Multi Camera System Analysis for Autonomous Navigation using End-to-End Deep Learning

1st José A. Diaz Amado
Pós-dout. USP (ICMC)
Professor (IFBA) - BA
sportingjada1@hotmail.com

2thd Jean Amaro
Mestrando USP (ICMC)
São Carlos, Brasil
jean.amaro@usp.br

3rd Iago P. Gomes
Mestrando USP (ICMC)
São Carlos, Brasil
iago.pg00@gmail.com

4nd Denis Wolf
Prof. USP (ICMC)
São Carlos, Brasil
denis@icmc.usp.br

5nd F. S. Osorio
Prof. USP (ICMC)
São Carlos, Brasil
fosorio@usp.br

***Resumo*—This work aims to present an autonomous vehicle navigation system, based on an End-to-End Deep Learning approach, and to study the impact of different image input configurations to the system performance. The proposed methodology in this work was to adopt and test different configurations of RGB and Depth images captured from a Kinect device. We adopted a multi-camera system, composed by 3 cameras, with different RGB and/or Depth input configurations. Two main systems were developed in order to study and validade de different input configurations: the first one based on a realistic simulator and the second one based on a mini-car (small scale vehicle). Starting with the simulations, it was possible to choose the best camera/input configuration, then we validated that using the real vehicle (mini-car) with real sensors/cameras. The experimental results demonstrated that a multi-camera solution, based on 3 cameras, allow us to obtain better autonomous navigation control results in a End-to-End Deep Learning based approch, with a very small final error when using the proposed camera configurations.**

***Index Terms*—Deep Learning, End-to-End, Self-Driving Car, Image based Navigation, RGB+Depth.**

I. Introdução

Robôs autônomos são desenvolvidos para se movimentar livremente dentro de um determinado ambiente, sem intervenção humana, como por exemplo, os carros autônomos. [1]

A crescente concorrência envolvendo grandes empresas de tecnologia e da indústria automobilística está tornando a pesquisa e desenvolvimento de carros autônomos cada dia mais desafiadora e interessante. Desafiadora pois já ocorreram até registros de acidentes fatais [2], sinalizando assim, que está tecnologia ainda não está pronta para ser entregue ao mercado. Interessante porque, a cada dia, vemos um incremento do número de empresas e de universidades trabalhando e pesquisando neste tema [3]. As pesquisas mais relevantes em carros autônomos estão nas áreas de navegação, detecção de obstáculos e localização. O aprendizado *End-to-End* destaca-se e tem sido aprimorado em função do bom desempenho das técnicas de Inteligência Artificial neste últimos anos. [4]

Aprendizado Profundo (AP), comumente chamado de *Deep Learning*, é uma ramificação do campo de Aprendizado de Máquina, *Machine Learning* da Inteligência Artificial. O AP, se refere aos algoritmos e técnicas que extraem informações complexas de dados brutos a partir da combinação de características mais simples, combinadas em múltiplos níveis e obtidas por Aprendizado Automático. Assim, quanto mais etapas, maiores serão as combinações e a profundidade do aprendizado. Um dos algoritmos de AP de destaque são as Redes Neurais Convolutivas (RNC), *Convolutional Neural Networks* (CNN). [5]

O uso de Redes Neurais em Visão Computacional aplicada em sistemas de navegação autônoma teve início em 1989 com o ALVINN (*Autonomous Land Vehicle in a Neural Network*), veículo terrestre autônomo com uma Rede Neural. Este utilizou uma rede de três camadas para poder seguir adequadamente uma estrada, utilizando imagens de uma câmera e um localizador, e como saída os valores para o controle da direção do carro. [6]

Em 2004, foi proposto o DAVE (*DARPA Autonomous Vehicle*), veículo autônomo do DARPA (*Defense Advanced Research Projects Agency*), Agência de Projetos do Depto. de Defesa dos Estados Unidos. Nesse projeto, foi usado um carrinho de controle remoto elétrico, com duas câmeras de vídeo (visão estereoscópica), que deveria navegar de forma autônoma em um ambiente ao ar livre com árvores, rochas, paredes e outros obstáculos. A sua navegação foi proposta a partir de uma RNC de seis camadas. [7]

O projetos DAVE e ALVINN demonstraram que uma Rede Neural poderia dirigir um carro em estradas, onde posteriormente a empresa NVIDIA (fabricante de GPUs), publicou em Abril de 2016 alguns resultados de seu trabalho com carros autônomos. Neste projeto foi desenvolvida uma arquitetura RNC, conhecida como "Pilotnet", treinada a partir dos dados brutos de uma câmera (RGB) juntamente com os valores de direção, executados por uma pessoa no carro (*End-to-End Learning*). Esta abordagem mostrou-se eficiente para um sistema que aprendeu a dirigir em estradas com e sem faixas, em estacionamentos com pouca visibilidade e em estradas não pavimentadas. Nesta experiência, foram usadas 72 horas de vídeo, como *dataset*. [8] [9]

Nos artigos [10], [11], destaca-se a incorporação de câmeras monoculares RGB em diferentes posições do veículo, para ajudar no processo de localização em um mapa. A fusão da odometria, dados do GPS e das imagens foram importantes neste processo de aprimoramento para a localização do carro. Já em [12] ele melhora a estimativa de movimento de um carro, onde é utilizada a odometria visual. É apresentado um método baseado em um processo de aquisição não sincronizado de imagens, obtidas em diferentes posições do carro, por meio de câmeras monoculares RGB. A aquisição sempre

é feita por 3 imagens de diferentes câmeras, mantendo um padrão de ligação em forma triangular. O seguinte artigo, mostrou bons resultados para o desenvolvimento de localização e percepção em ambientes 3D com o objetivo de implementar um processo de navegação autônoma utilizando *deep learning* [13], por meio de um único sensor vindo de um conjunto de câmeras e fornecendo um sistema multivisão com cobertura de *360 graus*. Em outra abordagem, no artigo [14], foi analisado o problema de perspectiva focal de diferentes imagens em um determinado ambiente, fornecido por câmera monoculares RGB de uso externo, que juntas geram distúrbios entre elas, como por exemplo, ruído. Um método foi proposto para resolver este problema, além de recomendações de posicionamento físico das câmeras, para diminuir esta interferência. Finalmente, podemos citar [15] [16], que apresenta um sistema *"End-to-End"* para um carro convencional autônomo, utilizando câmeras com cobertura de 360 graus de visão, além de um planejador de rotas.

No estado-da-arte apresentado, foram descritas diversas abordagens de incorporação de um sistema multi-câmeras, as quais são utilizadas para definir posição, localização e processos de autonomia de um carro. Podemos ver também que são utilizadas imagens RGB e de profundidade, para chegar a estes objetivos. No entanto, não foi analisada separadamente a incorporação destas imagens, aos sistemas propostos. Fica a dúvida se realmente era necessária a incorporação destas, como também a importância desta fusão de dados no processo de navegação.

Este trabalho propõe a implementação da arquitetura conhecida como Pilotnet [8], para navegação de veículos autônomos. Dada a considerável capacidade das Redes Neurais Profundas em abstrair padrões e características de uma imagem, além de explorar os benefícios do aprendizado *End-to-End*, as redes propostas foram submetidas a diferentes fontes de entrada, com o objetivo de comparar e definir as que tiveram um melhor desempenho no processo de controle autônomo do veículo, e assim definir a importância de cada imagem adquirida para esta finalidade. As fontes de entradas estarão formadas por imagens RGB e de profundidade. Estas informações serão obtidas por 3 câmeras posicionadas no carro. Logo após será apresentada a metodologia utilizada nesta proposta, além de resultados de validação.

II. MATERIAIS E MÉTODOS

A metodologia do trabalho pode ser dividida em duas partes: primeiro vamos definir o processo de treinamento, utilizando diversas abordagens. Na segunda parte mostraremos como foram obtidas as imagens, sejam por ferramentas computacionais (simulação) ou por meio de um protótipo automotivo em escala, para obter dados reais.

A. 3.1 Metodologia do processo de treinamento

A arquitetura da RNC implementada é conhecida como Pilotnet da NVIDIA, visto que já foi testada e validada para a aplicação específica em um sistema de navegação autônoma. A (Fig. 1) mostra a arquitetura, que utiliza imagens coloridas

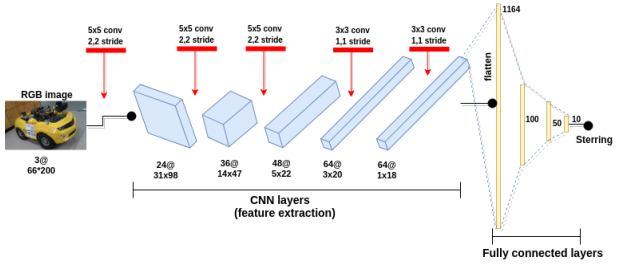

Figura 1. Arquitetura - Pilotnet.

com seus 3 canais e uma resolução de 66x200 pixels. Esta rede possui 3 camadas convolutivas com *kernel* de 5x5 e 2 com *kernel* de 3x3, seguidas pela *Flatten*, que corresponde aos 1152 mapas de características com dimensão 1x1 que são passados para as camadas totalmente conectadas [8]. Para estimar o erro da rede foi utilizada a média dos erros quadráticos e como processo de otimização foi utilizado o *"Adam optimization"*. A implementação da CNN foi baseada no código do *github*, utilizado no artigo. [4], a qual teve que ser adaptada para o processo que será explanado a seguir.

O projeto foca no elemento de análise e interpretação do resultado, que seriam as imagens obtidas pelas câmeras. Por este motivo, foram utilizadas 3 câmeras posicionadas nas partes frontal e laterais do carro. A câmera escolhida foi o *kinect v2* [17], porque é capaz de fornecer imagens RGB coloridas, como também imagens de profundidade.

Para definir a posição das câmeras no carro, utilizamos como critério as posições que maximizam o maior campo de visão do ambiente do entorno do carro, como também foram consideradas as recomendações apresentadas em [14]. A figura a seguir mostra as câmeras posicionadas no carro.

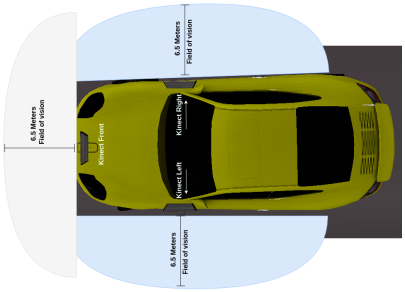

Figura 2. Posição de câmeras no protótipo.

Levando em consideração as caraterísticas da Pilotnet, que originalmente só recebe imagens RGB, foi necessário transformar as imagens de profundidade do *kinect v2*, que originalmente utiliza só um canal como resposta de dados *depth*, para 3 canais. Estas imagens de profundidade foram codificadas da escala de cinza para uma escala de variações de matiz(H), com saturação(S) e valor(V) fixos (padrão HSV), convertendo finalmente para 3 canais RGB. Uma vez obtidas as imagens, foram utilizados os seguintes critérios de combinações possíveis entre elas para gerar uma imagem RGB de entrada na pilotnet:

- M_1 - RGB(F): Imagem RGB da câmera frontal.
- M_2 - RGB-D(F): Imagem RGB-Depth da câmera frontal.
- M_3 - RGB-D(L) | RGB-D(F): Concatenação da Imagem RGB-Depth da câmera esquerda e Imagem RGB-Depth da câmera frontal.
- M_4 - RGB-D(F) | RGB-D(R): Concatenação da Imagem RGB-Depth da câmera frontal e Imagem RGB-Depth da câmera direita.
- M_5 - RGB(L) | RGB(F): Concatenação da Imagem RGB da câmera esquerda e Imagem RGB da câmera frontal.
- M_6 - RGB(F) | RGB(R): Concatenação da Imagem RGB da câmera frontal e Imagem RGB da câmera direita.
- M_7 - RGB-D(L) | RGB-D(F) | RGB-D(R): Concatenação da Imagem RGB-Depth da câmera Esquerda, Imagem RGB-Depth da câmera frontal e Imagem RGB-Depth da câmera direita.
- M_8 - RGB(L) | RGB(F) | RGB(R): Concatenação da Imagem RGB da câmera Esquerda, Imagem RGB da câmera frontal e Imagem RGB da câmera direita.
- M_9 - RGB-D(L) | RGB(F) | RGB-D(R): Concatenação da Imagem RGB-Depth da câmera Esquerda , Imagem RGB da câmera frontal e Imagem RGB-Depth do câmera direita.
- M_{10} - RGB(L) | RGB-D(F) | RGB(R): Concatenação da Imagem RGB da câmera Esquerda, Imagem RGB-Depth da câmera frontal e Imagem RGB-Depth do câmera direita.

Cada item que foi apresentado, representa uma imagem colorida de 3 canais. Todas as transformações das imagens foram concatenadas na horizontal e feitas em tempo real, tanto na aquisição de imagens para o *dataset*, como também para executar o processo de validação pela Pilotnet.

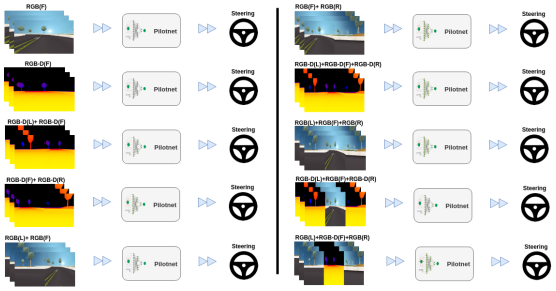

Figura 3. Pilotnet com imagens de 3 canais como entrada.

Após modificar a entrada da Pilonet de 3 canais para 6 canais, as imagens RGB e RGB-D das três câmeras, foram concatenadas simultaneamente e foram gerados novos dados para entrada de nossa rede, utilizando o seguinte critério:

- N_1 - (RGB(F) + RGB-D(F)): juntamos M_1 e M_2.
- N_2 - (RGB-D(L) | RGB-D(F)) + (RGB(L) | RGB(F)): juntamos M_3 e M_5.
- N_3 - (RGB-D(F) | RGB-D(R)) + (RGB(F) | RGB(R)): juntamos M_4 e M_6.
- N_4 - (RGB-D(L) | RGB-D(F) | RGB-D(R)) + (RGB(L) | RGB(F) | RGB(R)): juntamos M_7 e M_8.

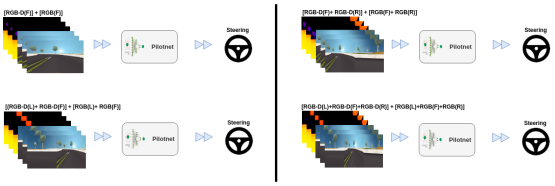

Figura 4. Pilotnet com imagens de 6 canais como entrada.

B. Obtenção dos Dados de Treinamento

Para o sistema de aprendizado, foi necessária a criação de um conjunto de dados, referido como *dataset*. O *dataset* está constituído por dados rotulados utilizados no treinamento, compostos por uma entrada e a sua respectiva saída. No presente trabalho, o *dataset* inclui imagens da trajetória percorrida a partir de um controle manual e os respectivos valores de controle de direção, que foram definidos em uma faixa de -45 *graus* a 45 *graus*, sendo os valores positivos correspondentes ao giro da direção à direita, e negativos à esquerda, sendo zero o ponto central. Esta etapa foi realizada de forma simulada, como também de forma real, que será apresentada a seguir.

1) Simulação computacional: Para implementar e aprimorar a metodologia apresentada, foi necessária a utilização de uma ferramenta de simulação computacional realista, que nos permitisse trabalhar com Robótica Móvel e Inteligência Artificial. A ferramenta adotada foi o V-REP 3.5, o qual é um software gratuito de simulação robótica, que conta com uma interface muito agradável, fácil de manipular e o mais importante, que nos permitiu gerar ambientes semelhantes aos utilizados pelo protótipo real [18]. Outra ferramenta computacional essencial para o desenvolvimento deste projeto foi o Sistema Operacional Robótico, conhecido como ROS, adotado na simulação bem como também com o veículo real. Ele permite interagir com diversos outros programas computacionais, facilitando o processo de aquisição e transferência de dados [19]. Foram desenvolvidos diversos ambientes simulados na plataforma V-Rep, com comunicação ROS e Python, com o objetivo de testar o melhor desempenho da Pilotnet. O ambiente de simulação utilizado para este projeto, pode ser visto na (Fig. 5). O carro simulado, tem caraterísticas mecânicas do modelo *Ackermann* [20], odometria, além das três câmeras *kinect* que foram incorporados ao sistema. Para gerar os dados de esterçamento no treinamento da Rede Neural, foi utilizado um volante de jogos virtuais, o qual fornece dados mais próximos da condução real de um veículo. Os processos de aquisição e execução da Rede Neural, podem ser vistos na (Fig. 6) e (Fig. 7) respectivamente.

2) Implementação Física: O protótipo desenvolvido para este projeto tinha como principal objetivo ser de baixo custo e que fosse possível uma transferência das tecnologias desenvolvidas nele para carros autônomos convencionais. Pensando nesta proposta, foi tomado como ponto de partida a procura de uma plataforma que tivesse as caraterísticas mais próximas de um carro convencional, e que nos permitisse aproveitar a estrutura mecânica deste. Após uma breve pesquisa no

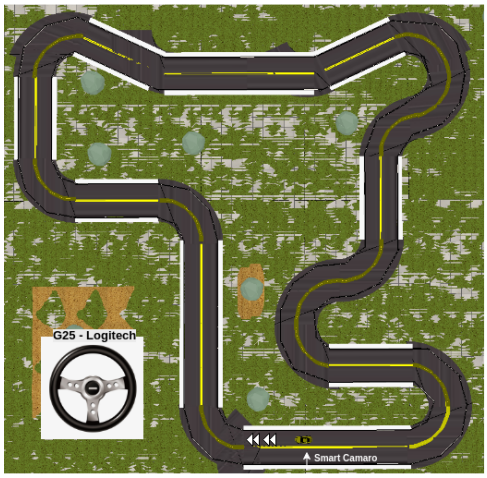

Figura 5. Pista de deslocamento do protótipo automotivo no V-REP

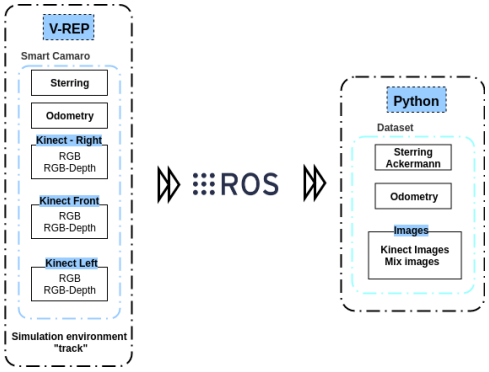

Figura 6. Aquisição de dados Vrep-ROS-Python.

mercado local, determinamos utilizar um protótipo de carro elétrico de escala 1/4 [21]. Este protótipo nos proporcionou a adoção do modelo de *Ackermann* [20], presente na maior parte dos veículos motorizados, além de um chassis de plástico resistente e com formato de um carro convencional, modelo "Camaro". Por este motivo, o protótipo recebeu o nome de **Smart Camaro**. Em seguida, foram feitas diversas alterações na estrutura deste, adaptando e instrumentando o veículo.

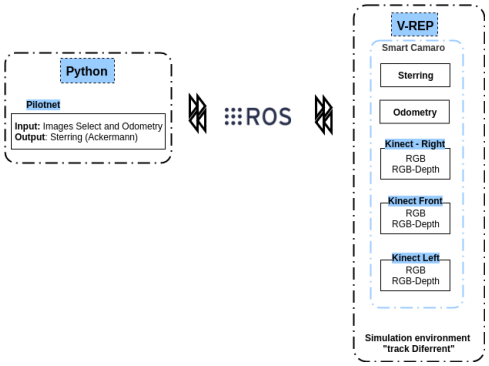

Figura 7. Run Pilotnet Python-ROS-Vrep.

Entre as mais importantes modificações, podemos citar a troca de pneus de plástico por pneus de borracha, dando maior aderência ao chão. Outra modificação foi a substituição do motor DC por um servomotor na direção do protótipo, com a finalidade de obter dados de esterçamento com maior precisão. Como último destaque mecânico, podemos citar o acoplamento de um sistema de medição de velocidade na roda traseira e a incorporação das câmeras.

A arquitetura de software do protótipo desenvolvido pode ser vista na (Fig.8). O gerenciador principal da arquitetura apresentada, foi implementado na placa de desenvolvimento *Jetson TX2 Developer Kits* [22]. Os sensores e atuadores incorporados no sistema foram: Ultrassons, GPS (Sistema de Posicionamento Global), IMU (Unidade de Medida Inercial), Encoder, Medidor de tensão/corrente do sistema, LIDAR, câmeras monoculares RGB e Kinect, Motor DC e servomotor DC. A placa de desenvolvimento conhecida como Arduino Mega, funcionou como um sub-gerenciador do sistema, no processo de aquisição e controle de atuadores. Para a comunicação sem fio, foram desenvolvidas três opções diferentes, a primeira utilizando comunicação de rádio frequência "RF", com distância de atuação de 100 metros. A segunda opção foi via "bluetooth" por meio de um controle utilizado para jogos virtuais e finalmente a comunicação "sem fio Wi-Fi", utilizando um roteador no sistema. Todo o processamento de informação foi feito na plataforma Linux 16.04, com incorporação do ROS (Sistema Operacional Robótico) e Python/C++. Na (Fig. 9), podemos ver o protótipo desenvolvido em escala 1/4.

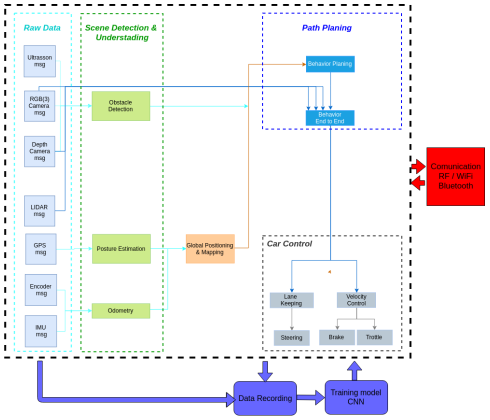

Figura 8. Arquitetura do Protótipo de carro autônomo de escala 1/4, Smart Camaro.

RESULTADOS

Como critérios de validação da metodologia apresentada, levamos em conta o valor de acurácia entre os valores desejados e os obtidos pelas redes, por meio de um *dataset* de validação. Outro critério de importante análise, foi o comportamento odométrico do carro no percurso definido na (Fig. 5), além do tempo que cada rede demorou para completar o percurso total da pista. Após analisar os critérios apresentados na metodologia, na (Tab. I) podemos ver os melhores resultados

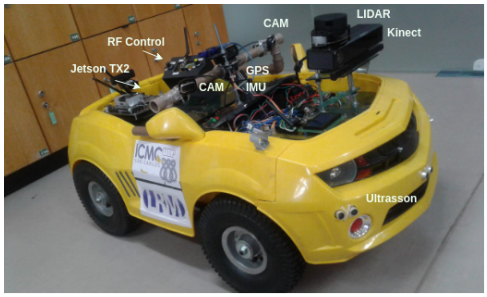

Figura 9. Protótipo de carro autônomo de escala 1/4, Smart Camaro.

em função a sua acurácia (simulação). A formula está dada a seguir:

$$\bar{E} = \left(\frac{\sum |y - \hat{y}|}{number\_images} \right) \quad (1)$$

Onde, $\bar{E}$ é o erro médio, y é a saída desejada, e $\hat{y}$ é a saída estimada pela CNN.

Na tabela I mostramos os erros médios em graus, para as possíveis combinações de imagens RGB e profundidade, obtidas pelas três câmeras. O melhor resultado, com imagens concatenadas no eixo horizontal, é a combinação M_8 com $2.41°$, e nas imagens concatenadas por canais, a combinação N_1 com $3.73°$. A Figura 10 mostra a comparação entre o ângulo de direção M_8 e N_1 durante a navegação, e vemos que ambas tem um comportamento similar.

Tabela I
ACURÁCIA DAS CNNS COM IMAGENS SIMULADAS

CNN	Mean Error	Lap Time	CNN	Mean Error	Lap Time
M_1	$3.35°$	03:38min	M_8	$\mathbf{2.41°}$	03:03min
M_2	$3.82°$	**03:00min**	M_9	$4.35°$	04:07min
M_3	$4.25°$	04:33min	M_{10}	$4.06°$	NC
M_4	$4.84°$	05:14min	N_1	$3.73°$	04:03min
M_5	$3.74°$	03:35min	N_2	$4.80°$	NC
M_6	$3.61°$	04:10min	N_3	$3.82°$	06:58min
M_7	$4.75°$	05:36min	N_4	$4.10°$	NC

Os melhores resultados com relação a odometria, são apresentada na (Fig. 11). Foram também enumeradas as vezes em que as redes saíram da trajetória original (erro de trajetória), sendo: M8=0; N1=6; M2=4; M1=3 e M6=3 vezes. Neste cenário se destaca a rede M8 com imagens [RGB(L)+RGB(F)+RGB(R], a qual apresenta um comportamento odométrico bem próximo ao da referência. Só duas redes não conseguirem concluir o percurso:M_{10} , N_2 e N_4.

No veículo real, por limitações de hardware, não foi possível adotar 3 Kinects como apresentado na simulação. Um *kinect v2* foi utilizado na parte frontal do protótipo e duas câmeras monoculares RGB nas laterais (webcams). Todas as possíveis combinações fornecidas pelas câmeras foram então implementadas. Vemos na (Fig. 12) algumas delas. Na (Fig. 13), podemos ver os melhores resultados obtidos na

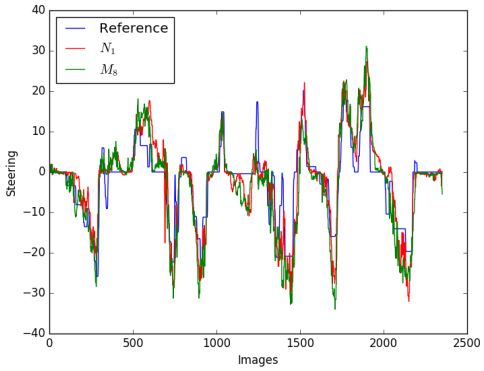

Figura 10. Angulo de direção da M_8 e N_1 no dataset de validação

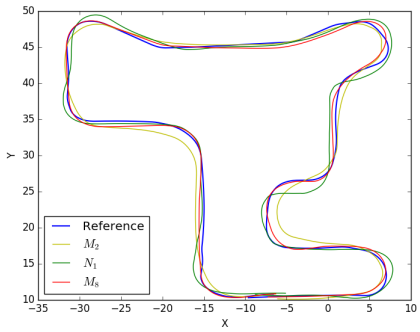

Figura 11. Odometria

validação das redes implementadas em um ambiente real, utilizando a plataforma **Smart Camaro**. Podemos destacar o comportamento da rede M10 que trabalhou com imagens RGB(L)+RGB-D(F)+RGB(R), teve um comportamento mais próximo da referência, além de obter a maior porcentagem de acurácia de todas as redes testadas, ver (Tab. II).

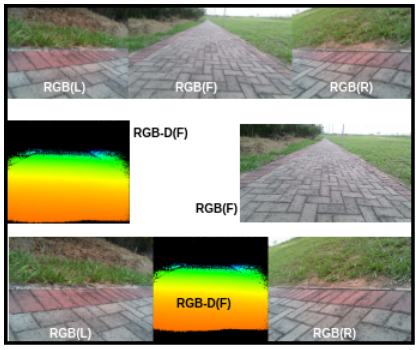

Figura 12. Imagens reais de 3 canais RGB

CONCLUSÕES

Por meio das imagens das três câmeras, foram testadas todas as possíveis combinações entre elas, com o objetivo de procurar o melhor desempenho da rede Pilotnet no processo de

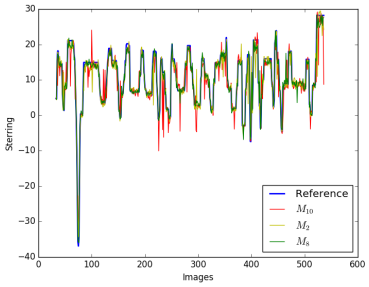

Figura 13. Validação das CNNs reais

Tabela II
ACURÁCIA CNNS REAIS

CNN	Erro	CNN	Erro
M_2	1.24%	M_{10}	1.07%
M_1	6.8%	M_8	1.59%

obter uma direção autônoma eficiente. Através da simulação, foram testadas 14 configurações de entradas e foi constatado que 7 delas apresentaram um desempenho regular para concluir a trajetória, 5 delas tiverem resultados muito bons na acurácia, odometria e no tempo do percurso. Finalmente, duas delas não conseguiram ser adequadas para o processo de autonomia. Graças aos bons resultados das simulações, foi possível implementar algumas destas abordagens em um protótipo real. O melhor resultado das redes com dados reais foi aquela que incorporou as imagens laterais (RGB) e imagem de profundidade na parte frontal.

Levando em consideração os resultados apresentados nas simulações como também na implementação real, podemos constatar a importância de trabalhar com 3 câmeras em um processo de navegação autônoma por meio de imagens. Trabalhamos com imagens RGB e de profundidade, e a combinação delas, e isto nos permitiu obter melhorias na odometria, como também na acurácia de desempenho. Existem algumas divergências nos resultados obtidos entre a simulação e a implementação real, os quais podem ser justificados pela qualidade de dados entregues a rede. Na simulação as imagens sintéticas são quase perfeitas, já na implementação real são imagens com ruído.

Nem todas as possíveis combinações de imagens, apresentarem bons resultados, levando a conclusão que o sistema pode estar sobrecarregado com informações que não são uteis. Outro ponto importante, foi o fato de que pelo menos 5 redes tiverem um desempenho aceitável na autonomia do carro, tornando possível a implementação de sistemas redundantes, caso uma das câmeras deixe de funcionar. Finalmente, com os resultados apresentados, podemos afirmar quão importante é em um sistema autônomo contar com uma cobertura de imagens ao redor do carro, mas considerando que nem todas irão fornecer informações relevantes para o melhor desempenho autônomo.

AGRADECIMENTOS

Agradecemos a NVidia pela doação da placa Jetson TX2 por meio do projeto *GPU Grant Program*.

REFERÊNCIAS

[1] R. Siegwart and I. Nourbakhsh, *Introduction to autonomous mobile robots*. Editora Mit Press, 2011.
[2] A. EFE, "Relatório aponta falha de carro autônomo do uber em atropelamento fatal," 2018. [Online]. Available: https://g1.globo.com
[3] ——, "O futuro da garagem no mundo dos carros autônomos," 2018. [Online]. Available: https://epocanegocios.globo.com
[4] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," *IEEE Intelligent Vehicles (IV)*, pp. 1856–1860, 2017.
[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[6] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems 1*, D. Touretzky, Ed. Morgan Kaufmann, January 1989.
[7] Y. Lecun, E. Cosatto, J. Ben, U. Muller, and B. Flepp, "Dave: Autonomous off-road vehicle control using end-to-end learning," Courant Institute/CBLL, http://www.cs.nyu.edu/˜yann/research/dave/index.html, Tech. Rep. DARPA-IPTO Final Report, 2004.
[8] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars." *CoRR*, vol. abs/1604.07316, 2016.
[9] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.
[10] D. Gruyer, R. Belaroussi, and M. Revilloud, "Accurate lateral positioning from map data and road marking detection." *Expert Syst. Appl.*, vol. 43, pp. 1–8, 2016.
[11] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Motion estimation for self-driving cars with a generalized camera." in *CVPR*. IEEE Computer Society, 2013, pp. 2746–2753.
[12] R. Mhiri, P. Vasseur, S. Mousset, R. Boutteau, and A. Bensrhair, "Visual odometry with unsynchronized multi-cameras setup for intelligent vehicle application," *IEEE Intelligent Vehicles (IV)*, pp. 1339–1344, 2014.
[13] L. Heng, B. Choi, Z. Cui, M. Geppert, S. Hu, B. Kuan, P. Liu, R. M. H. Nguyen, Y. C. Yeo, A. Geiger, G. H. Lee, M. Pollefeys, and T. Sattler, "Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system," *CoRR*, vol. abs/1809.05477, 2018.
[14] L. Kneip, P. T. Furgale, and R. Siegwart, "Using multi-camera systems in robotics: Efficient solutions to the npnp problem," *IEEE Intl. Conf. on Robotics and Automation*, pp. 3770–3776, 2013.
[15] S. Hecker, D. Dai, and L. V. Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in *ECCV*, 2018.
[16] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," *CoRR*, vol. abs/1811.10119, 2018.
[17] A. Goral and A. Skalski, "Accuracy assessment of kinect for xbox one in point-based tracking applications." in *ICMV*, ser. SPIE Proc., A. Verikas, P. Radeva, and D. P. Nikolaev, Eds., vol. 9875. SPIE, 2015, p. 987522.
[18] R. Kozak and R. Berggren, "Virtual reality educational pathfinders (vrep)," *2013 3rd Interdisciplinary Engineering Design Education Conference*, pp. 19–22, 2013.
[19] M. Quigley, B. P. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros : an open-source robot operating system," 2009.
[20] J. Hrbacek, T. Ripel, and J. Krejsa, "Ackermann mobile robot chassis with independent rear wheel drives," *Proceedings of 14th International Power Electronics and Motion Control Conference EPE-PEMC 2010*, pp. T5–46–T5–51, 2010.
[21] B. Bandeirante, "Camaro." [Online]. Available: http://brinquedosbandeirante.com.br/produtos/camaro-amarelo-rc-el-6v/
[22] C. C. Smith, "Implementing full device cloning on the nvidia jetson platform."