DESENVOLVIMENTO DE UM SISTEMA EMBARCADO PARA O CÁLCULO DE ESPAÇO LIVRE NA FRENTE DE UM VEÍCULO AUTÔNOMO UTILIZANDO MÉTODOS DA INTELIGÊNCIA ARTIFICIAL

Giovanni Corradini Nunes [1]; Alexandre Harayashiki Moreira [2]; Eduardo Lobo Lustosa Cabral [3]

[1] Aluno de Iniciação Científica do Instituto Mauá de Tecnologia (IMT);
[2] Professor do Instituto Mauá de Tecnologia (IMT);
[3] Professor do Instituto Mauá de Tecnologia (IMT).

Resumo. *Esse trabalho tem como o objetivo calcular o espaço livre a frente de um veículo utilizando câmeras e métodos da inteligência artificial. Um modelo YOLO (versão 11) (KOTTHAPALLI et al., 2025) foi retreinado para a segmentação da rua e outro modelo YOLO foi retreinado para a deteção de pessoas e objetos a frente do veículo. Após essa etapa, foi utilizado o modelo Depth-Pro (BOCHKOWSKI et al., 2024), para utilizar a técnica de destilação de modelos com a finalidade de desenvolver um modelo aluno capaz de ser embarcado em um hardware com baixa capacidade computacional. Após a criação do modelo aluno, foi elaborado um algoritmo para a junção dos dois modelos para a definição dos limites da rua a partir da determinação das distancias desses pontos. O algoritmo foi testado e implementado em um hardware embarcado NVIDIA JETSON AGX ORIN (NVIDIA, 2023).*

Introdução

Com o avanço das tecnologias embarcadas na indústria automotiva, os veículos têm se tornado progressivamente mais inteligentes e autônomos ao longo dos últimos anos. Entre os sensores comumente utilizados para a determinação do espaço à frente de veículos autônomos, destacam-se os sensores LiDAR, amplamente reconhecidos por sua alta precisão na geração de mapas tridimensionais e pela robustez diante de diferentes condições ambientais (YADAV et al., 2023). Apesar dessas vantagens, o custo elevado dessa tecnologia impacta diretamente o preço final das plataformas que a utilizam, o que limita sua adoção em larga escala. Nesse contexto, torna-se estratégica a busca por alternativas mais acessíveis e de menor custo.

Paralelamente, técnicas de visão computacional e inteligência artificial têm avançado de forma significativa, especialmente no campo da estimativa de profundidade monocular. Trabalhos recentes demonstram que redes neurais profundas conseguem estimar profundidade com boa precisão utilizando apenas câmeras convencionais. (XUE et al., 2020) por exemplo, apresentaram um método auto supervisionado hierárquico para estimativa de profundidade absoluta voltado para aplicações em veículos autônomos. (LI et al., 2023) propuseram uma abordagem baseada em redes convolucionais com módulos de atenção para aprimorar a estimativa de profundidade em ambientes estruturados. De forma complementar, (AFSHAR et al., 2023) desenvolveram um método eficiente baseado em YOLO e mecanismos de atenção para estimar, em tempo real, as distâncias relativas de veículos, demonstrando que sistemas puramente visuais podem competir com sensores dedicados.

Essas investigações reforçam que a substituição parcial ou total de sensores LiDAR por câmeras associadas a modelos de inteligência artificial é tecnicamente viável e economicamente atrativa, reduzindo significativamente o custo do sistema sem comprometer o desempenho de forma relevante. Diante desse cenário, este trabalho propõe o desenvolvimento de um método para determinar o espaço livre à frente de veículos autônomos utilizando câmeras e técnicas de inteligência artificial para estimativa de distâncias e apoio à elaboração de trajetórias seguras.

Material e Métodos

O desenvolvimento deste trabalho de iniciação científica foi dividido em cinco etapas principais:

1. **Aquisição de imagens:** coleta de imagens no entorno do campus do Instituto Mauá de tecnologia para a formação de um *dataset*;
2. **Rotulação das imagens:** anotação manual das imagens obtidas, com o objetivo de treinar uma rede neural do tipo YOLOv11n para detecção de objetos relevantes ao cenário automotivo;
3. **Destilação do modelo Depth-Pro:** utilização do modelo Depth-Pro como rede professor para a geração de mapas de profundidade e posterior treinamento do modelo aluno;
4. **Integração dos modelos:** desenvolvimento de um algoritmo para a combinação das informações provenientes do modelo YOLO e do modelo aluno, permitindo a estimativa do espaço livre à frente do veículo;
5. **Implementação embarcada:** implantação do algoritmo final em um *hardware* embarcado, visando a validação do sistema em condições reais de operação.

Etapa 1 – Aquisição de imagens

Para a etapa de aquisição de imagens, foi utilizado o veículo de testes ORBI, pertencente ao grupo de pesquisa SMIR do Instituto Mauá de Tecnologia. A câmera foi posicionada na região central superior da dianteira do veículo, de forma a reproduzir o campo de visão de um sistema embarcado automotivo.
A captura das imagens foi realizada por meio de um programa desenvolvido em Python, utilizando a biblioteca OpenCV, configurado para registrar uma imagem a cada segundo. O procedimento foi conduzido no campus de São Caetano do Sul, abrangendo diferentes trajetos e condições de iluminação e clima. O objetivo dessa etapa foi compor um *dataset* diversificado, capaz de representar situações reais de tráfego e auxiliar no treinamento das redes neurais subsequentes.

Figura 1 – Imagens com diferentes condições de iluminação.

A Figura 1(a) foi retirada durante um dia nublado, é possível observar que possui uma iluminação mais uniforme, já a em (b) foi aquisitada durante um dia com bastante iluminação solar, resultando em sombra sobre os objetos e na rua, dificultando assim o aprendizado do modelo. Por fim, em (c) foi adquirida durante a noite, resultando uma diferente luminosidade da rua e dos objetos a frente do veículo.

Etapa 2 – Rotulação e treinamento do modelo tipo YOLO.

Após a aquisição das imagens em diferentes condições de iluminação no campus do Instituto Mauá de Tecnologia, iniciou-se a etapa de pré-processamento e rotulação. Primeiramente, todas as imagens foram redimensionadas para a resolução de 640×640 pixels, a fim de atender ao formato de entrada exigido pelos modelos do tipo YOLO durante o treinamento.
Para a rotulação, foi utilizada a plataforma Roboflow (DWYER et al., 2025), uma ferramenta Web que permite realizar anotações manuais e gerar automaticamente os arquivos de rótulos compatíveis com o formato YOLO. Na primeira fase da rotulação, foram identificados objetos como carros,

faixas de pedestre e lombadas, enquanto na segunda fase foi realizada a segmentação das áreas correspondentes à rua.
Adicionalmente, foram incorporadas imagens obtidas a partir da Roboflow Universe, contendo exemplos previamente segmentados de vias urbanas, com o objetivo de aumentar a diversidade do conjunto de dados. Ao final do processo, o *dataset* total utilizado contou com aproximadamente 8.000 imagens, que foram divididas em 80% para treinamento de 20% para validação, abrangendo uma ampla variedade de cenários e condições ambientais.

Figura 2 – Exemplo de rotulação de imagem utilizando RoboFlow.

Na Figura 2, é possível observar que, na rotulação (a), foram identificados apenas os objetos à frente do veículo, enquanto na rotulação (b), foi segmentada apenas a rua. Dessa forma, foi possível treinar dois modelos do tipo YOLO: um voltado para a detecção de objetos e outro para a segmentação da via.
Após a etapa de rotulação, utilizando Python e a biblioteca Ultralytics, foram realizados os treinamentos dos dois modelos. Em ambos os casos, foram empregadas aproximadamente 500 épocas e uma taxa de aprendizado (*learning rate*) de 0.0001, visando alcançar uma boa convergência e desempenho nos resultados.

Etapa 3 – Destilação do modelo Depht-Pro.

Com os modelos da YOLO já treinados, a etapa seguinte consistiu na destilação do modelo Depth-Pro que apresenta aproximadamente 500 milhões de parâmetros. Devido à sua alta complexidade, esse tipo de modelo demanda um elevado poder computacional para operar em tempo real, algo inviável na maioria dos sistemas embarcados. Como alternativa, foi aplicada a técnica de destilação de modelos (*model distillation*), que visa transferir o conhecimento de um modelo complexo para outro modelo mais leve, com menor número de parâmetros e maior eficiência computacional.
Nesse processo, o modelo original é denominado modelo professor, responsável por gerar os dados de treinamento utilizados pelo modelo aluno, cuja arquitetura é projetada especificamente para execução embarcada. Para a geração desses dados, foram reutilizadas as mesmas imagens empregadas no treinamento da YOLO, porém sem as anotações de rotulação. As saídas do modelo professor foram salvas em arquivos do tipo NumPy (.npy), juntamente com suas respectivas imagens, formando o novo *dataset* supervisionado.
O modelo aluno foi implementado em Tensorflow utilizando uma rede neural convolucional (CNN) com arquitetura do tipo ResU-Net (LI et al., 2022), que combina blocos residuais e mecanismos de

atenção. Essa configuração melhora o desempenho em relação às arquiteturas Encoder–Decoder tradicionais, permitindo uma extração de características mais eficiente, mesmo com conjuntos de dados reduzidos. Além disso, para aumentar a robustez e reduzir o *overfitting*, foram aplicadas técnicas de *data augmentation*, que consistem em modificar artificialmente as imagens de treinamento, como alterar contraste, brilho e iluminação, de modo a simular diferentes condições visuais e ampliar a diversidade dos dados.

O modelo aluno foi treinado utilizando um *dataset* composto por aproximadamente 7.000 imagens, mantendo-se o padrão de resolução adotado no treinamento da YOLO (640 × 640 pixels) para garantir uniformidade no processamento e na aquisição das imagens. O conjunto de dados foi dividido em 80% para treinamento e 20% para validação, permitindo uma avaliação equilibrada do desempenho do modelo. Durante o processo de treinamento, foram utilizados 1.000 épocas e uma taxa de aprendizado (learning rate) de 0,0001; esses parâmetros foram definidos com o objetivo de otimizar a convergência e minimizar o erro de generalização.

Além disso, para o treinamento, foi implementada uma função de custo customizada, desenvolvida com o objetivo de combinar a precisão na estimativa de profundidade com a coerência estatística das distribuições de profundidade preditas. Essa função de perda integra dois componentes principais: o *Mean Squared Error* (MEAN SQUARED ERROR, 2025), que mede o erro médio quadrático entre o mapa de profundidade real e o predito, e a divergência de *Kullback–Leibler* (KL DIVERGENCE, 2025), que compara as distribuições de probabilidade das profundidades entre o valor real e o estimado.

Para o cálculo dessas distribuições, cada imagem é convertida em um histograma suavizado de profundidades, normalizado por meio de uma técnica de *"soft histogram"* com kernel gaussiano. Dessa forma, o modelo não apenas aprende a prever os valores absolutos de profundidade, mas também a preservar a distribuição relativa das distâncias em cada cena, o que contribui para uma representação mais consistente e estável do espaço tridimensional. A função de custo final é expressa como uma combinação ponderada dessas duas métricas, definida por:

Equação 1 – Função de custo.

$$L = \alpha \cdot \mathrm{MSE}\left(y_{\mathrm{true}}, y_{\mathrm{pred}}\right) + \beta \cdot D_{KL}\left(P(y_{\mathrm{true}}) \parallel P(y_{\mathrm{pred}})\right)$$

onde α e β são hiperparâmetros que controlam a contribuição de cada termo. Essa formulação permite equilibrar o aprendizado entre precisão numérica e coerência estrutural, resultando em um modelo aluno mais robusto e adaptado às restrições de *hardware* embarcado.

Etapa 4 – Integração das redes neurais.

Após a etapa de destilação de modelos, foi utilizada a máscara de segmentação da rua gerada pelo modelo YOLO para a extração dos contornos da via. Inicialmente, os pontos pertencentes à borda da segmentação foram filtrados, de modo que apenas os pixels de limite da região segmentada fossem mantidos. Entretanto, observou-se que, para um mesmo valor de coordenada X, havia múltiplos valores de Y, o que representava diferentes pontos verticais sobre uma mesma coluna da imagem. Para contornar esse problema, aplicou-se uma filtragem adicional, selecionando apenas o ponto correspondente ao maior valor de Y para cada X, garantindo assim que apenas o contorno inferior da segmentação fosse preservado, representando de forma consistente o limite visível da rua.

Em seguida, a mesma imagem foi processada pelo modelo aluno, desenvolvido a partir da destilação de modelos, para gerar o mapa de profundidade correspondente. A partir desse mapa, foi possível associar a cada ponto filtrado da saída da YOLO sua respectiva distância em relação ao veículo, obtendo assim uma estimativa da distância dos obstáculos pixel a pixel presentes à frente do veículo autônomo.

Etapa 5 – Implementação no *hardware* embarcado.

Após o desenvolvimento do algoritmo de determinação do espaço livre à frente do veículo, tornou-se necessário selecionar um *hardware* capaz de executar o sistema de forma eficiente. Após uma análise das opções disponíveis no mercado, optou-se pela NVIDIA JETSON AGX ORIN (32 GB), devido ao seu elevado poder computacional aliado a dimensões físicas compactas e baixo consumo energético (cerca de 40 W em operação sem limitação de potência). Essas características tornam a plataforma especialmente adequada para aplicações embarcadas em veículos autônomos, nas quais há restrições de espaço e eficiência energética é um fator crítico.

Figura 3 – Especificações NVIDIA JETSON AGX ORIN.

	Jetson AGX Orin series				Jetson Orin NX series		Jetson Orin Nano series		
	Kit para Desenvolvedor Jetson AGX Orin	Jetson AGX Orin 64GB	Jetson AGX Orin Industrial	Jetson AGX Orin 32GB	Jetson Orin NX 16GB	Jetson Orin NX 8GB	Super Kit para Desenvolvedor Jetson Orin Nano	Jetson Orin Nano 8GB	Jetson Orin Nano 4GB
Desempenho de IA	275 TOPS		248 TOPS	200 TOPS	157 TOPS	117 TOPS	67 TOPS	67 TOPS	34 TOPS
GPU	GPU de arquitetura NVIDIA Ampere de 2.048 núcleos com 64 Núcleos Tensor			GPU de arquitetura NVIDIA Ampere de 1.792 núcleos com 56 Núcleos Tensor	GPU de arquitetura NVIDIA Ampere de 1.024 núcleos com 32 Núcleos tensor		GPU de arquitetura NVIDIA Ampere de 1.024 núcleos com 32 Núcleos tensor		GPU de arquitetura NVIDIA Ampere de 512 núcleos com 16 Núcleos Tensor
Frequência Máxima da GPU	1.3 GHz		1.2 GHz	930 MHz	1173MHz	1173MHz	1020MHz	1020MHz	1020MHz
CPU	CPU ARM® Cortex®-A78AE v8.2 de 12 núcleos de 64 bits L2 de 3 MB + L3 de 6 MB			CPU ARM® Cortex®-A78AE v8.2 de 8 núcleos de 64 bits L2 de 2 MB + L3 de 4 MB	CPU ARM® Cortex®-A78AE v8.2 de 8 núcleos de 64 bits L2 de 2 MB + L3 de 4 MB	CPU ARM® Cortex®-A78AE v8.2 de 6 núcleos de 64 bits L2 de 1.5 MB + L3 de 4 MB	CPU ARM® Cortex®-A78AE v8.2 de 6 núcleos de 64 bits L2 de 1.5 MB + L3 de 4 MB		
Frequência Máxima da CPU	2.2 GHz		2.0 GHz	2.2 GHz	2 GHz		1.7 GHz	1.7 GHz	1.7 GHz

Na figura 3, é possivel obrservar que a NIVIDA JETSON AGX ORIN conta com uma capacidade computacional de até 200 TOPS (*Tera Operations Per Second*), ou seja, é capaz de realizar até 200 trilhões de operações por segundo. Esse desempenho é especialmente importante em aplicações que envolvem redes neurais convolucionais (CNNs), pois tais modelos demandam uma grande quantidade de operações matriciais e convoluções em tempo real.

Após a definição do hardware, foi necessário realizar a conversão do modelo aluno para formatos mais adequados ao ambiente embarcado, garantindo o máximo desempenho possível. Inicialmente, o modelo foi convertido do formato TensorFlow para o formato ONNX (*Open Neural Network Exchange*), um padrão aberto que facilita a interoperabilidade entre diferentes frameworks de redes neurais. Essa conversão permite maior flexibilidade e compatibilidade com outras ferramentas de otimização.

Em seguida, o modelo ONNX foi convertido para o formato TensorRT, que é uma biblioteca de otimização desenvolvida pela NVIDIA para inferência de redes neurais em GPUs. O TensorRT realiza otimizações como fusão de camadas, quantização de pesos e execução em precisão mista (FP16/INT8), resultando em uma redução significativa no tempo de inferência e melhor aproveitamento da GPU da Jetson Orin. Dessa forma, o sistema pôde operar de forma mais eficiente e com desempenho em tempo real, fundamental para aplicações embarcadas em veículos autônomos.

Com a otimização e implantação no *hardware* embarcado, o sistema proposto atingiu o desempenho necessário para operação em tempo real, consolidando a viabilidade da solução desenvolvida para aplicação em veículos autônomos.

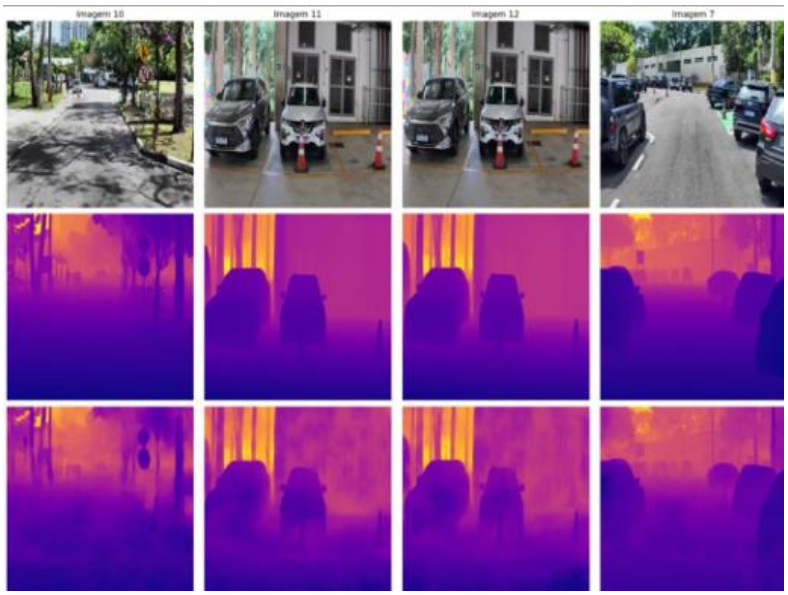
Resultados e Discussão

Após o treinamento do modelo YOLO e do modelo aluno, bem como a execução do algoritmo de união entre ambos e a realização dos testes experimentais correspondentes, foram obtidos os resultados apresentados a seguir, os quais permitem avaliar o desempenho e a eficiência do sistema.

Figura 4 – Segmentação da Rua.

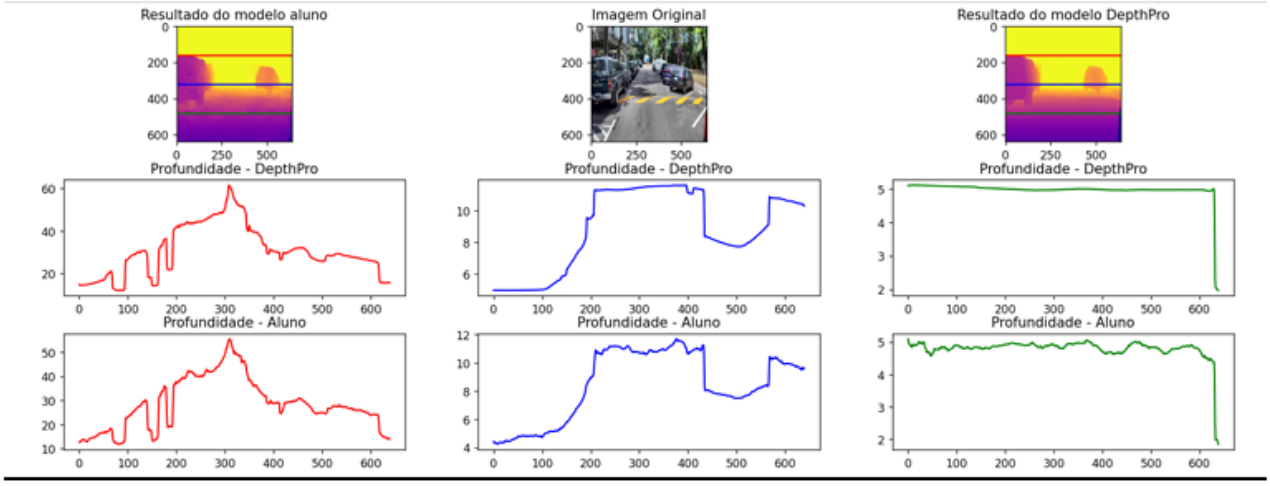
Na Figura 4, é possível observar o resultado do treinamento do modelo YOLO para à segmentação de rua. Em média, o modelo apresentou uma exatidão de 95%, com um tempo médio de execução de 0,05 segundos por imagem no modo FP16, o que demonstra sua viabilidade para aplicações em tempo real.
Já na Figura 5, são apresentados os resultados comparativos entre o modelo aluno (ResU-Net) e o modelo professor (Depth-Pro). As imagens da primeira linha correspondem aos exemplos originais, as da segunda linha representam os mapas de profundidade gerados pelo modelo professor e, por fim, as da terceira linha mostram os resultados produzidos pelo modelo aluno.
Após o processo de destilação, o modelo aluno desenvolvido passou a conter aproximadamente 35 milhões de parâmetros, em contraste com os cerca de 500 milhões de parâmetros do modelo original (Depth-Pro). Essa redução de aproximadamente 93% na quantidade total de parâmetros resultou em uma diminuição significativa do custo computacional, permitindo sua execução em *hardware* embarcado com maior eficiência, além de reduzir o tempo de inferência de aproximadamente 180 segundos para cerca de 100 ms.

Figura 5 - Comparação de saídas do modelo aluno e Depth-Pro.



Na Figura 6, é possível comparar as distâncias estimadas pelos modelos aluno e professor (Depth-Pro). As distâncias foram obtidas ao longo de algumas linhas horizontais da imagem, permitindo uma análise detalhada da correspondência entre as previsões. Observa-se que os valores produzidos pelo modelo aluno estão bastante próximos aos obtidos pelo modelo Depth-Pro, evidenciando uma boa transferência de conhecimento durante o processo de destilação. A principal diferença identificada é a presença de um ruído ligeiramente maior nas previsões do modelo aluno, o que é esperado em função de sua menor complexidade e número de parâmetros.

Figura 6 – Comparação das distancias entre modelo aluno e Deph-Pro.



Na Figura 7, é possível observar as etapas realizadas para a estimativa das distâncias. Inicialmente, são identificadas as bordas da segmentação da rua. Em seguida, realiza-se um processo de filtragem, mantendo apenas os pontos com coordenadas X associadas aos maiores valores de Y, correspondentes à borda inferior da região segmentada. Por fim, esses pontos são fornecidos ao

modelo aluno, que retorna os valores estimados de distância para cada um deles. Dessa forma, é possível determinar a variação das distâncias dos objetos ao longo da imagem.

Figura 7 – Algoritmo de determinação das distancias.

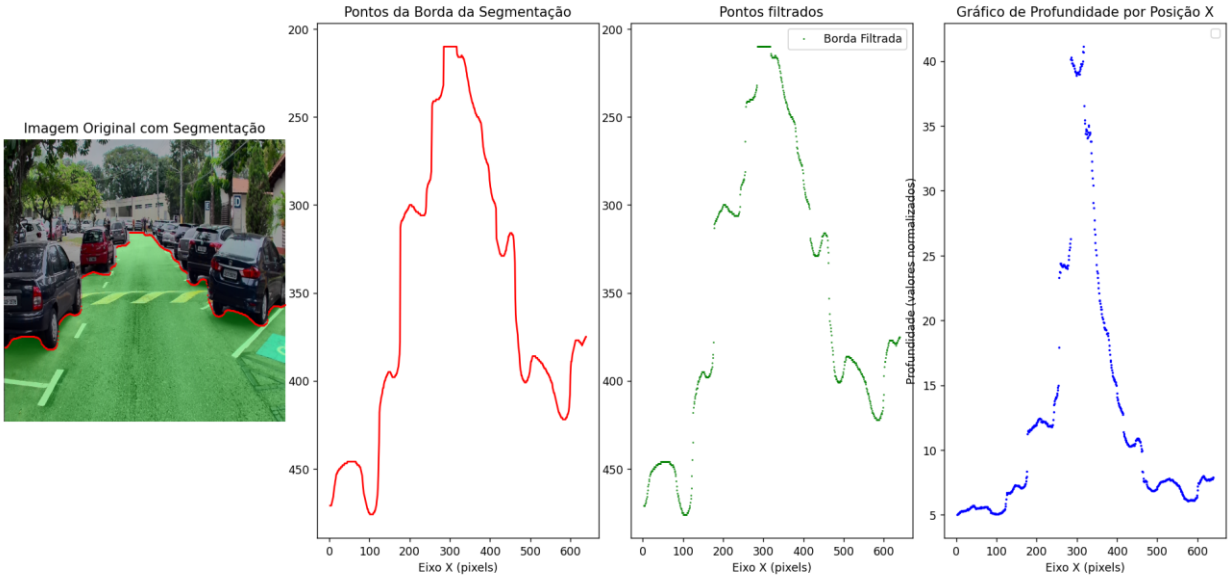
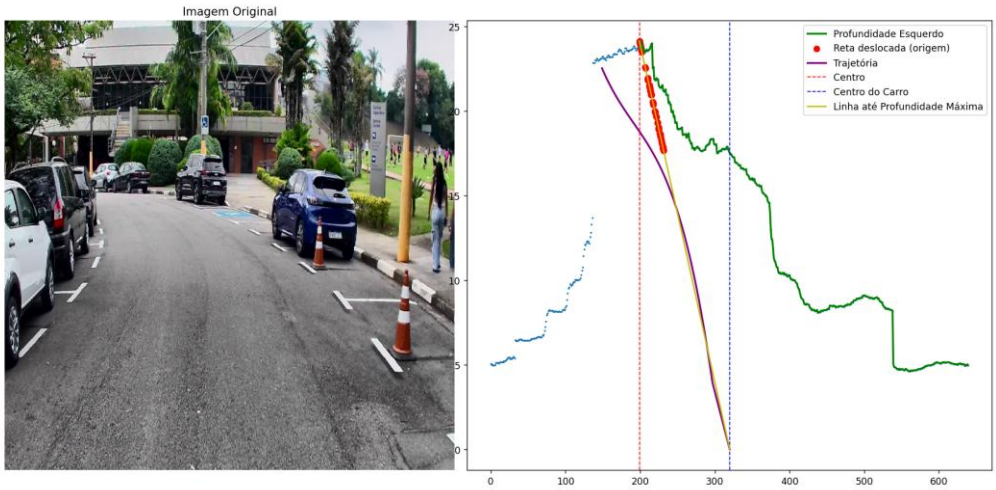


Figura 8 – Protótipo de algoritmo de planejamento de trajetória.



Na Figura 8, é apresentado um protótipo de algoritmo de trajetória desenvolvido para estimar o ângulo de direção do volante de um veículo autônomo, com base na segmentação da rua e nas distâncias estimadas pelo modelo aluno.

O algoritmo inicia traçando uma reta que parte do centro da imagem representando o centro do veículo até o ponto de maior distância calculada pelo modelo aluno. Antes disso, os pontos de saída do modelo são filtrados para remoção de outliers, garantindo maior estabilidade nas estimativas.

Em seguida, os pontos dessa reta são comparados com os pontos das distâncias do lado esquerdo e direito. Caso a distância entre eles seja menor que o limite definido pelo algoritmo, os pontos são ajustados lateralmente para o lado que apresentar maior distância. Se ambos os lados estiverem abaixo do limite, o ponto é reposicionado na média das duas distâncias.

Após o ajuste, é aplicada uma curva *spline* para suavizar o caminho gerado, resultando em uma trajetória contínua e realista. Dessa forma, o algoritmo é capaz de prever a trajetória do veículo por

aproximadamente 20 metros à frente, fornecendo uma base robusta para o controle direcional autônomo.

Conclusões

Esse trabalho de iniciação científica apresentou o desenvolvimento de um sistema embarcado para a determinação do espaço livre à frente de um veículo autônomo, utilizando técnicas de visão computacional e inteligência artificial como alternativa de menor custo em relação a sensores tradicionais, como o LiDAR. Por meio da combinação de um modelo de segmentação baseado no YOLO e de um modelo de estimativa de profundidade derivado da destilação do Depth-Pro, foi possível integrar a percepção do ambiente em um sistema unificado capaz de operar em tempo real.
Os resultados obtidos demonstram a eficiência do processo de destilação de modelos na transferência de conhecimento entre arquiteturas de diferentes complexidades. O modelo aluno, com aproximadamente 35 milhões de parâmetros, apresentou um desempenho computacional significativamente superior, reduzindo o tempo médio de inferência de 170 segundos por imagem (modelo Depth-Pro) para 100 milissegundos por imagem (modelo aluno em INT8), sem perdas expressivas de precisão. Essa otimização possibilitou a execução do modelo em hardware embarcado, como a NVIDIA Jetson AGX Orin, de forma estável e com consumo energético reduzido.
Além disso, o algoritmo de trajetória desenvolvido representa um protótipo inicial voltado à previsão do caminho do veículo com base na segmentação da via e nas distâncias estimadas pelo modelo aluno. Embora os resultados obtidos tenham sido satisfatórios, demonstrando a capacidade de prever trajetórias de até 20 metros à frente, o algoritmo ainda apresenta grande potencial de evolução. Melhorias futuras podem incluir o refinamento dos critérios de decisão lateral, a integração de múltiplos sensores e a aplicação de técnicas mais avançadas de planejamento e controle de trajetória.
Dessa forma, o sistema proposto se mostra uma solução promissora para aplicações em veículos autônomos, conciliando baixo custo, eficiência computacional e precisão adequada, abrindo caminho para o desenvolvimento de sistemas de navegação autônoma mais acessíveis e escaláveis.

Referências Bibliográficas

AFSHAR, Mehrnaz Farokhnejad; SHIRMOHAMMADI, Zahra; GHAFOURIAN GHAHRAMANI, Seyyed Amir Ali; NOORPARVAR, Azadeh; HEMMATYAR, Ali Mohammad Afshin. An Efficient Approach to Monocular Depth Estimation for Autonomous Vehicle Perception Systems. Sustainability, v. 15, n. 11, 8897, 2023. DOI: 10.3390/su15118897.

BOCHKOWSKI et al. (2024). BOCHKOVSKI, Aleksei; DELAUNOY, Amaël; GERMAIN, Hugo; SANTOS, Marcel; ZHOU, Yichao; RICHTER, Stephan R.; KOLTUN, Vladlen. Depth Pro: Sharp Monocular Metric Depth in Less Than a Second. Disponível em: arXiv, 2 out. 2024. arXiv:2410.02073 [cs.CV].

DWYER et al. (2025). DWYER, B.; NELSON, J.; HANSEN, T. et al. Roboflow (Versão 1.0). Computer Vision. Disponível em: https://roboflow.com. Acesso em: 22 ago. 2025.

Gatti e Cabral (2024). D. E. GATTI, E.L.L. CABRAL. Sistemas para percepção do espaço livre à frente de um veículo e cálculo da distância de seus limites.

KOTTHAPALLI et al. (2025). KOTTHAPALLI, Manikanta; RAVIPATI, Deepika; BHATIA, Reshma. YOLOv1 to YOLOv11: A Comprehensive Survey of Real-Time Object Detection Innovations and Challenges. [S. l.: s. n.], 4 ago. 2025. arXiv preprint arXiv:2508.02067. Disponível em: arXiv. Acesso em: 22 ago. 2025.

KLDIVERGENCE. *tf.keras.losses.KLDivergence | TensorFlow v2.16.1*. Disponível em: https://www.tensorflow.org/api_docs/python/tf/keras/losses/KLDivergence. Acesso em: 12 nov. 2025.

LI et al. (2022). LI, ZAN; ZHANG, HONG; LI, ZHENGZHEN; REN, ZUYUE. Residual-Attention UNet++: A Nested Residual-Attention U-Net for Medical Image Segmentation. Applied Sciences, Basel, v. 12, n. 14, p. 7149, 2022. DOI: 10.3390/app12147149.

LI, Guofa; SONG, Xuedong; GAO, Ruipeng; TAO, Dan. Monocular Depth Estimation for 3D Map Construction at Underground Parking Structures. *Electronics*, v. 12, n. 11, p. 2390, 2023. DOI: 10.3390/electronics12112390.

NVIDIA Corporation. NVIDIA Jetson AGX Orin Technical Overview. 2023. Disponível em: https://developer.nvidia.com/embedded/jetson-orin. Acesso em: 17 ago. 2025.

RAVI et al. (2024). RAVI, Nikhila; GABEUR, Valentin; HU, Yuan-Ting; HU, Ronghang; RYALLI, Chaitanya; MA, Tengyu; KHEDR, Haitham; RADLE, Roman; ROLLAND, Chloe; GUSTAFSON, Laura; MINTUN, Eric; PAN, Junting; ALWALA, Kalyan Vasudev; CARION, Nicolas; WU, Chao-Yuan; GIRSHICK, Ross; DOLLÁR, Piotr; FEICHTENHOFER, Christoph. SAM 2: Segment Anything in Images and Videos. arXiv preprint, 2024. Disponível em: arXiv:2408.00714.

MEAN SQUARED ERROR. *tf.keras.losses.MSE | TensorFlow v2.16.1*. Disponível em: https://www.tensorflow.org/api_docs/python/tf/keras/losses/MSE. Acesso em: 12 nov. 2025.

XUE, Feng; ZHUO, Guirong; HUANG, Ziyuan; FU, Wufei; WU, Zhuoyue; ANG JR., Marcelo H. Toward Hierarchical Self-Supervised Monocular Absolute Depth Estimation for Autonomous Driving Applications. *arXiv*, 2020. Disponível em: https://arxiv.org/abs/2004.05560. Acesso em: 15 nov. 2025.

YADAV, Shreedhar Prasad; FERREIRA, Pedro; PIECHULEK, Andreas; OLIVER, Michael; BOYER, Kim; REVERDY, Paul; OSTEIKOETXEA, Ander; SHIM, Jae Eun. *LiDAR-Based Perception for Autonomous Vehicles: A Survey*. Disponível em: arXiv, 12 jan. 2023. arXiv:2301.05061 [cs.RO].