Gustavo Moitinho Trindade

PROPOSTA DE ALGORITMO PARA DETECÇÃO E RASTREAMENTO DE PISTA USANDO REDES NEURAIS E OPENCV

Vitória da Conquista

29 de março de 2021

Gustavo Moitinho Trindade

PROPOSTA DE ALGORITMO PARA DETECÇÃO E RASTREAMENTO DE PISTA USANDO REDES NEURAIS E OPENCV

Trabalho de Conclusão de Curso apresentado como requisito parcial para o cumprimento da disciplina Trabalho Supervisionado II.

Orientador: Roque Mendes Prado Trindade

Universidade Estadual do Sudoeste da Bahia

Departamento de Ciências Exatas

Ciência da Computação

Vitória da Conquista

29 de março de 2021

Resumo

Sistemas de Assistência Avançada ao Motorista — ADAS estão ficando cada vez mais populares. Para que os carros consigam assistir ao motorista de forma eficaz, é necessário que ele tenha diversos sistemas para detecção, rastreamento e modelagem de pista. Em casos onde a sinalização está presente, é importante que o veículo a detecte de forma consistente e use-a para se posicionar na pista. Para isso, foi proposto um algoritmo que usa redes neurais totalmente convolucionais junto com estratégias de análise de imagem com o OpenCV para gerar um modelo de pista baseado na sinalização de linhas da pista.

Palavras-chaves: ADAS, visão computacional, redes neurais, direção autônoma, análise de imagens.

Lista de ilustrações

Lista de abreviaturas e siglas

GPS — *Global Positioning System*

ADAS — *Avanced Driver Assistance System*

CHEVP — *Canny/Hough Estimation of Vanishing Points*

Sumário

1 Introdução

1.1 Contextualização do problema

Com os avanços tecnológicos, é crescente o processo de automação em diversos setores dos sistemas produtivos. A tendência com o passar dos anos é observar tarefas que antes eram realizadas apenas por humanos, sendo também realizadas por máquinas de forma compartilhada ou apenas por sistemas computacionais.

Entretanto, as máquinas enfrentam uma série de problemas para se tornarem úteis em determinados serviços. Assim, em processos mecânicos, as máquinas precisam ser mais eficientes e/ou eficazes em comparação com atividades desenvolvidas por humanos. Caso contrário, há poucas justificativas plausíveis para a automação desses serviços.

Um dos campos da automação em que máquinas tem tido dificuldades é na navegação automotiva (VIVACQUA, 2017). Quando o caminho é predefinido e existem mecanismos precisos de geolocalização, o problema é, em geral, simples. Mas, quando essas condições não se verificam, o problema é muito complexo. Muita pesquisa ainda é necessária, principalmente na área de visão computacional, até que essa tecnologia chegue à um nível aceitável de segurança e desempenho.

No decorrer deste trabalho, o termo veículo autônomo, exceto quando explicitamente for dito o contrário, refere-se apenas a veículos terrestres, principalmente carros comuns.

1.2 Motivação

Nas pesquisas recentes na área da computação, a visão computacional tem sido objeto de estudo em muitos trabalhos como Jun Wang et al. (2019), Zou et al. (2019), Wang, Teoh e Shen (2004), e Wang, Shen e Teoh (1998). O interesse na área de visão computacional é devido a grande quantidade de informação disponível por meio da luz. Câmeras podem obter muita informação sobre o ambiente, e são um sensor muito abundante.

Usando apenas visão computacional, é possível modelar o caminho usando apenas algoritmos clássicos, como o algoritmo B-Snake com o Canny/Hough Estimation of Vanishing Points — CHEVP, com taxa de sucesso de mais de 95%. O algoritmo também é eficiente, com uma taxa de 2 frames/s em 2003 (WANG; TEOH; SHEN, 2004), onde o hardware era muito pior. Espera-se que, com os avanços do hardware, seja eventualmente possível processar imagens com maior nível de detalhamento em menos tempo. Alcançando assim resultados melhores, que podem ser aplicados em ambientes reais.

É claro, em ambientes reais, apenas 95% de acurácia ainda é muito pouco, é um tema recorrente na pesquisa em veículos autônomos de que esses últimos 5% são muito mais difíceis de alcançar. É necessário, para a segurança da população em geral, que os veículos autônomos sejam pelo menos tão confiáveis quanto os seres humanos, preferencialmente até melhores.

Caso esses veículos sejam desenvolvidos, a população geral veria vários benefícios. Não apenas na potencial redução de acidentes (VIVACQUA, 2017), quanto na possibilidade de aumentar a independência de pessoas que não podem dirigir, tanto deficientes motores quanto sensoriais, deficientes mentais e até mesmo idosos. A locomoção é considerada por muitas empresas e pessoas em geral uma capacidade ubíqua, mas muitas pessoas não tem essa capacidade. A inteligência artificial pode servir para melhorar a qualidade de vida dessas pessoas.

Aplicações como taxis autônomos podem oferecer uma outra opção para as pessoas que precisam de transporte. Um taxi autônomo seria mais barato por não necessitar de um motorista — que cobra uma taxa, por seu trabalho — e pode trabalhar $24h$ por dia, evitando deixar o veículo parado. Pode até mesmo ser possível até os próprios donos de veículos autônomos simplesmente possam ativar um modo que faça o carro trabalhar como taxi autônomo. Isso enquanto o motorista trabalha, dorme, ou simplesmente não precisa dirigir por algumas horas. Isso poderia diminuir o número necessário de veículos em uma cidade, fazendo um uso mais eficiente dos recursos.

Grandes empresas como a NVIDIA e AMD estão investindo em hardware que não só seja eficiente em sintetizar gráficos, mas também analisar imagens digitais. Consórcios como o Khronos Group, formados por grandes empresas de hardware e software, recentemente disponibilizaram uma nova API para computação gráfica, chamada Vulkan, que diferentemente do OpenGL, também é capaz de ser usada para o processamento computacional generalizado. Essa tendência da utilização de processadores gráficos para a computação genérica é conhecida como GPGPU (*General Purpose Graphics Processing Unit*), e ela está se tornando muito popular, ao ponto de serem criadas outras APIs específicias para a esse modo de computação. A tecnologia CUDA, por exemplo, é criada com intenção primária o suporte a aplicações de redes neurais. Existe também a opção livre OpenCL, que ocupa esse mesmo nicho. Isso mostra como o interesse dos produtores de hardware está se espalhando: de uma indústria focada em renderização, para uma área mais generalizada, abrangendo áreas como a análise de imagens digitais (NVIDIA, 2011) e processamento de linguagem natural.

Na parte de software, redes totalmente convolucionais estão sendo desenvolvidas, conseguindo excelente resultados em segmentação semântica. A NVIDIA já disponibiliza o código de bibliotecas com redes neurais que podem detectar rua, pedestres e carros em uma imagem (NVIDIA, 2019).

Mesmo com todo esse desenvolvimento, a navegação autônoma é um problema em aberto, e muitos especialistas na área estimam que ainda há muita pesquisa à ser feita para que cheguem à viabilidade. Felizmente, há muito otimismo entre os pesquisadores. Em geral, acredita-se que os veículos autônomos serão desenvolvidos e terão um papel importante no nosso futuro: um futuro com menos acidentes de trânsito, menos poluição e maior democratização do transporte (YURTSEVER et al., 2020).

1.3 Problema

É necessário aumentar a acurácia dos algoritmos de detecção de pista. Também é desejável, por fatores de segurança, que os algoritmos utilizados na navegação autônoma sejam explicáveis. Um algoritmo explicável é aquele que permite um mais fácil desenvolvimento de provas formais de invariantes. Redes neurais, embora tenham se mostrado muito eficazes na análise de imagens, são de difícil introspecção. Isso dificulta a processo de explicar o seu funcionamento (WANG, Jianyu et al., 2017), diminuindo a confiabilidade do sistema.

Atualmente, as estratégias de redes neurais fim-a-fim não são consideradas bons alvos para provas formais. Muito pelo contrário, tendem ser enganáveis por redes neurais adversariais. Em Xu et al. (2019) e Thys, Van Ranst e Goedeme (2019), foi mostrado que é possível criar padrões de imagem, que, se portados por pessoas — estampados em camisetas ou carregados em um cartão — enganam redes neurais, fazendo-as acreditar que a pessoa portadora da imagem, não é uma pessoa, ou seja, tornando-a efetivamente invisível para o sistema. Em nenhum dos casos a estatégia é 100% eficaz, mas isso ainda indica que confiar apenas em uma rede neural fim-a-fim pode impactar negativamente na confiabilidade do sistema.

Em Jianyu Wang et al. (2017), é chamada a atenção ao problema de que muitas redes neurais cometem erros de classificação quando se deparam com elementos não usuais, ou seja, que estavam fora do que lhe fora apresentado na fase de treinamento. Wang apresenta um exemplo interessantíssimo, em que a performance de uma rede neral diminui significativamente quando a imagem de uma guitarra é sobreposta à de um macaco na selva: reconhece o macaco como um humano e a guitarra como um pássaro. No artigo, a explicação para o fenômeno é presumida como sendo o fato de que a rede neural não está acostumada a ver macacos com guitarras, ou guitarras na selva.

Ao mesmo tempo, as estratégias simbólicas tendem a ser explicáveis, mas ao mesmo tempo más generalizadoras. Existe dificuldade do desenvolvimento de um algoritmo de detecção de pista simbólico que não seja influenciado por ruídos comuns como diferenças de sombreamento, ângulo, luminosidade e irregularidade na sinalização.

Por esse motivo, é interessante a combinação das técnicas conexionistas e simbó-

licas, combinando as vantagens de ambas as estratégias.

1.4 Hipótese

É possível combinar o melhor da previsibilidade dos algoritmos clássicos com a flexibilidade e performance das redes neurais ao usar redes neurais totalmente convolucionais para fazer a filtragem da imagem e extração dos píxeis das linhas da pista, e depois usar algoritmos clássicos para criar um modelo da pista com esses píxeis.

1.5 Objetivos

1.5.1 Objetivo geral

Propor um algoritmo para detecção de pista que esteja apto para ser usado no controle de um veículo autônomo.

1.5.2 Objetivos específicos

- Definir o ambiente em que o veículo estará incluso;
- Identificar os pontos problemáticos dos possíveis trajetos do veículo;
- Pesquisar sobre análise de imagens digitais, principalmente com o uso da biblioteca OpenCV;
- Identificar os componentes de hardware necessários para sensores, atuadores e processamento;
- Desenvolver uma maneira eficaz de detectar as marcações da pista;
- Modelar a pista a partir dos dados obtidos das marcações;

1.6 Justificativa

Com o avanço dos agentes inteligentes, tem sido mais comum a implementação de tecnologias Advanced Driver Assistance System – ADAS, em veículos comuns (PERSLOW; CARLSON, 2015). Pesquisas recentes também mostram que essas tecnologias têm potencial para diminuir o número de acidentes (BENSON et al., 2018). Dentro dessa área, novas tecnologias de Simultaneous Localization And Mapping – SLAM, e controle estão sendo desenvolvida desde vários anos atrás (WANG; TEOH; SHEN, 2004; NARANJO et al., 2005), e mesmo assim no momento ainda não existe a aplicação em larga escala de

mecanismos mais sofisticados de automação. O desenvolvimento de pesquisas que investigam tecnologias de visão, planejamento e controle podem, eventualmente, viabilizar a implementação dos veículos autônomos e ADAS em outras situações.

1.7 Metodologia

O objetivo do trabalho é desenvolver um produto tecnológico, neste caso uma implementação de um sistema de identificação e rastreamento de pista. Para realizar esse objetivo, foi feito um estudo bibliográfico das técnicas e algoritmos de visão computacional mais recentes. Também foi feito um estudo do ecosistema de software e ferramentas que estão sendo utlizados para a solução deste tipo de problema.

Os requisitos do sistema são:

- Deve ser robusto o suficiente para identificar a pista em seu ambiente por todo o trajeto esperado;
- Deve ser rápido o suficiente para atingir uma taxa de atualização de pelo menos $10Hz$ em hardware comercial;

Como o sistema espera ser usado em hardware real, veículos físicos precisão ser construídos ou adapatados para rodar o algoritmo. Sem eles, é impossível validar se o método realmente funciona.

Dois ambientes foram propostos para os testes. Um é completamente artificial, para veículos de pequeno porte: feito com madeira pintada de preto e 3 faixas em branco. Duas faixas sólidas com $1m$ de distância entre elas e uma faixa segmentada, central a essas duas. O outro ambiente foi escolhido como sendo as próprias ruas da universidade.

Depois disso, dois veículos foram adaptados para cada caso, e paralelamente ao desenvolvimento dos veículos, os algoritmos foram desenvolvidos, e avaliados. As avaliações podiam ser feitas em simuladores quando os veículos não estavam disponíveis (em construção). Datasets também foram usados para os testes de partes do algoritmo. Os veículos e algoritmos são os artefatos produzidos no trabalho.

Com o desenvolvimento dos artefatos, espera-se obter dados e métodos sobre o desenvolvimento de veículos autônomos.

1.7.1 Desenvolvimento do software

A metodologia de desenvolvimento do software é baseada em prototipação ágil. Como o projeto é de natureza experimental. Muito do código escrito será substituído por código que oferece um melhor desempenho. Uma ferramenta de controle de versão —

especificamente o Git — é usada para que mesmo o código das versões antigas não seja perdido, mantendo um histórico das alterações.

A arquitetura do software pode ser então apenas descrita com poucos detalhes: O software é dividido em duas camadas, um *front-end* e um *back-end*. O *front-end* é basicamente uma rede neural, ou seja, é uma caixa preta. Sabemos apenas que a entrada esperada é uma imagem, e sua saída é outra imagem. A primeira é a visão de uma câmera dentro de um carro, filmando a pista, a segunda é uma matriz que contém o valor 0 para um píxel que não corresponde a uma linha da pista, e contém 1 quando a rede prevê que aquele píxel faz parte de uma marcação de linha da pista.

A segunda camada, o *back-end*. Recebe essa imagem — que é uma matriz — do *front-end* e tenta encontrar dois polinômios que modelam as linhas da esquerda e da direita. Esses polinômios são a saída dessa camada.

1.8 Organização do trabalho

O trabalho foi dividido em cinco capítulos, sendo este o primeiro capítulo, introdutório.

No segundo capítulo é apresentado o breve estado da arte. Nele são descritos as tecnologias e teorias que foram ou estão sendo aplicadas com sucesso na área.

No terceiro capítulo é apresentado o referencial teórico. Nele são descritos as teorias usadas para a criação dos modelos e algoritmos deste trabalho.

No quarto capítulo está descrito o processo pelo qual o projeto foi desenvolvido. Nele são descritos os algoritmos e os testes realizados.

No quinto capítulo são apresentadas as considerações finais.

2 Breve estado da arte

A automação de veículos não é algo novo. Aviões são veículos de passageiros que já tem um auto nível de automação desde 1990 (NORMAN, 1990). Os automóveis, no entanto, não costumam ter quase nenhum nível de controle automático. Costuma-se atribuir como razão para esse caso o fato de que aviões e automóveis, em geral, operam em ambientes muito diferentes, com requisitos diferentes. Aviões operam em ambientes bem abertos, com poucos ou nenhum obstáculo, além de dificilmente voarem por perto de outros veículos — ao menos depois de alçarem voo. Os automóveis, pelo contrário: andam em ambientes abertos ou fechados, comumente cheios de obstáculos. Além disso, é esperado que um automóvel transite a apenas alguns metros de outros veículos regularmente.

Hoje mesmo, carros de diversas montadoras já vêm com certo nível de automação. Esterçamento automático para manter o carro no centro da pista já é algo usado em ambientes reais, geralmente com o requisito de que um humano esteja presente para assumir o controle a qualquer momento. O progresso na automação continua, com empresas tentando ir além. Para isso, foi criado um índice, de 0 a 5, que mede o grau de automação de um veículo. Em Harner (2020), é dada uma explicação sobre os níveis de automação, resumidamente apresentada a seguir:

- Nível 0 — Sem automação.
- Nível 1 — Assistência ao motorista: Neste nível, o carro é capaz de agir sozinho com algumas funções, mas o humano ainda deve ter controle da aceleração, frenagem e prestar atenção nos arredores. Alguns carros atuais tem funcionalidades como frenagem automática, quando detecta que é muito provável de que o carro bata em algo.
- Nível 2 — Automação parcial: Já é comum que empresas implementem esse nível de autômação nos veículos, o carro seria capaz de acelerar e controlar o volante automaticamente em algumas situações, dado que o humano esteja sempre presente para assumir o controle.
- Nível 3 — Automação condicional: Pode-se dizer que é o nível em que a tecnologia está entrando. Neste nível, o veículo é responsável por toda a percepção do ambiente, e tem todo o controle da navegação. O humano ainda deve estar prestando atenção por segurança, mas nas condições ideais, o carro não precisaria de nenhuma invervenção humana.
- Nível 4 — Alta automação: Este nível é parecido com o nível 3. Mas o carro é capaz de dispensar completamente a necessidade de intervenção do motorista em

grande parte das situações. Entretanto, em alguns casos, como mau tempo ou má sinalização, o sistema detecta a perda de confiabilidade e requisitará a atenção de um humano.

- Nível 5 — Automação completa: Todos os níveis anteriores dependiam de que um humano capacitado estivesse ao menos dentro do veículo, para assumir o controle dependendo das condições. No nível 5, isso é desnecessário, pois o veículo é totalmente autônomo, não desempenhando o papel de motorista pior do que um humano capacitado, e, portanto dispensando a necessidade de um. Esse é o objetivo máximo (por enquanto) da pesquisa em veículos autônomos.

Um dos pontos fundamentais para a construção de um veículo capaz de atingir os níveis 2 e adiante, é a capacidade da máquina de perceber o ambiente, e compreendê-lo, de alguma forma. Algoritmos de detecção de pista são uma das maneiras de começar a resolver esse problema. Detecta-se a pista para que o sistema chegue a alguma conclusão sobre o ambiente em que está agindo.

2.1 Arquitetura típica de um veículo autônomo

Em geral, um veículo autônomo tem dois sistemas: Um de percepção e um de decisão. O primeiro coleta dados dos sensores, câmeras, radares, GPSs e/ou sonares. O segundo controla o veículo (aceleração e esterçamento). Os veículos costumam ter uma ampla variedade de sensores diferentes, que serão usados em conjunto para criar a mais precisa estimativa possível de seu estado (localização, velocidade e direção) (BADUE et al., 2020).

O computador geralmente tem acesso a mapas da região em que pretendem navegar, mas não é raro que sejam capazes de gerar mapas por si mesmos, caso estjam em uma região desconhecida. É sempre desejável, no entanto, que o carro seja capaz de fazer edições no mapa caso os seus sensores contradizam esses (BADUE et al., 2020).

2.2 Sensores típicos

2.2.1 Câmera

A câmera é, atualmente, um dos sensores mais comuns. A sua ubiquidade pode levar a uma impressão de simplicidade, mas os dados coletados por ela estão entre os mais complexos entre os sensores populares com os veículos autônomos. Uma imagem contém muito mais informação do que é intuitivamente aparente, e por causa disso a câmera tem um lugar privilegiado entre os sensores no campo da navegação autônoma. Infelizmente, a grande quantidade e complexidade dos dados de uma imagem colocam-a na posição de um

sensor poderoso, mas difícil de se interpretar. A câmera também, infelizmente, é muito afetada por problemas comuns como baixa iluminação (por exemplo, dirigir à noite) e mau tempo (YURTSEVER et al., 2019).

Felizmente, as redes neurais convolucionais estão gerando bons resultados frente ao desafio da interpretação de imagens. O trabalho de Long, Shelhamer e Darrell (2015) mostrou claramente a capacidade das redes neurais convolucionais de classificar objetos em uma imagem pixel a pixel, aumentando muito a popularidade desse tipo de rede neural, as redes neurais totalmente convolucionais. Isso apenas solidifica a condição da câmera como um dos principais sensores nos veículos autônomos.

2.2.2 GPS

O GPS encontra amplo uso entre motoristas humanos, não é surpresa que ele teria um espaço entre os sensores dos veículos autônomos. As suas leituras, principalmente quando comparadas com um mapa (ao menos em áreas em que mapas existam), oferece informações preciosas para os algoritmos de direção.

Um grande problema dos GPS é a sua cobertura, um GPS não oferece garantias de funcionamento em situações como túneis, embaixo de árvores ou até mesmo em ruas cercadas por construções altas (BADUE et al., 2020). Situações como essas nem sempre podem ser evitadas, e um carro pode ter que ficar sem o sinal de GPS por bastante tempo.

Uma unidade de medida inercial (IMU) pode ser combinada com um GPS (GPS-IMU) para localizar um robô. Infelizmente, esse sistema não é considerado bom o suficiente para os requisitos de localização dos veículos autônomos (YURTSEVER et al., 2019).

2.2.3 Radar

Radares são sensores, que podem usar ultrassom ou ondas eletromagnéticas, para obter informação da sua distância para com outros objetos. Os radares são capazes de funcionar bem em situações em que as câmeras têm dificuldades, como no caso de iluminação baixa, e as vezes até com mau tempo. O Lidar — palavra formada pela junção das palavras inglesas *light* e *radar* — é um radar que utiliza lasers, ondas magnéticas com frequência relativamente alta, quando comparadas a ondas de rádio. Os radares à laser se tornaram muito populares em veículos autônomos, já que têm maior acurácia que os radares com ondas de rádio em distâncias mais curtas que $200m$, entretanto estão sujeitos a perda de eficácia em caso de mau tempo, como neblina. (YURTSEVER et al., 2019).

Os radares são sensores ativos, ou seja, dependem da emissão de sinais para poderem obter dados, diferentemente de uma câmera ou GPS, que apenas recebem sinais passivamente. Por esse motivo existe certa preucupação de que dois veículos autônomos

que utilizem radares possam interferir as leituras uns dos outros. (YURTSEVER et al., 2019).

2.3 Detecção de pista

Todos os veículos autônomos de nível 2 ou superior precisam implementar alguma forma de detecção de pista. Existem duas classificações de algoritmos para essa função: baseados em características e baseados em modelo. O primeiro tipo detecta características da pista, e assim consegue informação sobre a sua região na imagem. O segundo, cria um modelo de pista, e considera a pista como uma solução possível para esse modelo (WANG, Jun et al., 2019).

A primeira abordagem tem a desvantagem de depender muito da qualidade da imagem de entrada. Assim, ela é muito sensível a ruído, como sombras ou diferenças de cor no asfalto. Mesmo assim, ainda consegue resultados favoráveis, com o uso de uma série de tratamentos na imagem para reduzir esse problema (WANG, Jun et al., 2019).

A abordagem baseada em modelo é mais resistente a ruído, embora assumam algumas características sobre a forma da pista. Uma abordagem é a de estimar o ponto de desaparecimento no espaço de imagem, da imagem de entrada, uma spline pode ser traçada de modo a se encaixar com o modelo (WANG, Jun et al., 2019). Em 1998, foram utilizadas Catmull-Rom splines para a modelagem da pista. O algoritmo envolvia detecção de bordas com o Canny, depois disso, técnicas probabilísticas eram usadas para filtrar as bordas, usando o modelo da pista. No fim, buscava-se os pontos de controle e desaparecimento que seriam usados para traçar as curvas que representam a pista (WANG; SHEN; TEOH, 1998).

As B-Snake foram utilizadas em um trabalho similar ao anterior. As diferenças eram que a B-Spline era usada para modelar a curva, e que o CHEVP era usado para fazer as estimativas de ponto de desaparecimento (WANG; TEOH; SHEN, 2004).

Redes neurais fim-a-fim foram usadas com sucesso em Bojarski et al. (2016). Uma rede neural fim-a-fim é uma abordagem em que a rede neural é treinada, não para classificar pixeis, mas para controlar diretamente os atuadores do agente. Nesse caso, a rede neural recebe a imagem da pista, e como saída controla o esterçamento e aceleração do veículo.

Um ponto interessantíssimo no trabalho de Bojarski é a sua simplicidade. O computador recebe dados de três câmeras, esquerda, central e direita. E tem como saída apenas o controle de esterçamento do veículo (a rede de Bojarski não controla a aceleração do veículo, nem a frenagem). O método de treinamento é simples, a rede é treinada com pares de entradas da câmera e saídas esperadas de esterçamento, e o algoritmo da

retropropagação é aplicado para ajustar os pesos e viézes. Depois do treinamento, as câmeras da esquerda e direita podem ser dispensadas, e a rede prevê o esterçamento apenas com a imagem da câmera central (BOJARSKI et al., 2016).

2.4 Detecção de obstáculos

É importante lembrar que a detecção de obstáculos não é menos importante do que a detecção da pista. Muitos dos sensores apresentados anteriormente podem ser usados para a detecção de obstáculos.

Obstáculos podem tanto ser deformações na pista, como buracos, ou objetos com tamanho significativo como rochas ou partes de outros veículos e construções. Em caso de acidentes, a pista pode ficar rapidamente cheia de objetos não usuais, como rodas soltas ou um poste caído, para o qual o veículo pode não estar preparado.

Também podem ser considerados obstáculos os outros veículos, já que devem ser evitados quando estão muito perto. Já que estão se movendo, uma análise da sua posição em função do tempo, combinada com modelos heurísticos da navegação de um carro, pode ser usada para a previsão de sua posição na próxima iteração.

É claro que o veículo deve, idealmente, distinguir entre obstáculos vivos e não vivos. É muito mais importante saber detectar e desviar de um humano do que desviar de uma lata de lixo. O atropelamento de pedestres ainda é um dos grandes problemas relacionados aos sistemas de transportes usuais, principalmente em países em desenvolvimento como a Índia, e portanto a atenção de um motorista computador deve ser redobrada para garantir a detecção e prevenção desse tipo de acidente (SOLAIMAN, 2020; DAHL, 2004).

2.5 Detecção e interpretação da sinalização

As ruas e carros foram construídos primariamente para o tráfego de veículos controlados por humanos. Não é a toa que toda a sinalização é construída para ser identificada facilmente pelas pessoas. Um computador tem que se adaptar a esses sinais, já que não é esperado que sejam complementados com sinais específicos para máquinas até que essas máquinas que dirigem sozinhas mostrarem que podem realmente funcionar. Esse paradoxo faz com que as máquinas devem ser treinadas e adaptadas para processar sinais aos quais, no ponto de vista da engenharia de veículos autônomos, são pouco eficazes.

Placas de sinalização têm imagens e texto — geralmente números — para indicar ao motorista sobre curvas, lombadas, limites de velocidade, problemas na pista entre outras coisas. Sabemos que máquinas são, em geral, piores que humanos na análise de imagens e texto. O uso de sinais como ondas eletromagnéticas ou até mesmo imagens especializadas como o código de barras ou código QR facilitariam a detecção por máquinas. Alguns

trabalhos usam até mesmo de trilhos magnéticos, por baixo da pista, para uso exclusivo de veículos autômatos.

De todo modo, já que mudanças na infraestrutura de transporte são pouco prováveis por enquanto, os computadores devem ser capazes de detectar sinais criados para humanos, tão bem quanto os humanos.

É claro que deve-se lembrar de ambientes com má sinalização. Um veículo autônomo deve respeitar a sinalização quando presente, mas ainda ser capaz de agir razoavelmente quando esta estiver ausente.

2.5.1 Redes neurais como a YOLO

Redes neurais da arquitetura YOLO — *You Only Look Once* — foram usadas com sucesso para a detecção de pessoas, animais e veículos em imagens, de forma que possibilitam o discernimento da sua posição com uma caixa ao seu redor.

Um dos fatores interessantes da YOLO é que esta é geralmente treinada e testada para a detecção de objetos das mais variadas classes. Ela pode ser treinada em imagens completas, ou seja, imagens em que o objeto a ser detectado não está isolado no quadro. Essa característica é particularmente útil para veículos autônomos (DU, 2018).

2.6 Veículos já construídos

O *DARPA Grand Challenge*, foi um famoso desafio criado pela Agência de Projetos de Pesquisa Avançados de Defesa (*DARPA, em inglês*), em 2003 para incentivar o desenvolvimento dos veículos autômatos. O desafio envolvia a criação de um veículo capaz de navegar por um terreno desconhecido, de até 282 quilômetros, de maneira autônoma, em até 10h (THRUN et al., 2006).

O carro vencedor do desafio foi o do Time de Corrida de Stanford, com o carro Stanley, que pode ser visto na figura 1. Stanley usava uma série de sensores, quase todos montados no teto. Entre esses sensores estavam sensores *Lidar range finders* (radar de laser), uma câmera colorida, dois radares e um GPS (THRUN et al., 2006).

A arquitetura de software de Stanley era descentralizada. Os dados são recibidos e marcados com um *timestamp*. Esses timestamps são então usados para integrar corretamente os dados para o processamento (THRUN et al., 2006). Isso evita problemas comuns de concorrência, e facilita a paralelização das tarefas.

Dentro da pipeline de processamento de Stanley, está a camada de percepção, essa camada é a responsável por traduzir os dados dos sensores em modelos internos. O Stanley usa filtros de Kalman como módulo principal dessa camada. Esse filtro é capaz de estimar o estado do veículo: as suas coordenas, orientação e velocidade (THRUN et al.,

Figura 1 – Foto do Stanley (STANFORD UNIVERSITY, 2005)

2006). Nessa camada de percepção, existe um módulo que encontra o centro da rua usando lasers, essa infomação é dada como uma das entradas para a camada de planejamento e controle (THRUN et al., 2006).

Com as câmeras, Stanley mistura dados obtidos com o laser com dados da classificalão de pixeis para obter a área visível navegável (THRUN et al., 2006). No caso do desafio DARPA, os carros deviam percorrer o caminho numa estrada de chão, não sinalizada (THRUN et al., 2006). Por causa dessas especificações no desafio, não era necessário que o Stanley respeitasse a sinalização de tinta no asfalto, por exemplo.

No Brasil, o Laboratório de Robótica móvel da USP de São Carlos desenvolve um veículo chamado Carina (figura 2), que, em 2015, era o projeto mais avançado do país (LEITE et al., 2015).

Em Vivacqua (2017), o sistema de visão usa uma câmera para detectar duas faixas, a esquerda e a direita, modelando-as em um polinômio de segundo grau. A cada um desses polinômios é atribuido valor correspondente à confiabilidade da informação. Nesse mesmo trabalho, é também feita a filtragem da imagem com o intuito de ênfase nas cores comuns das faixas, usando-se a fórmula $V = \frac{R+G}{2}$, em que V é o novo valor do pixel, e R e G sendo os valores vermelho e verde dos pixeis originais.

Depois da filtragem da imagem, esta é binarizada. Os aglomerados de pixels formados pela binarização serão transformados em polinômios. Se a linha encontrada muda de direção muito bruscamente, esse aglomerado é dividido, detectando-se uma quebra na linha (VIVACQUA, 2017).

Figura 2 – Carina, veículo autônomo desonvolvido na USP. Fonte: Leite et al. (2015)

2.7 Limitações físicas dos veículos

É importante lembrar que os veículos não podem satisfazer quaisquer requisitos arbitrários de movimento. Na prática, isso significa que situações como curvas merecem uma atenção especial. Os veículos terrestres de quatro rodas usuais, como os carros usados no dia a dia, usam um sistema de direção regidos pelo menos em parte pela geometria de Ackermann. Isso deve ser levado em consideração nas curvas, um algoritmo deve preferenciar curvas mais suaves, tomando a decisão de quando girar o volante com antecedência.

O controle de velocidade também tem um grande impacto na capacidade de processamento aparente do veículo. Um veículo mais devagar tem mais tempo para a tomada de decisões, e assim os atrasos causados por software tem um efeito diminuído sobre o desempenho do carro. Pela mesma lógica, andar mais rápido deixa menos tempo para o algoritmo tomar decisões e ressalta os atrasos do software, diminuindo o seu desempenho e aumentando o risco de acidente.

2.8 A biblioteca OpenCV para a análise e tratamento de imagens

A biblioteca OpenCV já implementa uma série de algoritimos para o processamento e análise de imagens digitais. Ela é uma biblioteca de código livre, escrita em C++. (OPENCV, 2019). Também tem interfaces para muitas linguagens de programação, notavelmente o Python, o que simplifica o seu uso (WRAPPER, 2019). O Python pode ser usado para a criação de protótipos de maneira mais ágil, enquanto o C++ é melhor para implementações definitivas. Além disso, o Python provou-se mais simples em relação à instalação e uso de bibliotecas utilitárias.

A linguagem de programação Julia vem crescendo como linguagem de scripting de cunho científico. O intuito dela é ser usada da forma que o Python é usado: scripting simples, com pouco esforço, mas também tenta ser viável no mesmo nicho do Fortran, com cálculos matemáticos de altíssima performance. O seu maior problema é que não é ainda tão popular e, como consequência, acessível quanto o Python. Júlia não tem tantos materiais de ensino quanto linguagens mais estabelecidas como o Python.

Com as bibliotecas NumPy e SciPy (que curiosamente foram construídas a partir do grande legado de código Fortran), o Python pode ser usado para cálculos científicos com um desempenho maior do que o esperado do seu interpretador padrão. Além disso a sua sintaxe é muito simples e legível. Quando adicionados a isso todo o ambiente de aprendizado de máquina que teve o Python como alvo principal para a exposição de suas APIs, é fácil de ver o motivo do Python ter se tornado o padrão de fato de grande parte das pesquisas com redes neurais.

2.9 Torch e TensorFlow

O *torch* — mais conhecido por sua interface *PyTorch* — e o *tensorflow* são duas bibliotecas usadas para a simples construção de redes neurais. Ambas as ferramentas são amplamente utilizadas em projetos que envolvam aprendizagem de máquina e análise de imagens digitais. Neste trabalho, o PyTorch foi escolhido por ser o mais comum nos projetos acadêmicos que foram tomados como base para o desenvolvimento.

2.10 O simulador CARLA

O CARLA é um simulador de direção automotiva. Com ele, é possível testar algoritmos em um veículo simulado. Também é possível usá-lo para o obtenção de dados de teste em situações sob o total controle do usuário. Ele foi usado nas fases iniciais do projeto por ser simples para fazer testes. Eventualmente, os dados obtidos pelo CARLA foram substituídos pelos dados obtidos de vídeos reais.

3 Referencial teórico

3.1 Detecção de bordas com o algoritmo de Canny

Entre os algoritimos clássicos usados para a análise de imagens digitais estão o Canny e a detecção de linhas de Hough. O Canny é um algoritimo usado para detectar bordas em uma imagem. Ele funciona com o uso da convolução. Nela, uma matriz chamada kernel é criada, com a qual, para cada pixel da imagem, os seus arredores são multiplicados pela casa sobrejacente da matriz, e os resultados somados para encontrar o novo valor do pixel (CANNY, 1986).

O Canny usa a convoluções e gradiente para encontrar as bordas. Para cada pixel da imagem de entrada, o algoritimo processa-o junto com os valores ao redor e retorna 1 caso acredite que o pixel esteja numa borda, e 0 caso contrário (CANNY, 1986). O processamento do Canny pode ser visto na figura 3.1.

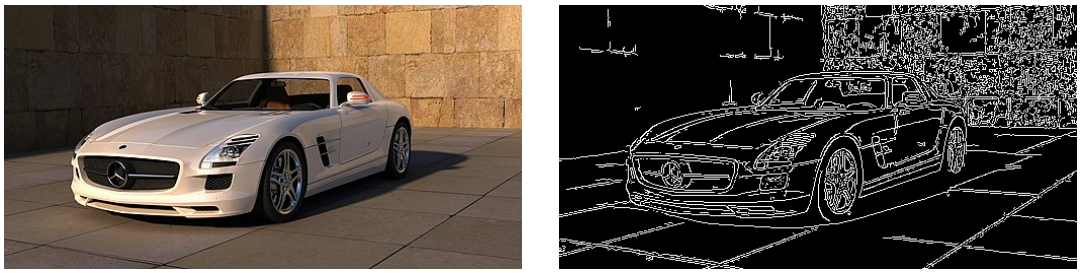


Figura 3 – Detecção de linhas do Canny

3.2 Detecção de linhas com a transformação de Hough

A transformação de Hough é uma técnica para encontrar linhas em uma imagem binarizada. Essa técnica envolve transformar cada pixel branco em uma reta, seguindo um padrão particular. Quando isso é feito, quando muitas dessas retas tem uma interseção em comum, ou perto disso, é dito que ela tem muitos votos, e portando uma grande chance de existir uma linha passando por aqueles pontos (FISHER et al., 2004).

A combinação do Canny, que binariza a imagem nas bordas, e a transformação de Hough, que encontra linhas em imagens binarizadas é simples: a saída do Canny pode ser colocada como entrada diretamente para o algoritmo de linhas de Hough.

Tanto o Canny quanto a transformação de Hough são técnicas a serem aplicadas em uma única imagem. Mas algumas técnicas utilizam visão stereo para poder obter mais informação do ambiente por meio da visão.

Figura 4 – A imagem mais a esquerda é a imagem original. Ao centro, é possível ver a mesma imagem, mas em escala de cinza. A imagem à direita está em escala de cinza, mas com um filtro que torna regiões amarelas mais claras, sem diminuir a intensidade da cor branca

Um dos problemas com a detecção de linhas com Hough, é que o processo de detecção de curvas é pouco intuitivo e de baixa performance. Ele é assim mais atrativo para a detecção de linhas retas, o que nem sempre é o suficiente.

3.3 Pré-processamento da imagem de entrada

Alguns ajustes podem ser feitos no processo, como por exemplo, a filtragem da imagem de entrada, para diminuir o ruído. Muitas vezes, a imagem de entrada pode ser convertida para a escala de cinza, o que diminui consideravelmente o custo de processamento sem acarretar em significante diminuição na eficácia (geralmente). Caso a linha a ser detectada dependa muito da cor, como encontrar uma linha vermelha em um fundo azul, usar escala de cinza pode ter resultados muito inferiores.

Nos casos em que a cor é importante, mas haja informação prévia sobre os tipos de cores que estarão formando os contrastes importantes, é possível obter um bom meio termo ao considerar uma função customizada que pega os valores dos canais R, G e B e converta-as em um único canal C, que pode dar ênfase em uma das cores, por exemplo. Numa pista, as marcações tendem a ser brancas ou amarelas, portanto pode-se criar uma função que converta a imagem para a escala de cinza, mas dê mais intensidade as regiões amarelas. Essa abordagem é utilizada em Vivacqua (2017), e pode ser vista na figura 4.

Filtros como o filtro Gaussiano podem ser usados para diminuir o ruido, enquanto mantém as características mais importantes da imagem, como o contraste de cor e intensidade nas bordas entre objetos. Ele é comumente utilizado antes de operações como a detecção de linhas e bordas.

O tamanho da imagem de entrada também tende a ser mantido constante pelo

algoritmo, para evitar problemas imprevistos. Por esse motivo grande parte das implementações começa por reduzir o tamanho da imagem de entrada para uma imagem de baixa resolução, com um tamanho fixado no código. Geralmente a resolução das câmeras atuais é muito alta, mais do que é suficiente para uma boa detecção das características do ambiente. Usar imagens muito grandes pode simplesmente aumentar a carga de processamento do algoritmo, aumentando o tempo de processamento de cada quadro.

Em um sistema como um veículo autônomo, a latência do processamento dos quadros deve ser baixa, já que o sistema é de tempo real, e não pode ficar esperando por dados para tomar uma decisão. Um atraso pode ser questão de vida ou morte. É claro, que resoluções mais altas podem ser usadas caso o poder de processamento aumente, mas como o processamento é limitado, é sempre interessante procurar melhores resultados com menos recursos.

3.4 Sistema Global de Posicionamento para localização

O Sistema Global de Posicionamento — GPS, tem uma grande utilidade para várias áreas de navegação autônoma. Estratégias com GPS são utilizadas com fusão de sensores em veículos autônomos terrestres (YURTSEVER et al., 2019). Junto a um mapa, ele pode ser usado para obter a localização do veículo. O problema do GPS é que existe uma margem de erro relativamente alta, e que aumenta em alguns ambientes, principalmente em ambientes urbanos, onde prédios altos diminuem a sua precisão, e túneis podem incapacitar o sistema completamente (YURTSEVER et al., 2019).

3.5 Fusão de sensores em veículos autônomos

Um veículo autônomo, quando inserido no contexto onde os humanos trafegam, pode causar acidentes. Sendo assim sistemas críticos, o uso de apenas um sensor, que pode falhar, não é recomendado. Estratégias que usam fusão de sensores conseguem ser mais confiáveis (ZHU, B. et al., 2019). Desse modo, todo o tipo de abordagem sensorial é desejável, pois a combinação poderá melhorar os resultados do sistema em situações não favoráveis.

Entre as outras abordagens estão o uso de sensores laser Lidar, que tem as desvantagens de serem um pouco mais caros (BERRIEL, 2016; VIVACQUA, 2017) e modificarem a aparência externa do veículo. As câmeras comuns, RGB, também são muito interessantes, pois são capazes de coletar uma grande quantidade de informação do ambiente, sob o custo de necessitar de mais poder de processamento (VIVACQUA, 2017).

Também existem trabalhos em que a estratégia utilizada envolvia *landmarks*, que eram elementos pré-determinados que ao serem detectados pelos sensores, seriam âncoras

para calcular a posição do veículo (VIVACQUA, 2017).

3.6 Redes Neurais Artificiais

Uma Rede Neural Artificial — RNA — é um modelo computacional inspirado no cérebro humano. Uma das suas caracteristicas mais importantes é a sua capacidade de aprender a reconhecer padrões em dados. O fato de que ela pode aprender algoritmicamente é essencial para a sua popularidade, pois problemas de reconhecimento de padrões são notoriamente difíceis de serem resolvidos por algoritimos manuais.

O modelo computacional consiste de um grafo dirigido ponderado. Um neurônio é um nó, contendo uma função de ativação, e está conectado aos outros neurônios por arestas, que contêm pesos.

O *perceptron* foi um modelo proposto por McCulloch-Pitts. Ele é caracterizado por não ter camadas escondidas, apenas duas camadas: entrada e saída. O perceptron de McCulloch-Pitts foi muito criticado por não ser capaz de resolver problemas, quando o espaço de dados não era linearmente separável. Mesmo funções simples como o *XOR* não podiam ser aprendidas.

Um neurônio do perceptron pode ser visto na figura 5.

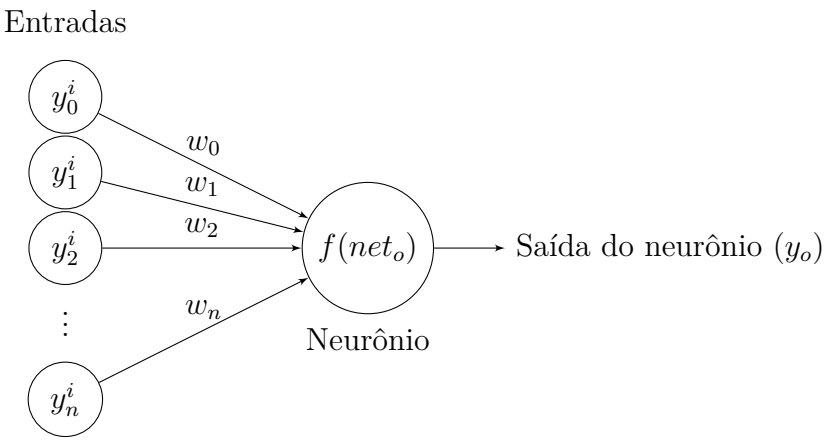


Figura 5 – Modelo de um neurônio simples de um perceptron

Assim, o neurônio pode ser representado matematicamente como:

$$net_o = \sum_{n=0}^{N_i} y_n^i w_n + b_o$$

$$y_o = f(net_o)$$

y_o é a saída do neurônio o. Considera-se os neurônios que fornecem a entrada para o neurônio o os neurônios de entrada i. A soma de todas as saídas y_n^i multiplicadas pelos

pesos w_n é denominada net_o. A saída de um neurônio o é y_o. b_o é chamado de viés do neurônio o (STAUDEMEYER; MORRIS, 2019).

A função f é geralmente uma função em $\mathbb{R} \mapsto (0, -1)$, como por exemplo a função sigmoide:

$$f(x) = \frac{1}{1 - e^{-x}}$$

Ou frequentemente, a função *ReLU — Rectified Linear Unit*:

$$f(x) = \begin{cases} 0 & \text{se } x < 0 \\ x & \text{caso contrário} \end{cases}$$

A ReLU não é derivável em zero, mas isso não se mostra um grande problema: a derivada em zero pode ser arbitrariamente escolhida como 0 ou 1, sem maiores problemas.

3.6.1 Perceptron de Múltiplas Camadas — MLP

O perceptron pode ser organizado recursivamente em múltiplas camadas. Existem duas camadas obrigatórias, a de entrada e a de saída, e opcionalmente uma ou mais camadas escondidas, que ficam entre as de entrada e saída. A camada de entrada é especial, pois não precisa de uma função de ativação: considera-se que o valor de ativação de um neurônio da camada de entrada seja o próprio valor de entrada associado a este neurônio.

No MLP, todo neurônio de uma camada — com exceção da camada de entrada — tem uma conexão com cada um dos neurônios da camada anterior (STAUDEMEYER; MORRIS, 2019). Um exemplo de MLP com três camadas pode ser visto na figura 6:

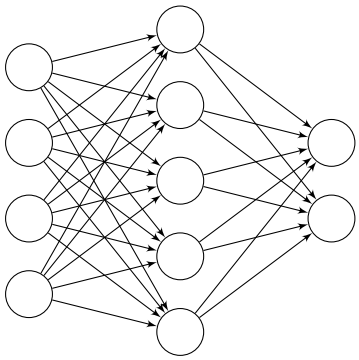


Figura 6 – Um exemplo de MLP com uma camada escondida

O MLP foi muito importante, pois ele é capaz de resolver o maior problema do perceptron. Um perceptron de múltiplas camadas é capaz de classificar corretamente dados não linearmente separáveis.

3.6.2 Aprendizado

Uma MLP pode aprender de três maneiras: aprendizado supervisionado, não supervisionado, e por reforço. Comumente, o algoritmo da retropropagação é usado para o treinamento de uma rede neural de modo supervisionado.

O aprendizado supervisionado consiste ensina a rede com exemplos. Esses exemplos vêm de uma grande base de dados, consistindo de pares entrada e saída esperada da rede. A rede é inicializada aleatóriamente, e subsequentemente, são apresentadas a rede uma entrada da base de dados. A saída da rede é então comparada com a saída esperada daquela entrada, e essa comparação é o que dá origem a taxa de erro da rede (STAUDEMEYER; MORRIS, 2019).

O algoritmo da retropropagação manipula os pesos da rede neural de maneira a diminuir essa taxa de erro. Isso é feito com o uso do gradiente.

Suponha a função E_o, que represente a taxa de erro de um neurônio o da camada de saída. Esse erro é obtido ao comparar as saídas desse neurônio com as suas saídas esperadas, sendo maior quanto maior a disparidade entre a saída real e a saída esperada. Para diminuir esse erro, pode-se manipular os pesos $W_{[h,o]}$, que representam os pesos das arestas que vem de um neurônio h até o. Os neurônios h são da camada anterior à de o (STAUDEMEYER; MORRIS, 2019).

$$\Delta W_{[h,o]} = -\eta \frac{\partial E_o}{\partial W_{[h,o]}}$$

Em que η é a taxa de aprendizado da rede.

Como uma MLP pode ter multiplas camadas escondidas, é necessário fazer esse processo recursivamente por todas elas. Para isso, pode-se calcular também:

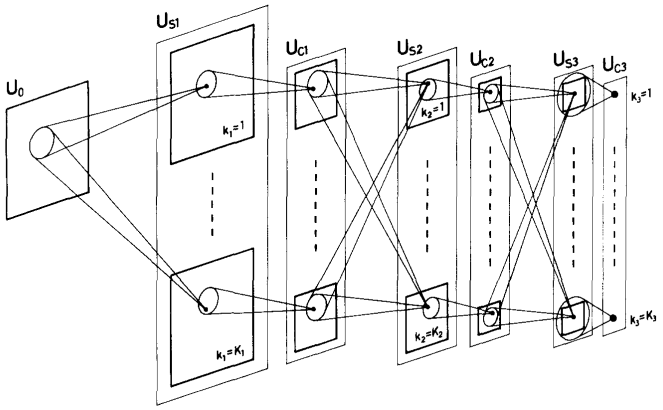
$$\gamma_h = \sum_{n \in O} \frac{\partial E_n}{\partial y_h}$$

Em que O é o conjunto dos neurônios da camada de saída e y_h é a saída de um neurônio h.

γ_h terá então a mesma função de E_o, mas agora será usado para propagar o erro para os neurônios da camada anterior à do neurônio h. Nesse caso, os pesos atualizados serão $W_{[j,h]}$.

Após todos os pesos da rede serem atualizados, confere-se novamente o erro: enquanto este for maior do que desejado, outra iteração do algoritmo é aplicada, com a esperança de que o erro se tornará menor a cada iteração.

Figura 7 – Modelo convolucional do Neocognitron, baseado nas células do córtex visual (FUKUSHIMA; MIYAKE, 1982)



3.7 Redes Neurais Convolucionais — CNN

As redes neurais convolucionais baseiam-se no MLP, mas tomam decisões para aumentar a eficiência computacional dos processos de treinamento e predição. O conceito chave da CNN é que pesos podem ser compartilhados por neurônios. Esse compartilhamento é possibilidado pelos filtros que são tão característicos da convolução (FUKUSHIMA; MIYAKE, 1982).

3.7.1 Neocognitron

Um dos primeiros trabalhos que podem ser considerados como redes neurais convolucionais for o neocognitron. O objetivo dessa rede era a detecção de caracteres escritos à mão — um caso de classificação de imagem. O neocognitron foi inspirado em estudos sobre o cortex visual humano, e provou-se eficaz na tarefa de análise de imagens (FUKUSHIMA; MIYAKE, 1982).

Na figura 7, pode-se ver as camadas separadas em U_0, U_C e U_S. U_0 é a camada de entrada, não sendo muito diferente da camada de entrada do MLP, com exceção de que é organizada como uma grade, se importando com a posição dos nós (a posição dos pixels é importante numa imagem). As camadas U_S contém as células S, essas são as únicas cujas sinapses são plásticas — ou seja, são as únicas cujos parâmetros são atualizados durante o treinamento. A camada U_C contém as células C, com sinápses fixas (FUKUSHIMA; MIYAKE, 1982).

A rede aprende sem supervisionamento, adaptando os seus parâmetros até que apenas uma célula C da última camada reaja a algum conjunto de entrada U_0. Quando a rede chega a esse estado, significa que ela classificou a entrada U_0 ao grupo reconhecido por C (FUKUSHIMA; MIYAKE, 1982).

Uma das mais importantes vantagens do neocognitron é a sua capacidade de econ-

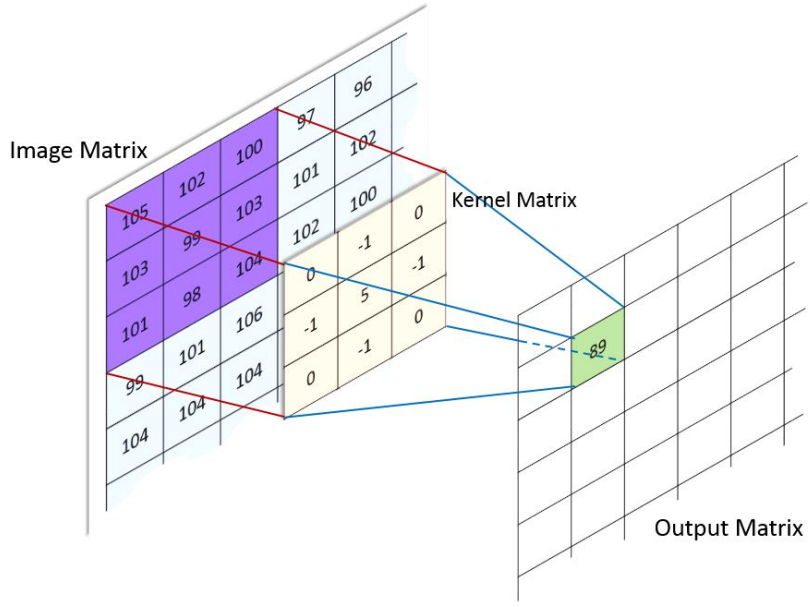
trar os objetos independentemente da sua posição na imagem. Ele também é capaz de ignorar pequenas distorções na imagem (FUKUSHIMA; MIYAKE, 1982).

3.7.2 Modelo usual de uma CNN

Atualmente, é muito popular a utilização de modelos baseados no neocognitron para o reconhecimento de imagens. As ideias principais são mantidas, mas muitas das construções do neocognitron são atualmente conhecidas por outros nomes. É com os nomes contemporânios que as CNNs serão explicadas as seguir.

Como entrada, a rede recebe uma imagem — que é uma matriz. Inicialmente, a CNN é composta por uma camada convolucional, e uma camada de amostragem, nesta ordem. Essas duas camadas são repetidas quantas vezes o arquiteto da rede achar necessário. Essas camadas recebem matrizes como entrada e geram matrizes como saída, assim podem ser organizadas em série sem problemas. Após essas camadas convolucionais, as matrizes resultantes são planificadas e os valores são usados como entrada em uma MLP convencional.

Figura 8 – Convolução em uma CNN



Como pode ser visto na figura 8, a convolução já diminui a complexidade espacial da entrada. A camada de amostragem serve para diminuir ainda mais essa complexidade, embora não demasiadamente, ou muita informação seria perdida. A vantagem da diminuição da complexidade é que a rede terá menos pesos para ajustar — o que torna o treinamento e predição mais rápidos. A diminuição do número de pesos (ou parâmetros) também diminui a chance de sobreajuste, que é o que acontece quando a rede se ajusta tão bem aos casos de treinamento que perde a capacidade de generalização para novos casos.

Um estudo feito por Zhang et al. (2018) mostra que uma rede neural, quando tem parâmetros o suficiente, pode "decorar" os casos de teste, mantendo assim uma baixíssima taxa de erro de treinamento, enquanto mostra-se incapaz de prever corretamente os resultados de situações novas.

Um outro estudo, mostra que mesmo sem a presença de overfitting, uma rede neural pode ser capaz de memorizar casos muito específicos (CARLINI et al., 2019). No estudo, o foco está em como essa memorização pode ser explorada maliciosamente. Não fica claro, com o estudo, se tal vulnerabilidade se aplica à visão computacional com CNNs. De todo modo, como um sistema para direção autônoma é crítico, é necessário que tal modo de exploração seja ineficaz.

3.8 Redes Neurais Recursivas

Uma rede neural recursiva (*Recursive Neural Networks, RNN*) baseia-se nos modelos anteriores, mas diferentemente destes, é um sistema dinâmico: uma RNN tem um estado interno, que muda com o tempo. Para que isso seja possível, essa rede permite que existam conexões circulares, que podem ir dos neurônios das camadas mais profundas da rede até as camadas anteriores. Isso permite que a rede tenha memória de curto prazo (STAUDEMEYER; MORRIS, 2019).

O treinamento dessas redes não é necessáriamente muito diferente do treinamento de uma MLP comum. Isso porque, em um intervalo de tempo finito, uma RNN pode ser *desenrolada* em um MLP comum (figura **??**. Depois de desenrolar a rede, os pesos são atualizados como se estivesse-se lidando com o MLP. A retropropagação em uma RNN é conhecida como retropropagação pelo tempo (STAUDEMEYER; MORRIS, 2019).

Redes neurais recursivas são muito usadas no processamento de sinais. É comum que sinais sejam arbitrariamente longos, tornando impossível a tarefa de criar uma rede neural tradicional que receba um sinal completo como entrada. Dividir o sinal em partes finitas e processá-las separadamente não é bom o suficiente, pois a interpretação de um sinal pode mudar dependendo da ordem dessas partes. Uma alternativa, é o uso das RNN. Divide-se o sinal de entrada em partes finitas, e coloca-as na rede na ordem original. Como a RNN tem um estado interno, ela é capaz de se lembrar de características dos sinais anteriores, e levá-las em consideração no tratamento dessas partes. As saídas da RNN são então coletadas em partes, e quanto juntas, fornecem um resultado que representa um ponto de vista holístico sobre o sinal de entrada (STAUDEMEYER; MORRIS, 2019).

3.8.1 LTSM RNN

A LTSM RNN, sigla inglesa para Rede Neural Recursiva de Longa Memória de Curto Prazo (*Long Short Term Memory RNN*), é um tipo específico de RNN que foca em

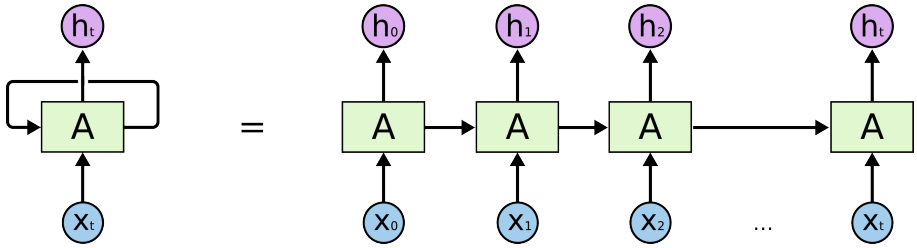


Figura 9 – Desenrolando uma RNN

manter a sua memória por mais tempo do que era usual nas redes recursivas tradicionais. O problema que precisava ser resolvido para possibilitar isso é conhecido como *desaparecimento do erro*. A solução encontrada foi a utilização de construções para manter o fluxo de erro constante entre os neurônios.

A LTSM tem encontrado amplo uso na área de processamento de linguagem natural.

3.9 Redes Neurais Completamente Convolucionais

Uma rede neural completamente convolucional, é um tipo de rede baseada na CNN, aplicada principalmente na segmentação semântica.

A segmentação semântica é uma aplicação bastante útil para a navegação autônoma usando CNN, marcando na imagem as áreas detectadas, e o que representam. Mas existe bastante dificuldade em aplicá-la em tempo real, graças a quantidade de processamento requerido em uma única imagem. Bibliotecas como a da Nvidia para segmentação semântica chegam a recomendar até mesmo o uso de oito GPUs para o processamento (ZHU, Y. et al., 2019; REDA et al., 2018).

Uma rede neural completamente convolucional tem uma série de camadas convolucionais intercaladas por camadas de amostragem, assim como a CNN tradicional. Mas diferentemente desta, seguindo essas camadas iniciais existem outras camadas convolucionais que fazem o *upsampling*, ou seja, tem como saída uma matrizes maiores do que tem de entrada.

3.10 Regressão polinomial

Um processo de regressão linear é um algoritmo usado para criar um método computacional capaz de relacionar elementos entre dois conjuntos, a partir de um conjunto finito de exemplos de relações entre esses elementos. Esse processo assume uma relação funcional $f : A \mapsto B$, em que A e B são conjuntos, potencialmente infinitos. A regressão é usada quando um método geral para computar $f(a)$ para qualquer $a \in A$ é desconhecido.

O algoritmo então irá aproximar — e talvez até descobrir de maneira exata — uma maneira de computar tal relação.

O algoritmo de regressão irá estimar uma função g — desta vez com um método para sua computação — tal que, em uma situação ideal:

$$\forall a \in A; f(a) = b \rightarrow g(a) \approx b$$

Para que essa estimativa seja possível, o algoritmo precisa de informação sobre a relação f. Essa informação é dada em forma de exemplos. Os exemplos são um conjunto de pares $E \subset A \times B$ em que sabe-se que:

$$\forall (a, b) \in E; f(a) = b$$

A função g é calculada de forma que:

$$\forall (a, b) \in E; g(a) \approx f(a)$$

O algorimo de regressão assume, que caso g aproxime f em um subconjunto finito do domínio de f, é provável que ele aproxime f por todo o conjunto de domínio de f.

Isso mostra que um algorimto de regressão não é perfeito. Deve ser notado, no entanto, que é comum que funções do mundo real não sejam muito surpreendentes, de forma que realmente, dado um conjunto abrangente de amostras, seja possível criar uma aproximação precisa da função objetivo.

A regressão polinomial é apenas um caso específico de regressão, em que a função g sempre será um polinômio. Polinômios são fáceis de computar, e um polinômio pode aproximar qualquer função por um subconjunto do domínio, caso tenha graus o suficiente. Isso os faz uma ótima escolha para esse tipo de trabalho.

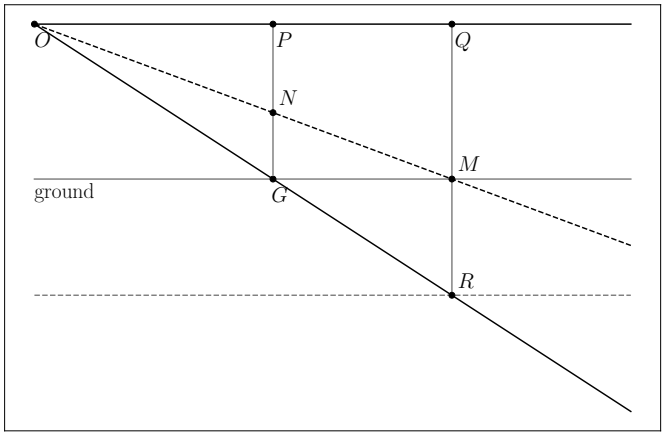
É interessante notar que as redes neurais, quando no paradigma supervisionado, são apenas um tipo bastante elaborado de regressão. Os dados de treinamento são amostras, e o modelo da rede neural é uma função que é aproximada para ser precisa em aproximar corretamente as relações mostradas nas amostras. Quando a rede neural consegue um resultado aceitável nos exemplos, assume-se que também terá bons resultados para todas as entradas em geral.

3.11 Cálculo de distância com câmera única

É importante que o veículo seja capaz de identificar a distância de objetos detectados na pista. Entretanto, o Carro Menor não tem uma câmera estéreo, apenas uma câmera simples. Por isso há dificuldade na noção de distância. Entretanto, esse veículo está sendo preparado para um caso mais simples, em que a pista é completamente plana.

Com a informação de que a pista é plana, é possível calcular a distância do veículo até objetos na pista com precisão considerável usando o modelo da figura 10.

Figura 10 – Modelo de visão de uma pista plana



Na figura 10, o ponto O representa a posição da câmera. Considera-se que a câmera esteja direcionada paralelamente ao chão, ou seja, que esteja apontada na direção da semirreta $\overrightarrow{OP}$. Embora a imagem seja bidimensional, para facilitar a compreensão dos elementos relevantes. Na realidade ela representa a visão de uma câmera no mundo real, tridimensional. O segmento $\overline{PG}$ representa na verdade um plano que pode ser obtido da extrusão deste perpendicularmente ao plano da imagem.

Apenas a parte da image abaixo do horizonte é de interesse, apenas a metade de baixo da imagem capturada. O campo de visão de interesse é delimitado verticalmente pelas semirretas $\overrightarrow{OP}$ e $\overrightarrow{OG}$.

O segmento $\overline{PG}$ representa o plano de projeção da imagem da câmera. Um ponto M na pista é projetado em um ponto N na imagem capturada pela câmera.

Identificam-se as seguintes propriedades na imagem: $\measuredangle QPG = 90^o$, $\measuredangle PQM = 90^o$, $\measuredangle PGM = 90^o$, e que $\measuredangle GNM = \measuredangle PNO = \Theta$.

Define-se:

$$d = OP \qquad B = QR$$
$$D = OQ \qquad \Theta = \measuredangle PNO$$
$$b = PG$$

Considerando $x \in [0, 1), x = \frac{NG}{PG}$, de forma que $bx \in [0, b), bx = NG$. Temos a

partir dos triângulos NGM e OPN, já que $\tan\Theta = \tan\Theta$:

$$\frac{bx}{D-d} = \frac{b}{D}$$
$$D = \frac{d}{1-x}$$

Com isso, é possível identificar, por exemplo, qual a distância até uma certa curva, o que é útil para o controle do veículo. Esses dados também podem ser utilizados para odometria.

4 Detecção e rastreamento de pista

Na etapa inicial do desenvolvimento do projeto, foram analisadas e estudadas possíveis tecnologias de visão computacional. Dentre esssas, o uso de redes neurais convolucionais foi uma das tecnologias mais proeminentes nos maiores projetos de veículos autônomos, se mostrando bastante promissoras. Para o desenvolvimento de redes neurais, entretanto, é necessária uma grande quantidade de dados sobre o ambiente.

Este trabalho tem como alvos dois veículos de escalas diferentes. O menor, chamado *Carro Menor* — até $30cm$ de comprimento e $20cm$ de largura — andará em um ambiente que não imita quase nenhum tipo de condição real, apenas uma pista preta com marcas brancas, plana. O segundo, o *Carro Maior* — com por volta de 1/4 do tamanho de um automóvel de passeio comercial — andará em um ambiente que tenta imitar uma situação real. Em ambos os casos, a pista tem sinalização e esta deve ser respeitada.

O *Carro Menor* poderá utilizar processamento de um computador externo, que nesse caso significa um computador que não está montado ao veículo. Isso é por causa da sua limitação de tamanho. Em seu controle, está sendo utilizado um *Raspberry Pi 3*, que é onde todos os sensores e atuadores estão conectados. Esse computador embarcado possui conectividade *Wi-Fi*, esta sendo então utilizada para a comunicação com um computador externo, onde a parte pesado do processamento é feita. O computador a bordo do veículo apenas captura os dados dos sensores e envia-os até o computador externo, onde este é processado e enviado de volta para o veículo. A latência extra resultante dessa transferência tem um custo negligível quando comparada com a limitação de hardware que existiria caso todo o processamento tivesse de ser montado a bordo de um veículo tão pequeno.

O *Carro Menor* carregará o seu próprio computador. E todo o seu processamento será interno.

Depois disso, começa o processo de obtenção de dados, em que imagens da pista são obtidas, com o propósito do treinamento de uma rede neural para o reconhecimento das linhas que sinalizam a pista. Para o caso do *Carro Maior*, como a situação imita a realidade, já existem várias bases de dados disponíveis, incluindo redes neurais pré-treinadas, as quais é apenas necessário fazer o *fine tunning*.

A rede neural utilizada para a detecção de imagens é um tipo de rede LSTM, o motivo da escolha de tal arquitetura, é que dirigir um veículo é um processo interativo e contínuo. A entrada no algoritmo não é apenas uma imagem, mas um vídeo: uma sequência de imagens que progride com o tempo. As informações detectadas sobre a pista numa imagem, não se tornam irrelevantes imediatamente após a chegada da próxima imagem. Muito pelo contrário, é possível até mesmo predizer (mesmo que não muito preciso) qual

Figura 11 – Carro Maior, fonte: o próprio autor

Figura 12 – Carro Menor, fonte: o próprio autor

será o próximo estado da pista, analizando a progressão dos estados anteriores. Portanto, usar os estados anteriores prova-se uma vantagem que não é aproveitada em uma CNN comum.

O lado negativo da escolha da LSTM, é que ela é mais complexa do que as CNNs, dependendo de mais tempo e memória para o seu treinamento, e em menor grau, também é mais pesada em questão de processamento da imagem, pois costuma possuir mais pesos e nós.

Este trabalho não usa a abordagem fim-a-fim, para maior controle sobre como os veículos agem. A rede neural apenas detecta as marcações na pista, e depois disso, algoritmos clássicos são usados para criar o modelo da pista. O modelo da pista também leva a informação temporal em consideração, pois ao reutilizar os dados da iteração anterior, evita-se mudanças muito bruscas causadas por falta de dados em uma iteração. Por exemplo, se uma faixa da pista está muito apagada em um pequeno trecho do caminho, não deve-se considerar que a pista desapareceu. Atualiza-se o modelo da maneira mais conservadora possível até que dados mais confiáveis sejam recebidos. Caso esses dados não cheguem, o veículo deve procurar parar, de maneira segura.

Com as redes neurais treinadas para obter as linhas de sinalização da pista em ambos os ambientes, o restante do algoritmo é similar em ambos os carros. Encontrar as linhas na imagem não é o suficiente para que o carro tenha ideia da situação em que está. Portanto, outra fase de análise de imagem é feita, na qual as linhas são separadas em faixa da esquerda e direita. Depois disso, essas são usadas para modelar uma curva que as represente. Com essa curva, é possível que o carro se situe na pista com precisão, dentro de uma faixa.

É claro que, em uma aplicação real, outros tipos de detecção devem ser utiizados,

como detecção de área navegável, de veículos e pedestres. Com esses outros dados, seria possível que o algoritmo fizesse um bom julgamento da situação e pudesse, por exemplo, estacionar no acostamento caso detectasse que não pode prosseguir de maneira segura. Ou mesmo, nos casos da automação nível 4 ou inferior, pedir pela intervenção de um humano.

Mesmo que apenas detectar a sinalização das faixas da pista não seja o suficiente para a implementação de um veículo autônomo funcional, ela ainda é importante. A sinalização deve sempre ser respeitada, e por isso, deve ser detectada corretamente. Em uma via com múltiplas faixas, o veículo deve ser inteligente o suficiente para não ocupar duas faixas, ou pior, atravessá-la e entrar na contra-mão.

É também importante que o veículo seja capaz de detectar e respeitar semaforos, limites de velocidade, e outros sinais em geral incluindo os que estão pintados diretamente na pista como as faixas de pedestre. Detectar e respeitar policiais e veículos prioritários, como ambulâncias e bombeiros. Deve detectar situações de congestionamento e acidentes, e até mesmo como agir em caso de buracos na pista. Os problemas para a implementação desse tipo de veículo podem ser ditos como quase incontáveis.

Uma das coisas que pode se tornar um facilitador dessa tecnologia é que, o design das pistas e sinalização pode ser melhorado para a detecção por computadores. Hoje em dia, toda a sinalização usada nas pistas, de faixas, placas e até semáforos, é completamente pensada para ser vista e interpretada por humanos. Caso os veículos autônomos se tornem populares, e consigam alguma aceitação nas pistas atuais, isso aceleraria o desenvolvimento de sinalização voltada para esses motoristas mecânicos.

O ambiente do Carro Menor é uma pista feita com partes modulares. A pista é dividida em retângulos — pisos, com duas faixas a $1m$ de distância entre elas, e uma faixa segmentada no centro, entre as outras duas. A pista pode então ser montada ao colocar os pisos lado a lado, de forma com que as faixas de um piso estejam sempre conectadas com faixas de mais dois pisos.

O piso é preto, e as faixas são brancas, coloração inspirada nas pistas de asfalto reais. O preto é fosco para evitar demasiada reflexividade, que é pode ser detrimental a capturação de imagens. Para simplificar ainda mais o problema, a pista fica dentro de um ambiente interno, com iluminação uniforme. Esse tipo de desafio tem poucos obstáculos, mas serve de ótimo ponto inicial para o teste dos algoritmos de controle: não é um ambiente real, assim não existe a possibilidade de acidentes que causem dano à pessoas ou equipamento. Esse ambiente simplifica a detecção das faixas, e permite o foco na parte simbólica do algoritmo: modelar a pista.

Com o Carro Maior, o teste será feito em uma situação parecida com a real, embora a escala possa, ou não ser a escala real. O chão é de asfalto e as faixas são pintadas como

na realidade. Podem existir placas de trânsito, faixas de pedestre e semáforos. A pista está num ambiente externo, sujeito a iluminação irregular do sol, com sombras e reflexos. Também pode ser influenciado por situações climáticas como pista molhada ou neblina. É claro, neste trabalho, não pretende-se testar em todas essas condições, mas essas serão condições possíveis de serem consideradas em trabalhos futuros.

O Carro Maior será testado nas ruas da UESB, que foram projetadas para automóveis convencionais. Entretanto, pretende-se levar esse mesmo veículo para competições nacionais em escala reduzida, imitando a situação real.

Em ambos os ambientes, o objetivo do algoritmo é encontrar as linhas mais a esquerda e a direita do veículo, e depois disso computar uma linha central ao qual o veículo deve se centralizar. Isso requer uma pista com sinalização adequada, mas isto está dentro das limitações esperadas. Com os testes é possível ver que os pontos mais difíceis do trajeto são as interceções. Durante elas, é bem mais difícil dizer qual é a linha que o carro deve seguir. De fato, sem informação prévia sobre o trajeto, preferencialmente com um GPS, é impossível que o veículo decida pra onde deve ir, a menos que aja aleatoriamente.

Quando não existem interceções no caminho, o veículo apenas deve continuar em frente em uma velocidade que o veículo considere adequada. Coisas como a detecção de obstáculos e verificação de limites de velocidade, entre outros sinais de trânsito, estão fora do escopo do trabalho.

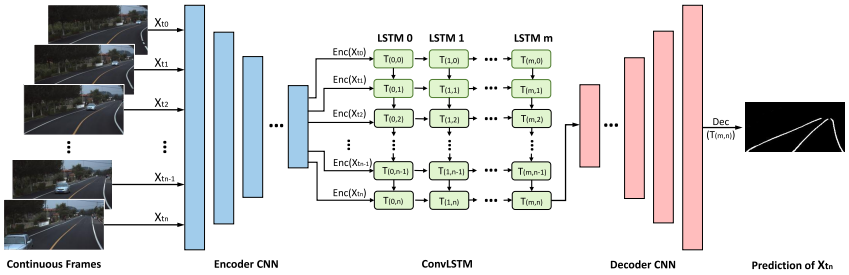
Já que a direção de um automóvel exige um processo de decisão em função do tempo, a LSTM foi o paradigma de rede neural escolhido. Um veículo está em um ambiente dinâmico, que segue regras conhecidas, por exemplo: um carro que estava em uma posição s_0 no tempo t_0, e em uma posição s_1 no tempo t_1 está se movendo em relação à câmera. Se o carro está se movendo no tempo t_1, então espera-se que a posição s_2 do carro no tempo t_2 seja diferente de s_1.

Pensando desse modo, uma rede neural que detecta um carro em certa posição numa imagem, pode esperar encontrar o mesmo carro perto desta mesma posição na próxima. Fica claro que os dados encontrados nas imagens anteriores é extemamente útil mesmo nas imagens subsequentes. Isso significa que a rede neural trabalha com mais dados sobre o ambiente em cada iteração: os dados recém chegados, e os dados de algumas imagens imediatamente anteriores à esses. É dessa maneira que a LSTM funciona.

4.1 Detectando as linhas da pista

Para a classificação de linhas da pista, é utilizada uma rede neural na arquitetura UNet ConvLSTM (ZOU et al., 2019) (figura 13). Essa rede neural é totalmente convolucional, e também recursiva. Uma rede neural convolucional apresenta uma operação de

Figura 13 – Modelo proposto por Zou da UNet ConvLSTM (ZOU et al., 2019)



filtragem de imagem, encontrando características na imagem, nesse caso as características são as linhas na pista. Como essa rede também é recursiva, esses filtros mudam com o tempo: cada imagem muda o estado interno da rede, mudando um pouco esse filtro. Isso permite a rede neural ser treinada para capturar padrões de movimento, que podem ser aproveitadas na detecção das linhas.

A vantagem do uso de uma rede neural em comparação com um filtro tradicional, é que a rede neural oferece melhor acurácia, sendo tanto melhores em encontrar as linhas, quanto têm menos falso positivos: não constumam encontrar linhas onde não existam.

Já que a rede neural não toma controle total do controle do veículo, uma menor parte do algoritmo sofre o problema da inerente inexplicabilidade de uma rede neural.

4.1.1 Arquitetura UNet-ConvLSTM

A arquitetura UNet-ConvLSTM é uma arquitetura muito eficaz na detecção de linhas da pista. Ele é formado pela mistura da CNN UNet, um modelo de rede neural convolucional que teve muito sucesso na segmentação biomédica. Ela se mostrou muito eficaz na detecção das linhas da pista, mas como o veículo autônomo tem como entrada um vídeo, apenas tratar imagens independentemente não faz completo uso dos dados. Para aumentar a acurácia da detecção, o LSTM pode ser usado para levar a evolução da entrada em relação ao tempo em consideração. A ConvLSTM é um modelo de LSTM adaptado para redes neurais convolucionais (ZOU et al., 2019).

A rede é organizada em três seções: a seção codificadora de uma CNN, nesse caso a UNet, depois o ConvLSTM, e finalmente a camada decodificadora da Unet. Como resultado a rede irá detectar as linhas da pista (ZOU et al., 2019). Esse modelo pode ser visto na figura 13.

A implementação do modelo usando o PyTorch foi disponibilizada pelos autores em um repositório no GitHub. O PyTorch é uma biblioteca em python que permite a interação com a biblioteca Torch. Essa biblioteca permite a fácil implementação de redes neurais por causa de sua ferramenta de gradiente automático: dada uma operação em tensores, a biblioteca computa a matriz jacobiana que representa a derivada daquela

operação. Operações podem ser combinadas, mantendo-se a invariante de que qualquer uma das operações suportadas é derivado automaticamente pela biblioteca.

A biblioteca também possui blocos de construção de redes neurais. E é capaz de operar em um modo acelerado por hardware, mais especificamente, placas de vídeo.

O modelo disponibilizado por Zou et al. (2019) pode então ser especializado para a situação mais específica do problema, num processo conhecido como *fine tunning*. Durante esse processo, dados de treinamento novos, coletados no ambiente ao qual o veículo deverá operar, são usados para adaptar a rede a esse ambiente. Isso também serve para adaptar a rede a mudanças na posição da câmera em relação ao veículo e a pista, e outras peculiaridades do sensor.

4.1.2 Conjuto de dados para testes e treinamento

Dois conjuntos de dados, criados e anotados por terceiros, foram considerados para testes no trabalho. Entre eles, o CULane foi escolhido por ter situações mais diversas, e mais imagens. As imagens foram obtidas em sequência, ou seja, como um vídeo, algo que é importante no contexto desse projeto. Para cada frame, existe uma anotação descrevendo as linhas da pista — como splines cúbicas — essas formam o "ground truth" no processo de treinamento e testes de acurácia (PAN et al., 2018).

4.2 Obtenção de amostras das linhas da pista

Depois de identificar com segmentação semântica, é necessário obter amostras dessas linhas, e de maneira eficiente, já que o problema deve ser resolvido em tempo real. As amostras são usadas para fazer o modelo da pista, e obtidas com o seguinte algoritmo:

1. Divida a imagem em N fatias verticais;
2. Para cada fatia vertical $fatia$ faça:
    a) Detecte todos os aglomerados de pixeis da $fatia$;
    b) Selecione o aglomerado imediatamente à esquerda do centro de visão horizontalmente, seu ponto central sendo e;
    c) Selecione o aglomerado imediatamente à direita do centro de visão horizontalmente, seu ponto central sendo d;
    d) Coloque o ponto e na coleção E, e o ponto d na coleção D;
3. Atualize o modelo da linha esquerda da pista com os pontos do conjunto E;
4. Atualize o modelo da linha direita da pista com os pontos do conjunto D;

Nos passos 3 e 4, a atualização dos pontos é feita da seguinte maneira:

4.2.1 Atualização do modelo

A pista é modelada por dois polinômios, um a esquerda e outro a direita, delimitando a faixa da pista. O veículo deve sempre se manter no centro dessas faixas, saindo delas apenas quando for necessário, por exemplo, no caso de uma mudança de faixa ou cruzamentos, ou mesmo para evitar um acidente. Cada novo frame carrega novas informações sobre as linhas, que podem ser usadas para atualizar esses dois polinômios.

Isso significa que o modelo não é recriado do zero com a chegada de novos dados, mas evolui progressivante com a chegada destes. A velocidade com que o modelo é atualizado depende da quantidade de dados que são extraidos da imagem de entrada e a confiabilidade destes. Se as linhas são muito bem demarcadas, então o modelo pode ser atualizado mais agressivamente. Se existem falhas ou inconsistências com o conhecimento prévio de uma marcação normal (como linhas que se curvam abruptamente), o modelo tende a permanecer como estava anteriormente, com ajustes menores.

Essa atualização deve ser sintonizada de modo a oferecer reações rápidas, enquanto evita tomar decisões precipitadas. Esses dois atributos estão em conflito um com o outro. Acelerar a responsividade do modelo tende a fazer com que o modelo reaja demais à uma entrada errônea. Esperar por novos dados para ter certeza sobre a situação real do veículo evitaria um ajuste exagerado, mas faria com que o sistema demore mais de responder às mudanças no ambiente.

Pode ser visto no trabalho de Vivacqua (2017) que uma análise probabilística da confiabilidade dos dados foi usada para resolver esse problema. Futuramente, este trabalho pode implementar algo semelhante para melhorar os resultados.

O modo de sintonizar a taxa de atualização modelo é usando um número que indica a tendência do algoritmo a manter o modelo no estado anterior. Esse fator pode ser visto como o fator *inércia*. Quanto maior esse fator, menos o modelo é atualizado a cada quadro. Se ele for zero, o algoritmo desconsidera completamente os estado anterior e refaz o modelo do zero a cada iteração.

O modo com que o algoritmo considera a inércia é usando heurísticas. Assume-se que a maior parte dos pontos do quadro anterior ainda são relevantes na situação atual. Então amostras do modelo da situação anterior são obtidas para suplementar as amostras obtidas pela rede neural na iteração atual. Quanto maior o fator inércia, mais amostras são obtidas. Quando o fator é zero, nenhuma amostra é usada para suplementar os dados, o que faz com que o modelo seja extremamente reativo a situação atual.

O polinômio é então comparado com o polinômios dos dados de teste da CuLane, para avaliar a acurácia da rede. É importante ressaltar que os dados de teste obtém

dados de todas as linhas da pista, enquanto o algoritmo proposto obtém apenas as linhas imediatamente a esquerda e a direita do centro do veículo. Essas linhas são as mais importantes para manter o veículo dentro da faixa. Por isso, as linhas dos dados de teste que não forem as centrais não são consideradas no cálculo de acurácia.

4.3 Estimação do ponto de desaparecimento

Depois de encontrado os polinômios que modelam a pista, é preciso saber qual é o ponto de desaparecimento da pista em relação a visão da câmera, para que este polinômio seja interpretado corretamente. Os dados de amostra da pista são vistos como pares (x, y) que representam as suas coordenadas no vídeo da câmera. Essas coordenadas são normalizadas, sendo que $x = -1$ represente a borda esquerda e $x = 1$ a borda da direita. $y = -1$ e $y = 1$ representam as bordas de cima e de baixo respectivamente. Nem toda a imagem capturada pela câmera conterá a pista. Partes dessa imagem não conterão amostras.

Se a câmera estiver filmando uma pista completamente plana, o ponto de desaparecimento pode ser calculado com o ângulo da câmera em relação ao chão.

Entretanto, não podemos esperar que a pista seja plana por todo o trajeto do veículo. Por causa disso, é difícil saber até onde as linhas da pista estão indo. Mais precisamente, é difícil dizer se a partir de um determinado limite no eixo y da visão da imagem, não existem amostras porque não há pista naquela imagem, ou simplesmente por que o algoritmo falhou em obter tais amostras. Se o primeiro for o caso, então os dados das amostras anteriores naquele ponto com certeza estão incorretos. Cajo seja o segundo, então seria útil usar o modelo construído com as amostras anteriores para complementar os dados faltantes.

Por esse motivo, estima-se um ponto no eixo y que seja a linha do horizonte, essa linha, caso escolhida corretamente, é a linha imaginária em que o ponto de desaparecimento está. A partir dessa linha, a pista não é mais projetada nos sensores da câmera. Esses dados podem ser utilizados para o cálculo de distância da pista, caso haja conhecimento prévio do ângulo em que a câmera foi instalada.

Isso é importante por que em uma situação real, o carro deve ser capaz de identificar a distância de objetos na pista até ele mesmo. Por exemplo, outros veículos. Ele deve ser capaz de julgar se os carros estão parados ou se movendo. Se um carro a sua frente estiver se movendo, e existir uma distância segura entre eles, então faz sentido que o veículo autônomo continue. Caso a distância esteja se encurtando, então ele deve diminuir a velocidade também. Caso o veículo da frente pare abruptamente, ele deve ser capaz de perceber isso e freiar o mais rápido possível. Todas essas decisões são cruciais em um ambiente real, e são também consideradas nas simulações.

Caso o carro tenha apenas uma única câmera, e mesmo quando tenha mais de uma, é importante que ele seja capaz de estimar o ponto de desaparecimento para ser capaz de estimar essa distância.

Outro motivo, é que se o ponto de desaparecimento é desconhecido, então o algoritmo poderia pegar amostras de uma parte do polinômio que não modela nenhuma pista real. Isso é inaceitável em uma situação real.

5 Resultados e conclusão

Quando o projeto estava nos primeiros passos, ainda em 2019. A primeira versão do algoritmo foi desenvolvida. Essa primeira versão foi muito menos eficaz do que o esperado mas forneceu muita experiência sobre que tipos de problemas precisariam ser resolvidos. O mais importante deles foi a conclusão sobre a detecção de linhas com a abordagem clássica. A abordagem usando Canny e Hough-Lines se mostrou difícil de implementar de maneira a chegar a um nível satisfatório.

O maior problema da abordagem clássica para a detecção de linhas é a sua vulnerabilidade ao ruído. As diferenças entre as marcações das linhas da pista, e outros tipos de marcas, como imperfeições na pista, sombras e até mesmo faixas de pedestre são mais sutis do que inicialmente aparentes. No primeiro teste em uma competição de carro autônomo em Vitória da Conquista, o carro menor conseguiu completar uma volta pelo percurso, mas isso apenas depois de diversas falhas.

Existem muitos fatores que complicam a implementação desse tipo de detector. Os reflexos causados pelo sol, ou pela pista molhada, o asfalto que pode mudar de cor durante o percurso. Tudo isso era difícil de prever.

Em meados de 2019, o carro menor foi levado ao IFBA de Vitória da Conquista para a Segunda Mostra e Competição Robótica do IFBA. Durante os testes na UESB, o ambiente era mais fechado e o sol não atingia diretamente a pista, entretanto, durante a mostra de robótica, o ambiente era mais aberto e a iluminação portanto era muito diferente. Além disso, lá, ocasionalmente, o sol atingia a pista diretamente. Tudo isso fez com que o detector clássico parace de funcionar, e o veículo fosse incapaz de detectar as linhas corretamente. Algumas bordas encontradas eram feitas pelas sombras, que se descavam muito quando o sol atingia a pista, complicando a classificação dos pontos.

Alguns ajustes nos parâmetros da filtragem foram suficientes para que o veículo ainda conseguisse detectar as linhas, embora a detecção ainda não fosse consistente. Quando o veículo atingia uma curva, hora detectava-a corretamente, hora não. Era difícil prever. A pista tinha quatro curvas, em um formato de um quadrado com cantos arredondados. O carro andava no sentido anti-horário, e assim precisaria apenas fazer quatro curvas a esquerda no momento certo para concluir com sucesso o desafio.

Felizmente, quando a competição começou, nas tentativas que tínhamos direito, o carro Menor conseguiu completar pelo menos uma volta completa. Nenhum outro veículo foi capaz de repetir o feito, assim ele ficou em primeiro lugar.

Embora a competição foi um sucesso, a performance do veículo ainda foi muito

Figura 14 – Segunda Mostra e Competição Robótica do IFBA. O nosso veículo pode ser visto na pista no canto inferior direito. Fonte: o próprio autor.

ruim. Pode-se dizer que o único motivo dele ter vencido era por que a performance dos outros veículos não era melhor, também se saíram muito mal. Portanto, era necessário melhorar a detecção da pista. Também era necessário fazer vários ajustes ao veículo para melhorar a responsividade e consistência dos controles, que muitas vezes paravam de funcionar depois de algum tempo de uso.

Quando não era possível testar o algoritmo nos veículos, o simulador Carla fora usado. Essa ferramente fornece um meio extremamente simples de realizar experimentos. Infelizmente, ela também demanda muito processamento, e não é muito estável, por ainda estar em fase de desenvolvimento.

Durante esses testes, juntando ao conhecimento das fraquezas da detecção clássica, começou a busca por outras soluções. Redes neurais eram bem conhecidas pela equipe, e portanto logo estavam sendo usadas nos testes.

O uso das redes neurais forneceu um resultado muito superior. Além disso, com a grande quantidade de bases de dados anotadas, o trabalho de considerar e se adaptar a todo tipo de situação adversa fica para a máquina. Desse modo, a segunda versão do algoritmo, usando redes neurais, é muito superior. Infelizmente, ainda não houve oportunidades para o teste dessa versão nos veículos reais.

Depois que a detecção de linhas chegou a um nível satisfatório, o restante do algoritmo foi testado de uma maneira diferente: Os quadros eram processados pela rede neural, e os resultados eram guardados em um diretório. A segunda parte do algoritmo era desenvolvida separadamente, tendo como entrada a saída da rede neural, já pré-

processada.

Esse modelo fazia com que as iterações pudessem ser testadas com muita rapidez, pois não gastava-se tempo processando os frames com a rede neural (o processamento das imagens é uma das partes mais custosas). Além disso, não é difícil integrar novamente as duas partes dos algoritmos. Para um trabalho em que experimentos e ajustes eram feitos constantemente, essa estratégia poupou muito tempo. O restante do algoritmo não precisou ser substituído, apenas melhorado para fazer melhor uso das vantagens da nova detecção. Muitas das modificações na geração de modelo foram feitas pois a melhoria na filtragem destacou outros problemas da versão antiga.

A experiência obtida com a construção dos veículos também foi valiosa. Inicialmente, o controle de esterçamento era impreciso e causava confusão: era difícil saber se uma curva errada era um problema de software ou hardware. Um novo carro de controle remoto foi adaptado, este com um controle de esterçamento muito mais preciso. Isso imediatamente melhorou a performance do veículo no trajeto.

Também houve a alteração no modelo dos circuitos digitais do veículo. No começo, o controle físico, era feito por uma placa Arduino. Entretanto, como o Arduino não era muito flexível na parte de Software, um Raspberry Pi 3 foi acoplado junto ao veículo para o controle deste. O Raspberry se comunicava com o Arduino por meio da porta serial. Esse modelo mostrava muita latência para o controle. Em retrospectiva, talvez a comunicação com o Arduino não devesse ser por meio da porta serial, mas a solução aplicada foi a remoção completa dessa parte do circuito. O controle era feito totalmente pelo Raspberry. Isso envolvia um pouco mais de software, mas removia toda a latência de controle.

Ainda com o veículo de pequena escala, como o controle era feito por um computador externo, o que é permitido na competição para o qual está sendo desenvolvido. Houvera, no entanto, um grande problema com a latência do controle remoto. Um quadro devia ser transmitido do veículo até um computador mais potente. Mas um quadro é algo muito grande, que demora de passar pela rede. Não apenas isso, erros de rede podem acontecer, o que pode causar perda ou atraso de dados.

5.1 Testes com dados de exemplo

Na figura 15 é possível ver o algoritmo obtendo os dois polinômios a partir da imagem filtrada e desenhando-os na parte esquerda. Na parte direita, é possível ver a imagem filtrada pela rede neural (em preto e branco ao fundo) e os dados obtidos pela segunda camada do algoritmo (abordagem clássica) sendo desenhados por cima.

É possível ver que o algoritmo consegue obter as linhas da esquerda e da direita

Figura 15 – Visualização de testes, fonte: o próprio autor

a partir dos dados. O exemplo é tirado de um vídeo. Cada quadro subsequente mantém dados obtidos dos quadros anteriores. Tanto a rede neural, por ser LSTM, quanto o algoritmo clássico mantém o estado durante a iteração, para ajudar na detecção.

5.2 Conclusão

O modelo proposto mostra-se capaz de encontrar as linhas esquerda e direita da pista, cobrindo parte dos casos em que a rede neural não é capaz de obter dados suficientes, além de apresentar os dados de maneira mais fácil de ser computada: polinômios, em comparação com imagens.

O modelo de rede neural totalmente convolucional é mais eficaz em encontrar linhas na pista do que filtros clássicos, como o Canny. Uma rede neural com LSTM é capaz de resolver quase que completamente o problema da obstrução temporária que um veículo tem sobre a sinalização da pista. Assim, existe menos trabalho para a parte simbólica que é a síntese dos dados filtrados e processados sobre as linhas da pista em informações ainda mais abstratas, os polinômios.

Isso significa que é possível combinar a flexibilidade das redes neurais convolucionais com a previsibilidade de uma algoritmo simbólico clássico. Previsibilidade esta que é de grande importância em sistemas críticos. Algo que seria bem difícil de obter com uma abordagem que usa redes neurais de ponta a ponta.

Para trabalhos futuros, é possível usar o trabalho de Vivacqua (2017) com a medição de confiança das amostras obtidas na pista para a atualização do modelo. Também pode ser possível a utilização do filtro de Kalman para lidar com possíveis irregularidades na entrada.

Referências

BADUE, Claudine et al. Self-driving cars: A survey. **Expert Systems with Applications**, Elsevier, 2020.

BENSON, Aaron et al. **AAA Foundation for Traffic Safety Potential Reduction in Crashes, Injuries and Deaths from Large-Scale Deployment of Advanced Driver Assistance Systems**. 2018. Disponível em: <https://aaafoundation.org/potential-reduction-in-crashes-injuries-and-deaths-from-large-scale-deployment-of-advanced-driver-assistance-systems/>. Acesso em: 3 set. 2019.

BERRIEL, Rodrigo Ferreira. **Vision-based ego-lane analysis system: dataset and algorithms**. 2016. Diss. (Mestrado) – Programa de Pós-Graduação em Informática.

BOJARSKI, Mariusz et al. End to end learning for self-driving cars. **arXiv preprint arXiv:1604.07316**, 2016.

CANNY, John. A computational approach to edge detection. **IEEE Transactions on pattern analysis and machine intelligence**, Ieee, n. 6, p. 679–698, 1986.

CARLINI, Nicholas et al. The secret sharer: Evaluating and testing unintended memorization in neural networks. **28th USENIX Security Symposium (USENIX Security 19)**, p. 267–284, 2019.

DAHL, Richard. **Vehicular manslaughter: the global epidemic of traffic deaths**. [S.l.]: National Institue of Environmental Health Sciences, 2004.

DU, Juan. Understanding of object detection based on CNN family and YOLO. In: IOP PUBLISHING, 1. JOURNAL of Physics: Conference Series. [S.l.: s.n.], 2018. v. 1004, p. 012029.

FISHER, Robert et al. **Hough Transform**. 2004. Disponível em: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm>. Acesso em: 2 dez. 2019.

FUKUSHIMA, Kunihiko; MIYAKE, Sei. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. **Competition and cooperation in neural nets**, Springer, p. 267–285, 1982.

HARNER, Isabel. **The 5 Autonomous Driving Levels Explained**. 2020. Disponível em: <https://www.iotforall.com/5-autonomous-driving-levels-explained>. Acesso em: 24 out. 2020.

LEITE, Joel et al. Carro sem motorista será táxi em São Carlos. UOL Carros, 2015.

LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully Convolutional Networks for Semantic Segmentation. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, jun. 2015.

NARANJO, J.E. et al. Electric Power Steering Automation for Autonomous Driving. **International Conference on Computer Aided Systems Theory**, p. 519–524, fev. 2005. DOI: 10.1007/11556985_67.

NORMAN, Donald A. The 'problem'with automation: inappropriate feedback and interaction, not 'over-automation'. **Philosophical Transactions of the Royal Society of London. B, Biological Sciences**, The Royal Society London, v. 327, n. 1241, p. 585–593, 1990.

NVIDIA. **CUDA GPUs**. 2011. Disponível em: <http://web.archive.org/web/20110624101207/http://developer.nvidia.com/cuda-gpus>. Acesso em: 19 nov. 2019.

_____. **Improving semantic segmentation via video propagation and label relaxation**. 2019. Disponível em: <https://github.com/NVIDIA/semantic-segmentation>. Acesso em: 26 nov. 2019.

OPENCV. **Introduction**. Disponível em: <https://web.archive.org/web/20191203173240/https://docs.opencv.org/4.1.2/d1/dfb/intro.html>. Acesso em: 3 dez. 2019.

PAN, Xingang et al. Spatial As Deep: Spatial CNN for Traffic Scene Understanding. **AAAI Conference on Artificial Intelligence (AAAI)**, fev. 2018.

PERSLOW, Helena; CARLSON, Jeremy. **ADAS–Current & Future Perspectives**. [S.l.]: HIS Automotive, Frankfurt, 2015.

REDA, Fitsum A et al. SDC-Net: Video prediction using spatially-displaced convolution. **Proceedings of the European Conference on Computer Vision (ECCV)**, p. 718–733, 2018.

SOLAIMAN, SM. Corporate Vehicular Manslaughter Provisions In The Bangladesh Road Transport Act 2018: A Textual Comparison With Their Equivalents in Australia. In: SPRINGER. CRIMINAL Law Forum. [S.l.: s.n.], 2020. P. 1–49.

STANFORD UNIVERSITY. **Stanley**. 2005. Disponível em: <https://robots.ieee.org/robots/stanley/>. Acesso em: 23 out. 2020.

STAUDEMEYER, Ralf C; MORRIS, Eric Rothstein. Understanding LSTM–a tutorial into Long Short-Term Memory Recurrent Neural Networks. **arXiv preprint arXiv:1909.09586**, 2019.

THRUN, Sebastian et al. Stanley: The robot that won the DARPA Grand Challenge. **Journal of field Robotics**, Wiley Online Library, v. 23, n. 9, p. 661–692, 2006.

THYS, Simen; VAN RANST, Wiebe; GOEDEME, Toon. Fooling Automated Surveillance Cameras: Adversarial Patches to Attack Person Detection. **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops**, jun. 2019.

VIVACQUA, Rafael Peixoto Derenzi. **Back Lane Marking Registry: uma abordagem de localização e seguimento de caminho por veículos autônomos em via sinalizada**. 2017.

WANG, Jianyu et al. Visual concepts and compositional voting. **arXiv preprint arXiv:1711.04451**, 2017.

WANG, Jun et al. Lane detection algorithm based on temporal–spatial information matching and fusion. **CAAI Transactions on Intelligence Technology**, IET, v. 2, n. 4, p. 154–165, 2019.

WANG, Yue; SHEN, Dinggang; TEOH, Eam Khwang. Lane detection using catmull-rom spline. **IEEE International Conference on Intelligent Vehicles**, v. 1, p. 51–57, 1998.

WANG, Yue; TEOH, Eam Khwang; SHEN, Dinggang. Lane detection and tracking using B-Snake. **Image and Vision computing**, Elsevier, v. 22, n. 4, p. 269–280, 2004.

WRAPPER. **Wrapper package for OpenCV python bindings**. 2019. Disponível em: <https://pypi.org/project/opencv-python/>. Acesso em: 3 dez. 2019.

XU, Kaidi et al. Adversarial t-shirt! evading person detectors in a physical world. **arXiv**, arxiv–1910, 2019.

YURTSEVER, Ekim et al. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. **arXiv preprint arXiv:1906.05113**, 2019.

_____. A survey of autonomous driving: Common practices and emerging technologies. **IEEE Access**, IEEE, v. 8, p. 58443–58469, 2020.

ZHANG, Chiyuan et al. A study on overfitting in deep reinforcement learning. **arXiv preprint arXiv:1804.06893**, 2018.

ZHU, Bin et al. Fusion of Sensors Data in Automotive Radar Systems: A Spectral Estimation Approach. **arXiv preprint arXiv:1908.02504**, 2019.

ZHU, Yi et al. Improving Semantic Segmentation via Video Propagation and Label Relaxation. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, jun. 2019.

ZOU, Qin et al. Robust lane detection from continuous driving scenes using deep neural networks. **IEEE transactions on vehicular technology**, IEEE, v. 69, n. 1, p. 41–54, 2019.