Trabalho final da disciplina de Robótica Móvel (CI1020): Controle de um carro autônomo através de uma rede neural no ambiente de simulação da Skoods na modalidade *Racing Car*

Allan Cedric G. B. Alves da Silva
Universidade Federal do Paraná
Curitiba - PR, Brasil
acgbas19@inf.ufpr.br

1 INTRODUÇÃO

Atualmente, o segmento de carros autônomos está ganhando cada vez mais espaço em várias empresas de tecnologia, como a Tesla e a Waymo. A pesquisa nesse setor alia diversas áreas do conhecimento, como IA (Inteligência Artificial) e IoT (Internet das Coisas), além de conhecimentos em áreas da Engenharia Mecânica e em muitos outros.

Este artigo tem como ênfase a aplicação de um controle de um carro autônomo no ambiente simulado da Skoods, na modalidade *Racing Car*. Esse controle é realizado através de uma rede neural convolucional, utilizando-se de uma técnica para *Deep Learning* conhecida como *Behavioral Cloning*.

2 AMBIENTE DE SIMULAÇÃO

A Skoods possui um sistema completo e uma simulação feita em *Unreal Engine* integrados com o projeto *open source* de simulação de veículos autônomos, Microsoft AirSim. Este último possui APIs de comunicação com um ambiente simulado feito em *Unreal Engine*, e possui um controle extremamente intuitivo no que tange a área de programação.

Além disso, o Microsoft AirSim fornece um grande conjunto de APIs relacionadas a visão computacional, cinemática do veículo autonômo e muito mais. As referências em relação ao sistema da Skoods e ao Microsoft AirSim se encontram a seguir, vale a pena dar uma conferida.

Skoods - *GitHub*
Microsoft AirSim - *GitHub*

3 METODOLOGIA

Para esse trabalho, foi proposto o controle de um carro autônomo a partir de uma rede neural convolucional que recebe a informação de um sensor exteroceptivo, no caso uma imagem de uma câmera, e prediz o ângulo de direção do carro em relação as rodas, conhecido como *steering angle*. No mais, o controle da velocidade, realizado a partir do *throttle* (acelerador) do carro, foi modelado como uma lógica linear tendo como parâmetro o próprio *steering angle*.

A partir disso, será discorrido agora sobre a construção do *dataset* com uma breve exploração de dados com os diferentes *datasets* produzidos, além de retratar a técnica para *Deep Learning* conhecida como *Behavioral Cloning* e a construção do modelo neural convolucional utilizado para o controle do *steering angle* do carro. Além de outras integrações que foram feitas no código principal da Skoods para a simulação.

Toda a estrutura implementada em *Python* para simulação pode ser encontrada nesse link:

Trabalho final - *GitHub*

3.1 Construção do *dataset*

Como falado anteriomente, foi utilizada a técnica conhecida como *Behavioral Cloning*, a qual se fundamenta em um computador conseguir imitar um determinado comportamento ou ação humana, no caso o ato de controlar um volante (*steering wheel*) de um carro.

Para realizar isso, é necessário gravar o comportamento típico do carro em uma trajetória, no caso a pista da simulação. Neste trabalho, foi utilizado um par de informações: a imagem da câmera frontal do carro e o *steering angle* associado.

No que tange a gravação dos dados, houve um problema no funcionamento do recurso, *Data Recording*, do Microsoft AirSim. Para contornar esse problema, foi modelado uma classe em *Python*, a qual imita a ferramenta de gravação de dados do Microsoft AirSim. E essa classe se encontra no arquivo, *airsim_dataset.py*.

Para realizar o movimento típico do carro na pista, foi utilizado o próprio controle PID fornecido pelo código da Skoods em consonância com um arquivo de *waypoints* - conjunto de pontos geográficos. No mais, a gravação foi realizada a cada atualização do estado do carro na pista. Para informações mais detalhadas, consulte o script, *__main__.py* e o pacote *garage/* da Skoods.

No total, foram construídos 4 datasets específicos:

- *run-center-scene/* (38964 imagens)
- *run-center-seg/* (27394 imagens)
- *run-center-surf/* (31109 imagens)
- *run-fast4-surf/* (26088 imagens)

Cada um desses *datasets* possuem um diretório chamado *images/*, o qual contém imagens (256x144 - formato *png*) da câmera frontal do carro. E um arquivo de log no formato *tsv - Tab Separated Values* -, apresentando duas colunas: *Steering* e *ImageName*, respectivamente. A seguir um exemplo de uma parte de um arquivo de log:

raw_data > run-center-scene > airsim_rec.txt

	Steering	ImageName
2	0.000000	img_0.png
3	-0.000268	img_1.png
4	-0.000058	img_2.png
5	-0.000058	img_3.png
6	-0.000058	img_4.png
7	-0.000058	img_5.png

Figure 1: As 7 primeiras linhas do arquivo de log do *dataset, run-center-scene/*.

Cada *dataset* será utilizado para definir um certo modelo neural distinto. Todos esses *datasets* estão dentro do diretório, *raw_data/*.

3.1.1 *run-center-scene/*.

Esse *dataset* foi produzido a partir do arquivo de *waypoints*, *run-center.pickle*, o qual foi produzido especificamente para esse trabalho. A principal característica de seus dados é uma trajetória mais estável na pista, a fim de ajudar no aprendizado da rede neural.

A palavra *scene* é relacionada ao campo de visão computacional, e basicamente retrata o tipo de imagem utilizada pela câmera. Nesse caso uma imagem comum (*airsim.ImageType.Scene*), como mostra a figura a seguir:

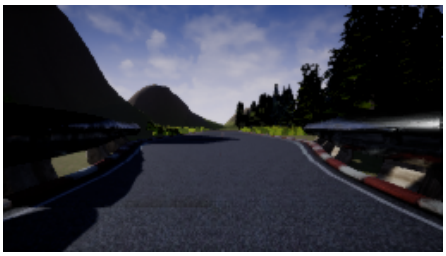

Figure 2: Exemplo do tipo de imagem usado nesse *dataset*.

3.1.2 *run-center-seg/ e run-center-surf/*.

Esses dois *datasets* possuem o mesmo racicínio do *dataset* anterior. Basicamente *seg* vem do tipo *airsim.ImageType.Segmentation*, e *surf* do tipo *airsim.ImageType.SurfaceNormals*.

Figure 3: Exemplo do tipo de imagem usado no *dataset, run-center-seg/*.

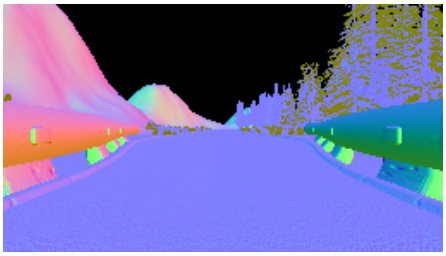

Figure 4: Exemplo do tipo de imagem usado no *dataset, run-center-surf/*.

3.1.3 *run-fast4-surf/*.

Esse *dataset* usa o mesmo tipo de imagem da figura imediatamente anterior. A maior diferença é o arquivo de *waypoints* utilizado, neste caso foi usado o próprio *run-fast4.pickle* disponibilizado pela Skoods. O interesse sobre esse arquivo vem do fato que a trajetória realizada pelo carro é relativamente instável, mas ao mesmo tempo mais rápida que os *datasets* anteriores.

3.2 Exploração e análise dos dados

No que tange o arquivo de log presente em cada *dataset*, será feita uma breve exploração e análise dos dados relacionados ao *steering angle*.

Logo abaixo, temos um *plot* da distribuição dos 1000 primeiros *steering angles* de cada arquivo log:

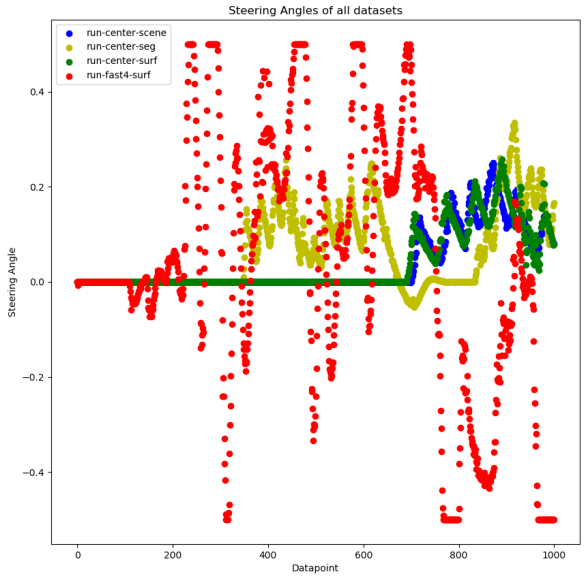

Figure 5: Distribuição dos 1000 primeiros *steering angles*.

É evidente que o *dataset*, *run-fast4-surf/*, apresenta mais oscilações no valor do *steering angle* ao longo do tempo. Isso deve se ao fato do carro se mover muito mais rapidamente, elevando-se assim um certo grau de instabilidade.

Ao passo que os *datasets* restantes apresentam oscilações menos intensas, e um controle mais moderado da velocidade para executar o trajeto da pista.

A partir disso, é esperado que o *dataset*, *run-fast4-surf/*, tenha um aprendizado mais difícil durante o treinamento. E por isso, o modelo neural gerado por ele apresentará alta variabilidade e sensibilidade nas predições realizadas. Causando assim, um desempenho relativamente inferior em relação aos outros *datasets*, os quais possuem controle maior sobre o *steering angle*.

3.3 Arquitetura da rede neural convolucional

Foi baseada na arquitetura proposta por um grupo de pesquisa da NVIDIA no paper: *End to End Learning for Self-Driving Cars*. É um paper extremamente completo e sintético que aborda a construção de uma CNN (*Convolutional Neural Network*) que prediz *steering commands* a partir de uma imagem.

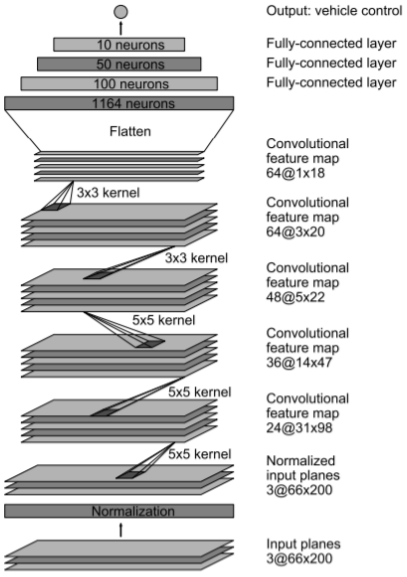

Figure 4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.

Figure 6: Rede neural convolucional proposta pelo grupo de pesquisa da NVIDIA [1].

3.4 Treinamento da rede neural

Antes de ser realizado o treinamento da rede neural, as imagens dos *datasets* serão recortadas para obter uma região de interesse, a fim de diminuir o número de parâmetros da camada de entrada da rede neural. E consequentemente tornando o treinamento mais rápido.

Além disso, a região de interesse vai envolver apenas o que é mais importante na imagem para predição do *steering angle*, diminuindo assim o erro do modelo na observação de outras coisas no ambiente.

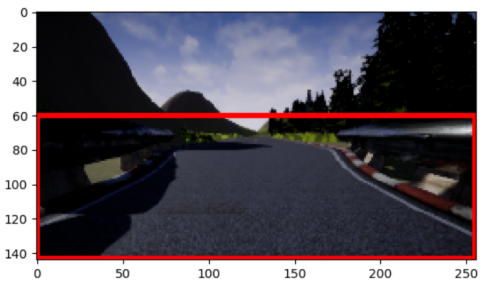

Figure 7: Demonstração da região de interesse.

Basicamente no início do script, *train_nvidia_model.py*, cada imagem após a leitura vai ser cortada para obter essa região interesse, a qual é um retângulo com o conjunto de pontos: (0,60); (0,143); (255, 60) e (255, 143). Vale ressaltar que esse recorte apenas está no escopo do script, a imagem original não vai ser realmente recortada.

Após a leitura das imagens e do arquivo de log do *dataset*, chegou o momento de treinar a rede neural, basicamente envolve o restante do código do script, *train_nvidia_model.py*. Considerando que o treinamento rodou em uma máquina com um processador, ***Intel(R) Core(TM) i7-7700HQ @ 2.8GHz***, e em uma placa de vídeo, ***NVIDIA GeForce GTX 1060 6GB GDDR5***. O tempo de treinamento foi de **aproximadamente 45 minutos para cada *dataset***.

Além disso, algumas *callbacks* de supervisão de treinamento foram adicionadas, a fim de ajudar o *debug* e a busca por um ótimo modelo neural. Os modelos neurais resultantes estão nos seguintes diretórios:

- *nvidia_model-run-center-scene/models/*
- *nvidia_model-run-center-seg/models/*
- *nvidia_model-run-center-surf/models/*
- *nvidia_model-run-fast4-surf/models/*

3.5 Integrações com o sistema da Skoods

Basicamente foi realizada a implementação de uma classe em *Python*, *AutoCar* (*autocar.py*), para o controle do carro autônomo no código principal da Skoods. Essa classe tem semelhanças com a da própria Skoods para o modo de controle PID, chamada *pid_car*.

Além disso, foi alterada a interface do console feita pelo código principal da Skoods nos arquivos *race.py* e *__main__.py*, a fim de facilitar a seleção de modos de qualificação e gravação do *dataset*.

3.6 Código de controle do carro autônomo com a rede neural

```
def race(self):
    ### === Atualização sobre a pose do carro (race.py) === ###
    self.updateState()
    ### === Predição === ###
    img = self.get_image(airsim.ImageType.SurfaceNormals)
    model_output = self.model.predict(img)
    self.controls.steering = float(model_output[0][0])*2
    if (self.state.speed <= 20):
        self.controls.throttle = 0.7 - (0.2*abs(self.controls.steering))
    else:
        self.controls.throttle = 0.5
    self.client.setCarControls(self.controls, self.name)
    return True
```

Figure 8: Algoritmo de controle do carro autônomo (Arquivo autocar.py).

O algoritmo de controle foi realizado de forma extremamente intuitiva, para o *steering* do carro temos o *output* do modelo neural multiplicado por um certo fator de suavização n, sendo $n > 0$. Isso deve se ao fato de que regulando esse fator, podemos aderir a velocidades maiores ou menores dependendo da necessidade. É esperado que para fatores maiores que 1 resultem em curvas mais acentuadas e fatores menores que 1 em curvas mais abertas.

No que tange o *throttle* do carro, foi modelada uma lógica linear que tem o *steering* como parâmetro. É esperado que com essa lógica, o *throttle* tenha um caráter mais entrosado com o *steering* do momento, além de suavizar a velocidade.

4 RESULTADOS

Neste momento será analisado os resultados obtidos a partir de um teste realizado com cada modelo neural produzido. O teste para cada modelo vai englobar **5 voltas consecutivas** no ambiente de simulação da Skoods.

O parâmetro de interesse do teste será **o tempo médio de realização de uma volta completa**, caso não complete todas as 5 voltas o seu valor vai ser dado como 'Incompleto'.

Datasets	Lap 1 (s)	Lap 2 (s)	Lap 3 (s)	Lap 4 (s)	Lap 5 (s)	Tempo médio (s)
run-center-scene/	X	X	X	X	X	'Incompleto'
run-center-seg/	74,1	67,9	67,2	67,2	67,1	68,7
run-center-surf/	75,2	67,6	X	71	68	70,45
run-fast4-surf/	X	X	X	X	X	'Incompleto'

Figure 9: Tabela com todos os tempos médios para cada *dataset*.

4.1 Análise dos resultados

Para o *dataset*, *run-center-scene/*, as voltas tiveram algumas inconsistências principalmente no que tange algumas curvas do trajeto. Entretanto, foram as sombras nas imagens que dificultaram muito uma predição mais correta do modelo. Por essas razões que o carro se perdeu durante o caminho e não conseguiu completar nenhuma das voltas.

Para o *run-fast4-surf/*, o trajeto realizado apresentava muitas oscilações, houve muita dificuldade em se controlar o *steering angle* com a velocidade apropriada. Por causa disso que o carro se perdeu em vários momentos e teve um desempenho inferior em relação aos outros *datasets*, como era esperado.

Os melhores resultados foram com os *datasets*, *run-center-seg/* e *run-center-surf/*. Esse primeiro teve um controle muito bom do *steering angle*, além de ser o único a completar todas as voltas em sequência com o melhor tempo médio. Ao passo que o *run-center-surf/* teve um perfil um pouco mais oscilatório em relação ao *run-center-seg/*, entretanto seu desempenho não deixa a desejar, pois conseguiu completar 4 voltas de 5.

Portanto, esses resultados evidenciam que no campo de visão computacional, as imagens do tipo *airsim.ImageType.Segmentation* e *airsim.ImageType.SurfaceNormals* tiveram um desempenho superior do que o tipo de imagem comum da câmera, *airsim.ImageType.Scene*. Isso deve se ao fato de que se ignora as sombras presentes no tipo de imagem comum da câmera, e o tratamento de cores é mais exaltado para os objetos do ambiente. Confiando assim uma predição melhor e mais estável.

5 CONCLUSÃO

Atualmente, técnicas de controle de carro autônomo são estudadas e desenvolvidas, principalmente aliadas a tecnologias, como LIDAR e GPS. A consistência de um veículo autônomo depende de vários fatores do ambiente, neste presente trabalho o desafio era a realização de um trajeto simples de corrida com poucas aleatoriedades no ambiente.

Apesar de se utilizar *datasets* com tamanhos extremamente pequenos, a rede neural foi capaz de performar de forma bem interessante o trajeto na pista. Como foi o caso dos *datasets*, *run-center-seg/* e *run-center-surf/*, os quais tiveram um bom grau de estabilidade tanto no *steering angle* como no controle de velocidade (*throttle*).

A principal dificuldade foi adaptar um velocidade mais intensa ao carro, pois era complicado sincronizar de forma coerente a velocidade do carro na simulação com o tempo gasto de predição do modelo neural. Como foi o caso do *dataset*, *run-fast4-surf/*.

Como proposta, o controle de carro autônomo presente nesse trabalho pode ser aperfeiçoado tanto na arquitetura da rede neural quanto na construção de um *dataset* mais equilibrado e mais abundante. Além disso, pode se testar diferentes estratégias de suavização do controle do carro como foi mostrado no código de predição.

Para finalizar, o projeto open source, *Microsoft AirSim*, possui diversas outras tecnologias que podem ser adicionadas ao carro autônomo para aperfeiçoar o seu desempenho, como sensores LIDAR e GPS. Além de outros serviços e APIs que facilitam o desenvolvimento de uma aplicação de veículos autônomos como um todo.

REFERENCES

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. arXiv:1604.07316 [cs.CV]