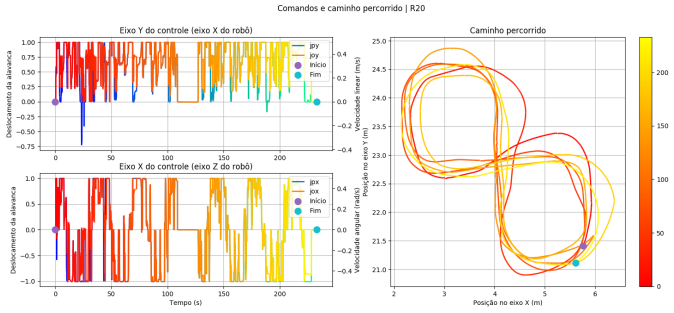

Fig. 13. Gráficos do usuário C para testes com R20.

A Tabela V mostra que o aprendizado aconteceu corretamente, as arquiteturas propostas aprenderam a sequência de dados e a arquitetura $(96, 64, 32)$ obteve o melhor resultado. Na Tabela VII, pôde-se verificar que a correção por rede neural tende a suavizar mais a direção do piloto, principalmente com limiar de 5%. Devido às diferenças de perfil e dificuldade do percurso em oito, temos maior variabilidade nos dados. Pode-se perceber que a rede neural estava atuando como desejado, suavizando os comandos de velocidade, ainda que alguns usuários não gostassem da correção.

Durante os testes, foi observado que o perfil de pilotagem variou bastante entre os usuários. Qualidades como a familiaridade do usuário com um controle de videogame, a experiência com jogos e a idade afetam o perfil de pilotagem. Algumas pessoas apresentaram pouca dificuldade para guiar o robô no percurso em oito, uma vez que a velocidade do robô era de $0.5m/s$, mais lento que a caminhada normal de um humano. Devido à pequena quantidade de pessoas que validaram o algoritmo, os resultados indicam apenas tendências, e não conclusões estatísticas de alta confiabilidade.

V. Conclusões

Este artigo teve por objetivo propor e validar modelos de redes neurais que pudessem predizer o próximo comando na pilotagem de um robô móvel de um usuário inexperiente baseado na pilotagem de um usuário experiente. Foram definidas as técnicas de pesquisa e os procedimentos experimentais. Foram propostas e testadas arquiteturas de redes neurais LSTM, e definida a estrutura do algoritmo de correção quanto aos métodos utilizados, que podem ser baseados em correção por rede neural ou correção por média. Uma base de dados foi construída com a pilotagem de um operador experiente e as arquiteturas propostas foram validadas quantitativamente. Foi verificado que a validação cruzada e o aprendizado convergiram nos modelos propostos, e validadas qualitativamente em testes com voluntários e entrevistas. Destes testes, foram verificadas algumas tendências, como a preferência dos usuários por liberdade na pilotagem. Por fim, pôde-se avaliar, de acordo com a definição de suavidade deste artigo, que os métodos de correção propostos resultam em suavização maior dos comandos de velocidade do que a pilotagem sem correção, e a correção por rede neural resultou em uma suavização maior do que a correção por média. Isso mostra que os métodos de correção propostos cumprem seu objetivo e suavizam a pilotagem de usuários inexperientes, indicando a viabilidade do uso de redes LSTM em módulos de auxílio de direção para veículos robóticos.

Para melhor desempenho da correção, podem ser feitas melhorias na qualidade e volume da base de dados, além da inclusão de imagens no conjunto de dados utilizados. Observou-se que o chaveamento entre o comando do piloto e do algoritmo não possui uma transição suave, principalmente no limiar de 20% de diferença. Uma melhoria possível seria usar um filtro de Kalman com modelo em espaço de estados para a fusão dos dados. Pode-se ainda usar uma fusão personalizada dependente do perfil de pilotagem do usuário, algo que pode ser identificado e chaveado em tempo real, uma vez que um perfil muito dissimilar ao da rede acaba acarretando em uma correção desagradável ao usuário.

Como trabalhos futuros pode-se citar estudos com maior número de trajetórias e aprendizado com diversos pilotos experientes, além de estudos comportamentais e de ergonomia referentes ao processo de auxílio na guiagem de robôs móveis.

Referências

[1] E. Kruse and F. M. Wahl, "Camera-based monitoring system for mobile robot guidance," in *Proc. 1998 IEEE Int. Conf. on Intelligent Robots and Systems*, vol. 2, 1998, pp. 1248–1253.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 1st ed. MIT Press, 2005.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed. MIT Press, 2016, http://www.deeplearningbook.org.

[5] S. Ghosh and P. O. Kristensson, "Neural networks for text correction and completion in keyboard decoding," *CoRR*, vol. abs/1709.06429, 2017. [Online]. Available: http://arxiv.org/abs/1709.06429

[6] J. Salas, F. B. Vidal, and F. Martinez-Trinidad, "Deep learning: Current state," *IEEE Latin America Trans.*, vol. 17, no. 12, pp. 1925–1945, 2019.

[7] A. Yala, C. Lehman, T. Schuster, T. Portnoi, and R. Barzilay, "A Deep Learning Mammography-based Model for Improved Breast Cancer Risk Prediction," *Radiology*, p. 182716, 2019.

[8] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep Learning based Recommender System: A Survey and New Perspectives," *ACM CSUR*, vol. 52, no. 1, p. 5, 2019.

[9] P. Univaso, J. Ale, and J. Gurlekian, "Data Mining applied to Forensic Speaker Identification," *IEEE Latin America Trans.*, vol. 13, no. 4, pp. 1098–1111, April 2015.

[10] J. Jimenez, I. Gonzalez, and J. Lopez, "Challenges And Opportunities In Analytic-Predictive Environments Of Big Data And Natural Language Processing For Social Network Rating Systems," *IEEE Latin America Trans.*, vol. 16, no. 2, pp. 592–597, Feb 2018.

[11] A. da Silva Abade., A. P. G. S. de Almeida., and F. B. Vidal, "Plant diseases recognition from digital images using multichannel convolutional neural networks," in *Proc. of the 14th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications*, INSTICC. SciTePress, 2019, pp. 450–458.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale Video Classification with Convolutional Neural Networks," in *CVPR*, 2014.

[13] M. Paolanti, L. Romeo, M. Martini, A. Mancini, E. Frontoni, and P. Zingaretti, "Robotic retail surveying by deep learning visual and textual data," *Robotics and Aut. Systems*, vol. 118, pp. 179 – 188, 2019.

[14] A. Plasencia, Y. Shichkina, I. Suárez, and Z. Ruiz, "Open source robotic simulators platforms for teaching deep reinforcement learning algorithms," *Proc. of the 13th Int. Symp. Intelligent Systems (INTELS'18), Russia*, vol. 150, pp. 162 – 170, 2019.

[15] J. M. Catacora Ocana, F. Riccio, R. Capobianco, and D. Nardi, "Cooperative multi-agent deep reinforcement learning in soccer domains," in *Proc. of the 18th Int. Conf. on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19, 2019, pp. 1865–1867.

[16] H. Sasaki, T. Horiuchi, and S. Kato, "A study on vision-based mobile robot learning by deep q-network," in *56th Annual Conf. of the Society of Instrument and Control Engineers of Japan (SICE)*, 2017, pp. 799–804.

[17] G. Shan, T. Wang, X. Li, Y. Fang, and Y. Zhang, "A deep learning-based visual perception approach for mobile robots," in *2018 Chinese Automation Congress (CAC)*, Nov 2018, pp. 825–829.

[18] T. Tongloy, S. Chuwongin, K. Jaksukam, C. Chousangsuntorn, and S. Boonsang, "Asynchronous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks," in *2017 2nd Int. Conf. on Robotics and Automation Engineering (ICRAE)*, 2017, pp. 68–72.

[19] D. Zhu, "Deep learning over iot big data-based ubiquitous parking guidance robot for parking near destination especially hospital," *Personal Ubiquitous Comput.*, vol. 22, no. 5-6, pp. 1109–1116, Oct. 2018.

[20] U. Côté Allard, F. Nougarou, C. L. Fall, P. Giguère, C. Gosselin, F. Laviolette, and B. Gosselin, "A convolutional neural network for robotic arm guidance using semg based frequency-features," in *IEEE Int. Conf. on Intel. Robots and Systems (IROS)*, 2016, pp. 2464–2470.

[21] F. Cruz, J. Twiefel, S. Magg, C. Weber, and S. Wermter, "Interactive reinforcement learning through speech guidance in a domestic scenario," in *2015 Int. Joint Conf. on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[22] A. G. Chavez, C. A. Mueller, A. Birk, A. Babic, and N. Miskovic, "Stereo-vision based diver pose estimation using LSTM recurrent neural networks for auv navigation guidance," in *OCEANS 2017 - Aberdeen*, June 2017, pp. 1–7.

[23] F. Nicola, Y. Fujimoto, and R. Oboe, "A LSTM neural network applied to mobile robots path planning," in *2018 IEEE 16th Int. Conf. on Industrial Informatics (INDIN)*, 2018, pp. 349–354.

[24] W. Khaksar, S. Vivekananthen, K. S. M. Saharia, M. Yousefi, and F. B. Ismail, "A review on mobile robots motion path planning in unknown environments," in *2015 IEEE Int. Symp. on Robotics and Intelligent Sensors*, 2015, pp. 295–300.

[25] S. Peng, H. Chen, C. Du, J. Li, and N. Jing, "Onboard observation task planning for an autonomous earth observation satellite using long short-term memory," *IEEE Access*, vol. 6, pp. 65 118–65 129, 2018.

[26] A. D. Nuovo, "Long-short term memory networks for modelling embodied mathematical cognition in robots," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–7.
[27] X. Zhao, S. Chumkamon, S. Duan, J. Rojas, and J. Pan, "Collaborative human-robot motion generation using lstm-rnn," in *2018 IEEE-RAS 18th Int. Conf. on Humanoid Robots (Humanoids)*, 2018, pp. 1–9.
[28] L. Sun, Z. Yan, S. M. Mellado, M. Hanheide, and T. Duckett, "3dof pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5942–5948.
[29] I. AdeptTechnology, "Pioneer 3-at - datasheet," Online, 2011. [Online]. Available: https://www.generationrobots.com/media/Pioneer3AT-P3AT-RevA-datasheet.pdf
[30] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

Thúlio Noslen S. Santos Possui bacharelado em Engenharia de Controle e Automação pela Universidade de Brasília (2019). Suas principais áreas de interesse são robótica, visão computacional e aprendizagem profunda.

Marcus Vinicius Lamar Possui graduação em Engenharia Elétrica pela Universidade Federal do Rio Grande do Sul (1991), mestrado em Engenharia Elétrica pela Universidade Federal de Santa Catarina (1996) e doutorado Engenharia Elétrica e da Computação pelo Instituto de Tecnologia de Nagoya, Japão (2001) em Visão Computacional e Redes Neurais. Atualmente é Professor Associado da Universidade de Brasília, atuando nas áreas de Inteligência Artificial e Visão Computacional.

Carla Cavalcante Koike Possui graduação em Engenharia Elétrica pela Universidade Federal do Ceará (1991), mestrado em Engenharia Elétrica pela Universidade Federal de Santa Catarina (1994) e doutorado em Imagem, Visão e Robótica - INP Grenoble , França (2005). Atualmente é Professora Associada da Universidade de Brasília, atuando nas áreas de Robótica móvel, robótica educacional e Aprendizagem Ativa.

Flávio de Barros Vidal recebeu o grau de Engenheiro Eletricista pela Universidade Federal de Juiz de Fora (UFJF) em 2002, Mestre e Doutor em Engenharia Elétrica pela Universidade de Brasília (UnB) em 2005 e 2009, respectivamente. Atualmente é Professor Associado do Departamento de Ciência da Computação(CIC) da UnB, atuando nas áreas de visão computacional, aprendizagem profunda, ciências forenses e biometria.