CLASSIFICADOR DE TRAJETOS PARA ROBÔS DE CORRIDA SEGUIDORES DE LINHA UTILIZANDO *DEEP LEARNING*[*]

FERRAZ JÚNIOR, Aldrumont
Instituto Federal de Educação, Ciência e Tecnologia São Paulo (IFSP)
aldrumontferrazjr@gmail.com

ATZINGEN, Gustavo Voltani
Universidade de São Paulo
gustavo.von.atzingen@gmail.com

PEREIRA, Gustavo Moura
Instituto Federal de Educação, Ciência e Tecnologia São Paulo (IFSP)
gustavopereira0@yahoo.com

BELGAMO, Anderson
Universidade Federal de São Carlos
abelgamo@gmail.com

RESUMO

A maioria dos projetos de robôs seguidores de linha de competição utiliza sensores infravermelhos para detecção da trajetória devido à simplicidade da programação e baixo custo dos materiais. No entanto, estes sensores limitam o controle do robô, uma vez que apenas captam informação no local do robô, impedindo o planejamento de ações como alteração de velocidade. O presente trabalho teve como objetivo desenvolver um algoritmo capaz de identificar e classificar disposições de trajetórias encontradas por robôs seguidores de linha, permitindo tomadas de decisões antecipadas. Foi desenvolvido um

[*] Trabalho de Conclusão de Curso de Curso apresentado em Março de 2020 no Instituto Federal de São Paulo, Campus Piracicaba, desenvolvido sob orientação de Prof. Dr. Anderson Belgamo.

Recebido em 15-02-2020 • Aceito em 15-06-2020

protótipo que realizou aquisição de imagens que então foram pré-processadas e alimentaram uma rede neural convolucional. A saída desta rede representava o tipo de pista à frente do robô, nas formas de reta, curva, cruzamento, zig-zag e parada. Após o treinamento de uma rede com 5400 imagens, obteve-se como resultado uma rede com precisão maior de 98% na classificação de novas imagens de teste.

PALAVRAS-CHAVE: *Aprendizado de máquina; inteligência artificial; rede neural convolucional; visão computacional*

INTRODUÇÃO

Instituições educacionais, buscando o aumento da aptidão científica de seus estudantes, fazem de torneios de robótica uma maneira de aumentar o interesse científico dos mesmos. Entre as categorias destas competições, uma das mais consolidadas é a de seguidores de linhas (TOSTA *et a.l*, 2016). Nesta modalidade robôs autônomos devem seguir uma linha de cor oposta à superfície ao redor (geralmente preto e branco respectivamente), encontrando situações como curvas abertas e fechadas, cruzamentos e pontos de parada. O competidor que realizar o percurso corretamente no menor tempo é o vencedor (LOPES *et al.*, 2017).

A grande maioria dos projetos de seguidores de linha utiliza como instrumento de detecção sensores infravermelhos, que identificam a trajetória a ser seguida pela reflexão de raios emitidos pelo dispositivo e por meio do processamento dessa informação se obtém a posição do robô em relação à trajetória (PAKDAMAN; SANAATIYAN, 2009). Assim, executa-se alguma ação através dos atuadores como motores. O motivo desse método ser o mais utilizado nas competições está relacionado a menor dificuldade na programação do sistema, no entanto apresenta algumas desvantagens, a saber: (i) a utilização de sensores infravermelhos que exige obrigatoriamente um alto contraste entre a linha a ser seguida e a superfície ao redor; (ii) o sensor utilizado apenas capta a posição no local do robô, impossibilitando o planejamento de algumas estratégias que necessite de informações antecipadas, como por exemplo diferenciar a velocidade do atuador em curvas e retas.

Uma metodologia mais eficiente para solucionar os problemas apresentados é a utilização de câmeras, uma vez que através de imagens capturadas um processamento digital pode ser realizado para identificar

Recebido em 15-02-2020 • Aceito em 15-06-2020

objetos e trajetórias (LORSAKUL; SUTHAKORN, 2007). Este método possibilita a captura de vários pontos da trajetória ao mesmo tempo, viabilizando a execução de ações com antecedência. Outra vantagem é que com a utilização de uma câmera, a linha da trajetória e a superfície ao redor não necessita ser especificamente preta e branca (necessário quando se utiliza sensores infravermelhos).

O processo de captar imagens e identificá-las computacionalmente remete-se ao termo Visão Computacional (MARENGONI; STRINGHINI, 2009.), área tecnológica muito pesquisada por indústrias e setores científicos devido às suas possíveis aplicações, como identificação não só de objetos, mas também de rostos em vídeos ou imagens (MOLZ, 2001).

O reconhecimento e caracterização de objetos presentes em uma imagem pode ocorrer através de algoritmos específicos para extração de características, como identificadores de formas, contornos, cores, caracteres e etc (PERELMUTER *et al.*, 1995). Outro método para reconhecimento de imagens é a utilização de Rede Neural Artificial (RNA), uma técnica de inteligência computacional que por meio de modelos matemáticos têm a capacidade de identificar e aprender padrões, tomando decisões conforme o aprendizado (GIMENEZ, 2011).

O objetivo deste projeto foi desenvolver um algoritmo para robôs seguidores de linha que, utilizando técnicas de visão computacional e rede neural artificial, seja capaz de identificar e classificar situações de trajetórias, permitindo tomadas de decisões antecipadamente.

2. FUNDAMENTAÇÃO TEÓRICA

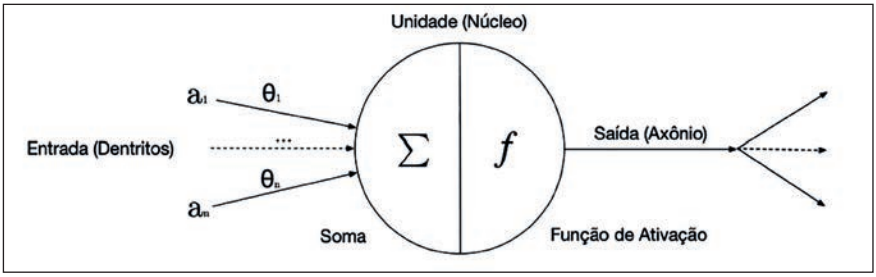
Neste capítulo foi elaborada uma revisão bibliográfica para situar e fornecer ao leitor conceitos fundamentais para o entendimento do projeto.

2.1 Redes Neurais

As RNAs são um método de computação constituído por elementos de processamento adaptativos densamente interconectados (chamados neurônios artificiais) que são capazes de realizar massivos processos paralelos (BASHEER; HAJMEER, 2000). O autor Bonon (2019, p. 10) reforça que "usando algoritmos de aprendizado de máquina, elas (redes neurais) são capazes de reconhecer padrões escondidos e correlações em dados brutos, agrupá-los e classificá-los". A **Figura 1** apresenta a representação de um neurônio artificial

Recebido em 15-02-2020 • Aceito em 15-06-2020

Figura 1 - Representação do neurônio artificial.



Fonte: Ferreira (2017, p 17)

O funcionamento dos neurônios artificiais, também denominado neurônios matemáticos, foi inspirado no sistema neural biológico. As unidades recebem sinais de entrada que são multiplicadas por pesos gerando as entradas ponderadas. A somatória destas entradas ponderadas é passada para uma função de ativação, geralmente não linear, a qual fornece o resultado final do processamento. A explicação matemática detalhada pode ser conferida no artigo do autor (FERREIRA, 2017).

Há diversas arquiteturas e aplicações de redes neurais artificiais, como as redes Perceptron (utilizadas para realizar classificações binárias) (STEPHEN, 1990), redes neurais recorrentes (aplicadas para o processamento de dados sequenciais como som e linguagem natural) (MIKOLOV *et al.*, 2010), e redes neurais convolucionais (descrita na subseção a seguir).

2.1.1 Redes neurais convolucionais.

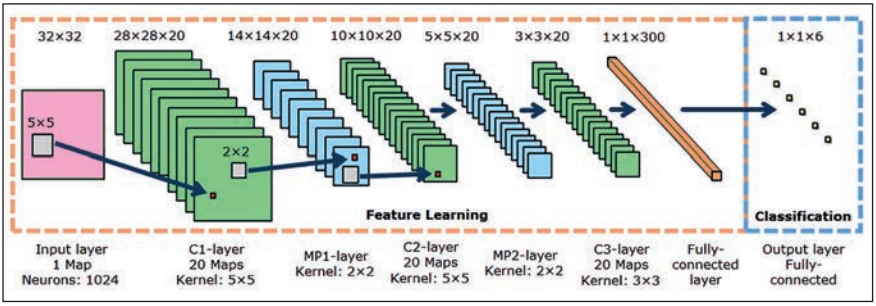
As Redes Neurais Artificiais Convolucionais (CNN) são redes que apresentam ótimo desempenho quando se trata de aplicações com reconhecimento visual. Toma-se como referência a competição internacional de reconhecimento visual *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), a qual tem como objetivo avaliar os melhores algoritmos para reconhecimento de objetos e classificação de imagem (RUSSAKOVSKY *et al.*, 2015).

Uma rede convolucional tem a capacidade de operar desde a extração de padrões da imagem até a classificação das características através de operações convolucionais, não sendo necessária a programação manual de um detector de padrões. A arquitetura requer uma grande quantidade de dados de entradas rotulados e um processador gráfico potente para acelerar o aprendizado (CHENG *et al.*, 2018).

Recebido em 15-02-2020 • Aceito em 15-06-2020

A estrutura de uma rede artificial convolucional (**Figura 2**) contém como particularidade as camadas Convolucionais e *Polling*, as quais em conjunto tem a capacidade de extrair de características de imagens para posteriormente serem classificadas em camadas totalmente conectadas. Mais detalhes sobre as redes convolucionais podem ser encontrados no artigo do autor (KRIZHEVSKY *et al.*, 2012).

Figura 2 - Representação de uma rede neural convolucional.



Fonte: NAGI *et al.(*2011, p. 2)

Para a elaboração das redes neurais utilizou-se a plataforma desenvolvida pela empresa Google denominada *Tensorflow*, a qual é fornecida como licença *open source* e permite a elaboração de diversas aplicações de aprendizado de máquina (ABADI *et al.*, 2016). Outra ferramenta utilizada foi a biblioteca *Keras*, que é uma interface de programação desenvolvida em *Python* e executada sobre a plataforma *Tensorflow*, facilitando o modelamento de redes neurais (CHOLLET *et al.*, 2015).

3. MATERIAIS E MÉTODOS.

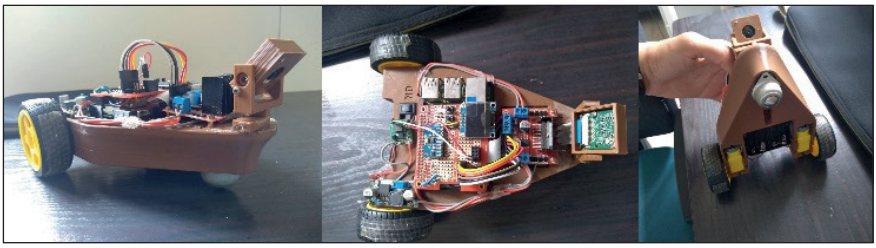
3.1 Protótipo

Foi desenvolvido um protótipo de seguidor de linha para aplicações de testes (**Figura 3**), um robô estruturado com: um chassi impresso em impressora 3D seguindo o modelo; dois motores de corrente contínua com taxa de operação entre 3 e 6 Volts; fonte de energia do sistema (quatro pilhas de 3V conectadas em série); um regulador de tensão; uma câmera de 5mp instalada a frente do chassi com um ângulo de 45 graus em relação ao solo,

Recebido em 15-02-2020 • Aceito em 15-06-2020

tendo uma visão de até 10 cm a partir do chassi, este foi o posicionamento escolhido para garantir que a câmera captura-se a pista que estivesse a frente no protótipo, e também detecta-se os pontos mais próximos do robô para melhores tomadas de decisão; um apoio esférico que possibilita a elaboração de curvas por parte do protótipo; e o microcomputador *Raspberry* Pi 3b+ para processamento das informações.

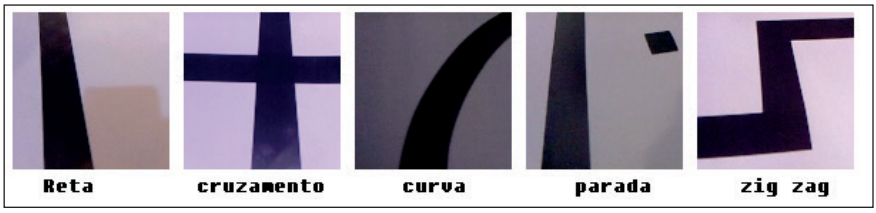
Figura 3 - Imagem do protótipo desenvolvido.



3.2 Obtenção e preparação dos dados de entrada

Para determinar quais classificações seriam realizadas pelo algoritmo de inteligência artificial desenvolvido, foram analisados alguns regulamentos de competições de seguidores de linha, como o do torneio de robótica do Instituto Federal de São Paulo (TRIF, 2019) e o do campeonato mundial de robôs Technoxian (TECHNOXIAN, 2018). Em geral, observou-se que os robôs competidores encontram situações de: parada, trajetória reta, trajetória curva, cruzamento das pistas e trajetórias com curvas 90 graus. Desta forma, a rede neural desenvolvida deve ser capaz de classificar estas situações (**Figura 4**) descritas em tempo hábil para tomada de decisões pelo sistema de controle já existente no robô.

Figura 4 - Visão do robô das situações encontradas durante o percurso.



Recebido em 15-02-2020 • Aceito em 15-06-2020

O primeiro passo para o desenvolvimento do projeto foi obter os dados de entrada e classificá-los manualmente. A quantidade de dados necessária é dependente de vários fatores como: nível de similaridade, relação entre os dados, precisão almejada, entre outros parâmetros (DOBBIN; SIMON, 2011).

Foi desenvolvido um algoritmo aplicado ao protótipo que captura 60 *frames* (480x640) por segundo e de forma manual, o robô foi movimentado em média 35 segundos sobre cada uma das cinco trajetórias presentes na **Figura 4**, captando ao todo mais de 8000 imagens. Analisou-se manualmente as imagens captadas e descartou-se os *frames* (**Figura 5**) que continham as seguintes situações: trajetória apresentando descontinuidade, *frames* com reflexos da luz e *frames* que não eram distinguíveis.

Figura 5 - Exemplos de frames classificados como inválidos

Outro procedimento necessário foi dividir os dados obtidos em *datasets* de treinamento, validação e validação cruzada. Os dois primeiros datasets são utilizados na etapa de aprendizado da rede, portanto são dados observados. Um problema que pode ocorrer é a rede ser aplicável apenas para estes dados específicos, não garantindo uma boa capacidade de generalização categorizando um *overfit* (FERREIRA, 2017).

Por este motivo é importante separar inicialmente parte dos dados para validação cruzada da rede, a forma desta divisão mais uma vez varia de acordo com os dados analisados. A literatura estudada recomenda uma divisão em que 60% a 90% dos dados sejam para o treinamento, dependendo da quantidade de dados existentes (RASCHKA, 2018).

Neste trabalho, a divisão ocorreu da seguinte maneira: 65% dos frames para Treinamento, 16% frames para validação e 19% para testes. Para garantir o não desbalanceamento da rede, preocupou-se em dividir estes dados de entrada de forma igualitária para cada categoria, conforme a **Tabela 1**.

Recebido em 15-02-2020 • Aceito em 15-06-2020

Tabela 1 - Separação dos dados por categoria

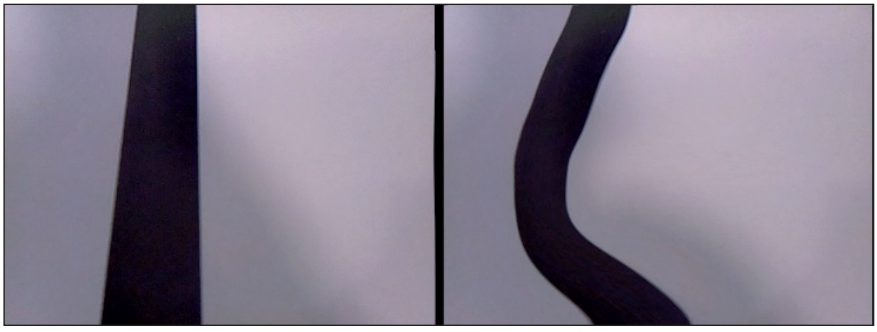
Categoria	Quantidade de Imagens		
	Treinamento	Validação	Validação cruzada
Cruzamento	864	216	250
Curva	864	216	250
Parada	864	216	250
Reta	864	216	250
Zig Zag	864	216	250

3.3 *Data augmentation*

Como citado anteriormente, as redes convolucionais exigem uma grande quantidade de dados para seu aprendizado, o que nem sempre é algo viável de se obter. Para amenizar este problema existem as técnicas de *data augmentation*, que consistem em aumentar o número de dados de forma a qual não comprometa o aprendizado (HAN *et al.*, 2018).

Em aplicações de processamento de imagem, o aumento da quantidade de *frames* pode ser gerado através da aplicação de: ruídos, translação, rotação, zoom, giros, espelho, inversão de cores, distorção, etc (HAN *et al.*, 2018). Durante o aumento de dados é importante manter as características principais que justificam a sua classificação (SALAMON; BELLO, 2017), por exemplo distorcer uma reta pode transformá-la em uma curva alterando sua classificação (**Figura 6**), por outro lado inverter as cores da mesma reta não faz com que o novo frame deixe de ser uma reta.

Figura 6 - Aplicado uma distorção a uma reta.

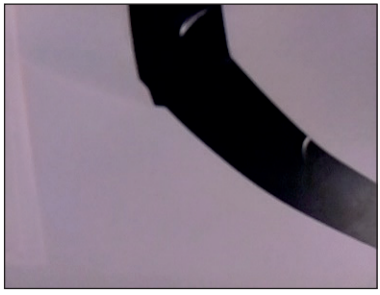


Recebido em 15-02-2020 • Aceito em 15-06-2020

Para o aumento de dados, utilizou-se a classe *ImageDataGenerator* da biblioteca *Keras*, a qual como o nome diz trata-se de uma função de geração de imagens. Os efeitos aplicados e a escolha das imagens são aplicados aleatoriamente, respeitando alguns parâmetros setados explicados adiante. O número de dados foi aumentado 3 vezes durante o treinamento. Alguns dos efeitos de utilizados são descritos a seguir:

- Rotação: durante o treinamento algumas imagens foram rotacionadas randomicamente em até 10 graus.
- *Zoom*: algumas imagens foram geradas com uma variação de até 10% de *zoom*
- Giro Horizontal: foi aplicado o efeito espelho, que inverte horizontalmente as imagens, não alterando as características determinantes para classificação da imagem. Ressalta-se que este feito só pôde ser utilizado pois as categorizações estabelecidas não consideram o lado (direita ou esquerda).
- Inversão de Cores: a inversão de cores pôde ser feita uma vez que não alteraria a classificação da imagem. Foi desenvolvido uma função específica já que não há de forma nativa na biblioteca Keras.
- Giro Vertical: o giro vertical não foi aplicado uma vez que geraria situações a qual o seguidor de linha não deve encontrar, exemplo a **Figura 7** na qual a visão apresentada o robô estaria fora da trajetória

Figura 7 - Situação a qual o robô de linha não se encontra em seu percurso.



3.4 Treinamento e modelamento da rede artificial

O modelamento da rede convolucional desenvolvida foi estruturado conforme demonstra a **Figura 8**. A rede obtém como dado de entrada uma imagem com as dimensões de 480x640 pixels (altura e largura respectivamente) e fornece como saída um vetor com as probabilidades da predição, a ordem dos dados deste vetor são respetivamente: cruzamento, curva, parada, reta, zig zag.

Recebido em 15-02-2020 • Aceito em 15-06-2020

Figura 8 - Estrutura da rede artificial convolucional.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 480, 640, 32)	3232
max_pooling2d_1 (MaxPooling2	(None, 120, 160, 32)	0
conv2d_2 (Conv2D)	(None, 120, 160, 64)	73792
max_pooling2d_2 (MaxPooling2	(None, 59, 79, 64)	0
conv2d_3 (Conv2D)	(None, 59, 79, 96)	614496
max_pooling2d_3 (MaxPooling2	(None, 28, 38, 96)	0
conv2d_4 (Conv2D)	(None, 28, 38, 128)	1228928
max_pooling2d_4 (MaxPooling2	(None, 13, 18, 128)	0
conv2d_5 (Conv2D)	(None, 13, 18, 128)	589952
max_pooling2d_5 (MaxPooling2	(None, 5, 8, 128)	0
flatten_1 (Flatten)	(None, 5120)	0
dense_1 (Dense)	(None, 512)	2621952
activation_1 (Activation)	(None, 512)	0
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
activation_2 (Activation)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 5)	2565

Alguns dos principais parâmetros definidos para o treinamento da rede foram:

- *Epochs* - Quantidade de etapas de aprendizado definiu-se 50;
- *Batch Size* - conjunto de amostras que percorrerão a rede artificial, definido como 600 imagens;
- *Steps per epoch* - quantos Batch sizes percorrerão a rede por epoch, neste caso 5 vezes (3 mil imagens serão treinadas a cada epoch) .
- *x_train* e *y_train* são os dados de treinamento
- *x_val* e *y_val* são os dados de validação

Algoritmo 1. Parte do código desenvolvido demonstrando os parâmetros utilizados

```
1. model.fit_generator(datagen.flow(x_train, y_train, batch_size=600),
2.                                  validation_data = (x_val, y_val),
3.                                  epochs = 50, steps_per_epoch = 5))
```

O treinamento foi realizado em um computador com 64Gb de memória Ram, um processador Ryzen 7 2700 (8 núcleos) e uma placa de vídeo

Recebido em 15-02-2020 • Aceito em 15-06-2020

RTX 2080 Ti. Cada *epoch* demorou cerca de 8 minutos totalizando mais de 6,5 horas de treinamento.

4. RESULTADOS E DISCUSSÃO

Seguindo os métodos descritos, a rede convolucional desenvolvida obteve uma precisão de 98,88% (1236 acertos), como detalhado na **Tabela 2.**

Tabela 2 - Tabela apresentando os acertos

Categoria	Quantidade de Acertos	Acurácia
Cruzamento	246	0,9968
Curva	249	0,9992
Parada	247	0,9976
Reta	246	0,9968
Zig Zag	248	0,9984
TOTAL	1236	

Para tentar identificar o motivo pelo qual a rede não teve uma taxa de acerto ainda maior, analisou-se quais foram as imagens classificadas erradas. A **Tabela 3** apresenta as classificações divergentes, demonstrando qual a classificação esperada, a classificação realizada pela inteligência artificial, a quantidade de erros, a quantidade de *frames* da categoria esperada e a taxa de erro.

Tabela 3 - Tabela apresentando classificações divergentes

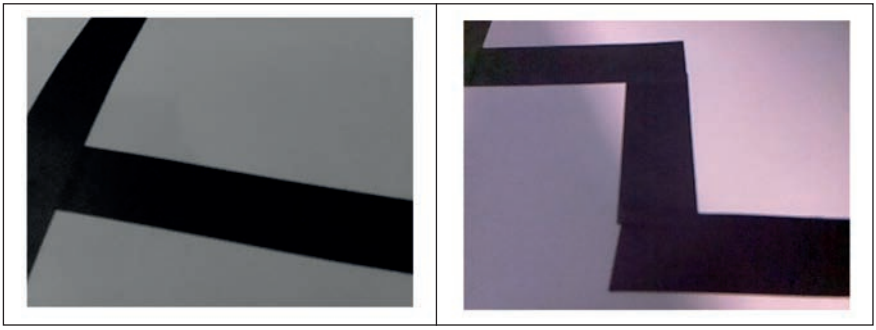
Classificação Esperada	Classificação Inteligência Artificial	Quantidade Erros	Quantidade de Frames da Categoria Esperada	Taxa de Erro
Cruzamento	Zig Zag	4	250	0,016
Curva	Zig Zag	1	250	0,004
Parada	Cruzamento	2	250	0,008
Parada	Reta	1	250	0,004
Reta	Parada	3	250	0,012
Reta	Curva	1	250	0,004
Zig Zag	Curva	1	250	0,004
Zig Zag	Parada	1	250	0,004
TOTAL		14		

Recebido em 15-02-2020 • Aceito em 15-06-2020

Através da tabela apresentada, pode se observar que quatro cruzamentos foram classificados pela rede neural como zig zag. Ao analisar estes frames observou-se que todos são semelhantes a **Figura 9 (a)**, ou seja, ambos os frames apresentam apenas metade do "X" característico de um cruzamento, exibindo características similares a de *frames* categorizados como Zig Zag (**Figura 9 (b)**).

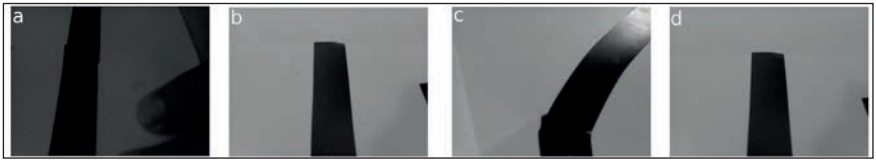
Figura 9 (a) - Frame classificado pela rede como zigzag.

Figura 9 (b): Frame categorizado zigzag.



Conforme informado na seção 3.2 (Obtenção e preparação de dados), deveriam ser descartados *frames* que continham alguma situação considerada atípica para um robô seguidor de linha. No entanto devido a pré-classificação manual da grande quantidade de imagens (8 mil), algumas destas imagens participaram indevidamente dos testes, são os casos de 4 frames presentes nas **Figuras 10**.

Figura 10 - Frames contendo situações atípicas para um robô seguidor de linha.



Nestes casos (**Figura 10**), o frame identificado como 10(c) apresenta o reflexo da luz ambiente, os frames 10(b) e 10(d) além de possuírem descontinuidades na trajetória, também apresentam uma parte muito pequena

Recebido em 15-02-2020 • Aceito em 15-06-2020

no quadrado ao lado da reta (característica fundamental para classificação de uma parada), o frame 10(a) apresenta uma situação totalmente atípica. Logo a predição incorreta destes frames são consequência de uma falha humana no pré-processamento, ou seja, se estas imagens tivessem sido corretamente descartadas a precisão da rede seria igual a 99,19%.

A pré-classificação de retas e curvas não seguiu um critério rigoroso de classificação, logo podem ter sido ora consideradas reta, ora considerado curva. É o caso da **Figura 11**, que pode gerar diferentes opiniões de classificações dos próprios observadores, neste caso o *frame* foi pré-categorizado como reta e classificado pela I.A. como curva.

Figura 11 - Frames pré-categorizado como reta e classificado pela I.A. como curva

Em robôs seguidores de linha, as situações de paradas são as que mais exigem atenção, é necessário garantir que o robô execute a ação de parada apenas ao se deparar com esta situação. Como apresentado na **Tabela 2**, considerando a existência de quatro casos de falsos-positivos (identificou incorretamente uma situação de parada) e três falsos-negativos (não identificou uma situação de parada), estes erros podem ser facilmente contornados através de um algoritmo de atuação que analise um conjunto de *frames* antes de executar determinada ação.

Assim, a probabilidade de a rede predizer um falso-negativo é igual a $1{,}2x10^{-4}$ (1,2%), já a probabilidade de ser detectado três falsos negativos é de $1{,}728x10^{-12}$. As chances da rede predizer um falso-positivo é menor ainda, uma vez que dentre os 1000 *frames* diferentes de parada, a rede identificou apenas quatro destas situações. Assim, a probabilidade de detecção de um falso-positivo é $4x10^{-3}$ e de três falsos-positivos é $6{,}4x10^{-8}$.

Desta forma, utilizando uma lógica de atuação apenas após três *frames* terem sido classificados com a mesma categoria, no caso parada, a eficiência do sistema será superior a 99,99%.

Recebido em 15-02-2020 • Aceito em 15-06-2020

O tempo para classificação de uma imagem no computador especificado na seção 2.4 (máquina que treinou a rede) é igual a 0,01 segundos. No embarcado NVIDIA *Jetson*, um *frame* é processado em 0,16 segundos, tempo de processamento satisfatório para aplicações em robôs seguidores de linha.

Estes tempos citados podem ser otimizados treinando a rede para receber imagens de entradas com resolução menor. É importante que essa redução seja realizada mantendo a proporção da altura e largura da imagem para manter as características principais do frame, não interferindo significamente no aprendizado.

Entre as melhorias que podem ser aplicadas estão: a identificação do sentido e ângulo das curvas; a adição de imagens inválidas no treinamento; e uma melhor seleção e classificação dos dados (pode utilizar a própria rede para diminuir o processo manual).

A alta eficiência (98,88%) da rede neural desenvolvida permite que em projetos futuros a própria rede realize a pré-classificação de imagens, diminuindo assim o trabalho manual, permitindo o aumento do número de dados e consequentemente aumentando a eficiência da rede.

CONSIDERAÇÕES FINAIS

Por meio dos resultados obtidos e discussão apresentada pode-se concluir que o objetivo estipulado foi satisfeito, isto é, foi desenvolvida uma rede neural que prevê cinco tipos de trajetórias (cruzamento, parada, curva, reta e zig zag) com 98,88% de precisão. Ao integrar este modelo em um sistema de atuação que considere um conjunto de frames, a eficiência do sistema pode ser superior a 99,99%, satisfazendo a necessidade de um seguidor de linha.

REFERÊNCIAS

ABADI, M. *et al*. *Tensorflow: A system for large-scale machine learning*. **12th *Symposium on Operating Systems Design and Implementation***. 2016. p. 265-283. Disponivel em https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf Acesso em outubro de 2019

BASHEER, I.; HAJMEER, M. N. *Artificial Neural Networks: Fundamentals, Computing, Design, and Application.* ***Journal of microbiological methods.*** 2000. Disponivel em https://www.researchgate.net/profile/Imad_Basheer/publication/12244015_Artificial_Neural_Networks_Fundamentals_Computing_Design_and_Application/links/5ad78cb4a6fdcc293584b69d/Artificial-Neural-Networks-Fundamentals-Computing-Design-and-Application.pdf Acesso em fevereiro de 2020

Recebido em 15-02-2020 • Aceito em 15-06-2020

BONON, C. E. de A. **. Estudo comparativo dos softwares de simulação para carros autônomos CARLA e Udacity.** Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Telecomunicações)-Universidade Federal Fluminense, 2019. Disponível em https://app.uff.br/riuff/bitstream/1/12578/1/TCC_REFIXED%20Carlos%20Eduardo.pdf. Acesso em novembro de 2019.

CHENG, J. *et al*. ***Recent advances in efficient computation of t convolutional neural networks.*** *Frontiers of Information Technology & Electronic Engineering.* 2018. p. 64-77. Disponivel em https://arxiv.org/pdf/1802.00939. Acesso em fevereiro de 2020

CHOLLET, F. *et al*. ***Keras.*** [S. l.], 2015. Disponível em: https://keras.io/. Acesso em fevereiro 2020.

DOBBIN, K.; SIMON, R.. ***Optimally splitting cases for training and testing high dimensional classifiers.*** *BMC medical genomics*. 2011. Disponível em https://www.researchgate.net/profile/Richard_Simon4/publication/51036331_Optimally_splitting_cases_for_training_and_testing_high_dimensional_classifiers/links/5419ace60cf203f155ae0ccc/Optimally-splitting-cases-for-training-and-testing-high-dimensional-classifiers.pdf Acesso em novembro de 2019

FERREIRA, A. **Estimação do ângulo de direção por vídeo para veículos autônomos utilizando redes neurais convolucionais multicanais**. 2017. Disponível em http://bdm.unb.br/bitstream/10483/17779/1/2017_ArthurEmidioTeixeiraFerreira.pdf Acesso em novembro de 2019

GIMENEZ, C. M. **Identificação de bovinos através de reconhecimento de padrões do espelho nasal utilizando redes neurais artificiais**. Dissertação (Mestrado em Qualidade e Produtividade Animal) - Faculdade de Zootecnia e Engenharia de Alimentos, Universidade de São Paulo, Pirassununga, 2011. doi:10.11606/D.74.2011.tde-24052011-085146. Acesso em fevereiro de 2020.

HAN, D. *et al*. ***A new image classification method using CNN transfer learning and web data augmentation***. *Expert Systems with Applications* 95. 2018. p.43-56. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417417307844 Acesso em novembro de 2019

KRIZHEVSKY, A. *et al*. ***Imagenet classification with deep convolutional neural networks.*** *Advances in neural information processing systems*. 2012. p.1097-1105. Disponivel em http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. Acesso em outubro de 2019

LOPES, W. A. *et al*. Projeto de robô autônomo seguidor de linha utilizando mapeamento de pista e controle híbrido. ***Anais do Computer on the Beach,*** p. 608-610, 2017. Disponível em: https://siaiap32.univali.br/seer/index.php/acotb/article/download/10544/5901. Acesso em janeiro de 2020

LORSAKUL, A.; SUTHAKORN J. *Traffic sign recognition for intelligent vehicle/driver assistance system using neural network on opencv****. The 4th International Conference on Ubiquitous Robots and Ambient Intelligence.*** 2007. Disponível em: https://www.researchgate.net/publication/228522786_Traffic_sign_recognition_using_neural_network_on_open-

Recebido em 15-02-2020 • Aceito em 15-06-2020

cv_Toward_intelligent_vehicledriver_assistance_system/link/5b0d029daca2725783ec6454/download. Acesso em janeiro de 2020

MARENGONI, M.; STRINGHINI, D. Introdução à Visão Computacional usando *OpenCV***. Revista de Informática Teórica Aplicada.** 2009. Disponível em https://seer.ufrgs.br/rita/article/view/rita_v16_n1_p125/7289. Acesso em fevereiro de 2020

MIKOLOV, T. *et al*. *Recurrent neural network based language model*. ***Eleventh annual conference of the international speech communication association***. 2010. Disponivel em https://www.researchgate.net/profile/Martin_Karafiat/publication/221489926_Recurrent_neural_network_based_language_model/links/0c960523991065d41b000000/Recurrent-neural--network-based-language-model.pdf. Acesso em fevereiro de 2020

MOLZ, R. F. **Uma Metodologia para o desenvolvimento de aplicações de visão computacional utilizando um projeto conjunto de *hardware* e *software*.** 2001. https://lume.ufrgs.br/handle/10183/2783. Acesso em fevereiro de 2020

NAGI, J. *et al*. *Max-pooling convolutional neural networks for vision-based hand gesture recognition*. ***IEEE International Conference on Signal and Image Processing Applications* (ICSIPA).** IEEE, 2011. Disponivel em https://ieeexplore.ieee.org/document/6144164. Acesso em outubro de 2019

PAKDAMAN, M.; SANAATIYAN, M. M. *Design and implementation of line follower robot.* 2009 ***Second International Conference on Computer and Electrical Engineering***. IEEE, 2009. p. 585-590. Disponivel em https://ieeexplore.ieee.org/abstract/document/5380235 Acesso em janeiro de 2020

PERELMUTER, G. *et al*. Reconhecimento de imagens bidimensionais utilizando Redes Neurais Artificiais. **Anais do VIII Sibgrapi**. 1995. p. 197-203. Disponivel em http://sibgrapi.sid.inpe.br/col/sid.inpe.br/sibgrapi/2013/02.18.15.52/doc/25%20Reconhecimento%20de%20imagens%20bidimensionais.pdf. Acesso em fevereiro de 2020

RASCHKA, S. ***Model evaluation, model selection, and algorithm selection in machine learning***. arXiv preprint arXiv. 2018. Disponível em: https://arxiv.org/pdf/1811.12808. Acesso em novembro de 2019

RUSSAKOVSKY, O. *et al*. *Imagenet large scale visual recognition challenge***. *International journal of computer vision***. 2015. p. 211-252. Disponivel em https://www.researchgate.net/profile/Hao_Su8/publication/265295439_ImageNet_Large_Scale_Visual_Recognition_Challenge/links/54cbd6a90cf29ca810f4525e/ImageNet-Large-Scale-Visual-Recognition--Challenge.pdf. Acesso em fevereiro de 2020

SALAMON, J.; BELLO, J. *Deep convolutional neural networks and data augmentation for environmental sound classification***. *IEEE Signal Processing Letters 24.3***. 2017. p. 279-283. Disponível em: https://arxiv.org/pdf/1608.04363. Acesso em novembro de 2019

STEPHEN, I. *Perceptron-based learning algorithms*. ***IEEE Transactions on neural networks*.** 1990.Disponivel em https://www.researchgate.net/profile/Steve_Gallant/publica-

Recebido em 15-02-2020 • Aceito em 15-06-2020

tion/5569039_Perceptron-based_learning_algorithms/links/0f317534430256482100000/Perceptron-based-learning-algorithms.pdf. Acesso em fevereiro de 2020

TECHNOXIAN. ***Fastest Line Follower***. [S. l.], 2018. Disponível em: https://www.technoxian.com/fastest-line-follower. Acesso em fevereiro de 2020

TOSTA, O. *et al*. **Projeto E Implementação De Um Robô Autônomo Seguidor De Linha Baseado Em Visão Computacional**, 2016. Disponível em: http://sistemaolimpo.org/midias/uploads/b46d3d2361f1b6c485fca4859ec2c4bd.pdf. Acesso janeiro de 2020

TRIF - **Torneio de Robótica do IFSP**: Regras e Instruções. [S. l.], 2019. Disponível em: https://drive.google.com/file/d/1UaKCzL25Rhg7-kHY3hpgAuTpYy0uB3Mz/view. Acesso em fevereiro de 2020

Recebido em 15-02-2020 • Aceito em 15-06-2020