SENSYNC: Instrumentalização de Plataforma Veicular para Testes de Sincronização Multissensor e Geração de Datasets

Alexsandro Ferreira Coelho, Abel G. Silva-Filho, Alef Gabryel Lorenco da Costa, Lucas Alves Barbosa
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
{afc7, agsf, aglc, lab10}@cin.ufpe.br

***Abstract*—Multimodal perception systems used in autonomous vehicles rely on precise temporal synchronization between sensors such as cameras, radars, and LiDAR. Small misalignments of only a few milliseconds can compromise perception and decision-making in dynamic scenarios. This work presents the SENSYNC architecture, an entirely software-based solution built on ROS 2 for acquiring and synchronizing sensors in a Jeep Renegade equipped with three cameras, three FMCW radars, and a VLP-16 LiDAR. The platform is evaluated under real urban traffic conditions, quantifying temporal offsets between sensors and the robustness of multimodal recording. The results show average offsets close to 1 ms for radar/camera pairs and typically below 25 ms for radar/LiDAR pairs, values suitable for multimodal fusion in urban environments. Approximately 12 GB of multimodal data were collected, with an average recording rate of 29 MB/s (1.74 GB/min). These results confirm the feasibility of achieving fully software-based synchronization in real vehicles.**

***Index Terms*—Synchronization, ROS 2, Sensors, Autonomous Vehicles, Multimodal Perception**

I. Introdução

Nas últimas décadas, veículos autônomos, juntamente com os *Advanced Driver Assistance Systems* (ADAS) e *Autonomous Driving Systems* (ADS), transformaram o setor automotivo, oferecendo maior segurança, eficiência e conforto. Para o correto funcionamento, esses sistemas dependem de uma percepção espacial precisa, obtida pela integração de múltiplos sensores como LiDAR, câmeras, radares, GPS e IMUs, cada um fornecendo informações complementares para a construção de uma representação consistente do ambiente ao redor do veiculo [1]. Entretanto, a simples fusão desses sensores não é suficiente se não houver alinhamento temporal adequado. A literatura mostra que atrasos de apenas 50 ms podem resultar em erros superiores a $1{,}4$ m na estimativa da posição de objetos a 100 km/h [2], evidenciando a sincronização temporal como um requisito crítico em veículos autônomos.

Nesse contexto, o *Robot Operating System 2* (ROS 2) tem sido amplamente adotado como *middleware* para integração multimodal. Baseado no protocolo *Data Distribution Service* (DDS), o ROS 2 fornece mecanismos de comunicação assíncrona, políticas configuráveis de Qualidade de Serviço (QoS) e carimbos de tempo consistentes nos cabeçalhos das mensagens [3]. Além disso, a biblioteca `message_filters` disponibiliza o filtro *ApproximateTime*, que permite agrupamento de mensagens de sensores distintos com base na proximidade temporal entre seus *timestamps*. Essa combinação torna possível implementar sincronização exclusivamente em *software*, sem recorrer a módulos externos de temporização ou interfaces dedicadas como PTP (IEEE 1588).

Apesar da existência de conjuntos de dados consolidados, como nuScenes e KITTI [4], há escassez de *datasets* multimodais adquiridos em contextos brasileiros, cujas condições urbanas, iluminação variável, vias estreitas, tráfego denso e sinalização heterogênea, diferem substancialmente dos cenários presentes nos principais bancos de dados globais [5]. Assim, arquiteturas reprodutíveis para aquisição sincronizada de dados em veículos reais são de grande relevância para a comunidade científica nacional.

Neste trabalho, propõe-se uma arquitetura de sincronização temporal baseada exclusivamente em *software*, utilizando ROS 2 Humble, aplicada a um Jeep Renegade equipado com câmeras, radares e um LiDAR. A principal contribuição é mostrar que, mesmo sem temporização externa, é possível alcançar sincronização consistente em ambiente real e gerar um *dataset* multimodal.

II. Trabalhos Relacionados

Em [6], o EverySync foi desenvolvido como um sistema de sincronização temporal baseado em *hardware* para câmeras, LiDARs, IMUs e GNSS/RTK, alcançando precisão inferior a 1 ms. Comparado ao VersaVIS, reduziu em até 45% o desvio temporal médio e melhorou 38% a precisão da trajetória reconstruída. Em [7], o *middleware SmartData* integrou simuladores de sistemas autônomos com GNSS, atingindo desvio médio de 6 ms entre sensores, evidenciando a importância da sincronização temporal.

A proposta de [8] consistiu em uma solução de *software* utilizando o ROS para sincronização de câmeras ZED, LiDAR Velodyne e GNSS/RTK, com erro abaixo de 30 ms e precisão de 1 cm, mostrando eficácia a baixo custo. Já [4] desenvolveu o *dataset* nuScenes, com câmeras, LiDARs, radares, GPS e IMUs sincronizados por disparos síncronos (erro máximo 1

ms), melhorando a qualidade dos dados para algoritmos de percepção em veículos autônomos. Este trabalho se diferencia ao implementar uma lógica adaptativa de sincronização utilizando o ROS 2, baseada em filas temporais dinâmicas e limites flexíveis para diferenças de *timestamps*, otimizando o agrupamento de mensagens periódicas.

III. Implementação da Arquitetura

Esta seção apresenta a arquitetura de aquisição e sincronização temporal dos sensores multimodais, descrevendo a infraestrutura embarcada, o fluxo de comunicação e o mecanismo de sincronização no ROS 2.

A. Infraestrutura Embarcada

Para validar a proposta, um Jeep Renegade foi equipado com três câmeras Intelbras VHDM 3105 G3, três radares Continental ARS408-21 e um LiDAR Velodyne VLP-16. Todos os sensores foram integrados à NVIDIA Jetson Orin, responsável pela centralização, sincronização e armazenamento dos dados, utilizando comunicação via *Ethernet*. A Jetson Orin foi escolhida pelo suporte à aceleração por GPU e pela compatibilidade com o ROS 2. A Fig. 1 apresenta o setup instalado na parte traseira do veículo.

Fig. 1. Setup instalado no Jeep Renegade

B. Aquisição de Dados

A aquisição de dados foi estruturada de forma modular na arquitetura de nós do ROS 2, em que cada nó opera como um processo independente em Python ou C++. Os nós de leitura dos sensores se comunicam entre si por meio de tópicos, serviços ou ações da DDS, atuando como *publishers* e *subscribers* para garantir a captura eficiente dos dados de cada sensor conforme Fig. 2.

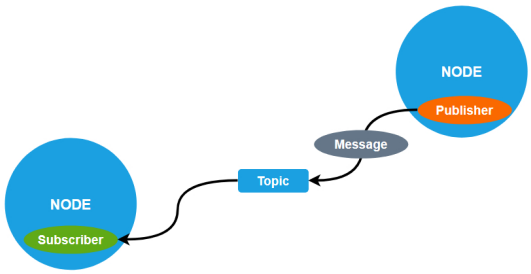

Fig. 2. Representação simplificada da comunicação entre nós no ROS 2

1) Câmeras: As câmeras VHDM 3105 G3 foram configuradas para 1280×720 pixels, ajustável até 30 FPS e suporte a *Wide Dynamic Range* (WDR), garantindo bom equilíbrio entre qualidade de imagem, largura de banda e robustez em cenários urbanos. Elas foram instaladas na parte superior do veículo (Fig. 3), em suportes metálicos com isolamento antivibração, preservando o alinhamento óptico e proporcionando cobertura visual de até 180°, com sobreposição parcial entre campos de visão. As imagens são publicadas no ROS 2 por meio de um nó baseado no pacote `image_transport`, que realiza a compressão em formato JPEG.

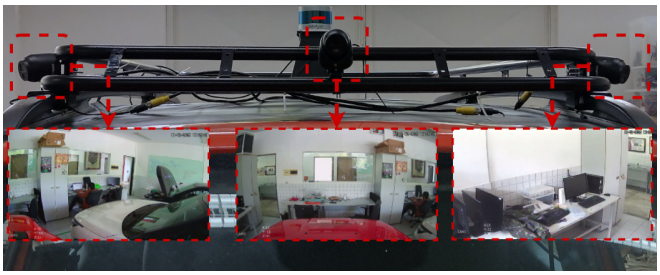

Fig. 3. Disposição das câmeras VHDM 3105 G3 no veículo

2) Radares: O subsistema de detecção por radar, utiliza três ARS-408 21, operando em 77 GHz, amplamente empregados em sistemas ADAS pela capacidade de detectar veículos, pedestres e obstáculos sob diferentes condições climáticas. Para garantir cobertura eficiente, os dispositivos foram instalados em suportes 3D nas regiões frontal e laterais do veículo, com alinhamento horizontal.

Os sensores transmitem dados nos modos *objeto* e *cluster* via barramento CAN. A interface Vector VN8912 atua como *gateway*, convertendo as mensagens CAN em pacotes TCP e enviando-as via *sockets* para a Jetson Orin, onde são convertidas para o formato `PointCloud2` no ROS 2. A Fig. 4 mostra a visualização dos *clusters* no *software* RadarVisual, utilizado durante os testes. Além da aquisição dos pacotes, a infraestrutura registra taxa de mensagens, variações de latência e perdas no barramento.

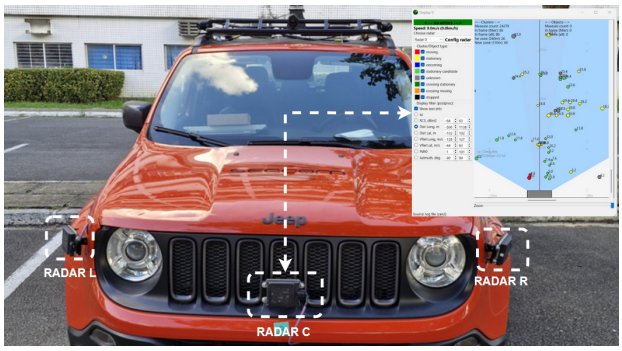

Fig. 4. Visualização dos clusters de radar no software RadarVisual

3) LiDAR: O LiDAR Velodyne VLP-16 foi utilizado para aquisição tridimensional do entorno, operando a aproximadamente 20 Hz, com 16 canais de varredura, taxa de até 300.000

pontos por segundo e alcance de 100 m. Seu campo de visão de 360° (horizontal) e 20° (vertical) fornece ampla cobertura espacial para percepção multimodal. O sensor foi montado em suporte impresso em 3D com isolamento antivibração, garantindo estabilidade da varredura e alinhamento com as câmeras.

A comunicação com a Jetson Orin ocorre via *Ethernet*, com pacotes UDP de aproximadamente 1.2 – 1.4 kB cada, garantindo largura de banda suficiente para transmissão contínua. No ROS 2, o pacote `velodyne_driver` interpreta os pacotes e os converte em mensagens `PointCloud2`, integrando o LiDAR ao restante da arquitetura.

C. Estratégia de Sincronização

A lógica de sincronização foi integrada à infraestrutura de software, utilizando recursos nativos do ROS 2 para alinhar fluxos multimodais. Para a comunicação em tópicos, são usadas *sensor messages*, que possuem campos específicos úteis para esse propósito: `stamp`, marcando o tempo referente àquela mensagem, e `frame_id`, onde é possível colocar uma *string* (texto) para identificação. O radar, com frequência mais estável em 13 Hz, foi adotado como referência temporal para os demais sensores.

1) Agrupamento Temporal: Utiliza-se o componente de filtro de tempo aproximado (ApproximateTime) do próprio *framework* ROS 2. Ele emprega um *subscriber* que agrupa mensagens, retornando em um *callback* aquelas que possuem tempos próximos, o que indica que pertencem ao mesmo grupo. Essa estratégia resultou em atrasos máximos de apenas algumas dezenas de milissegundos, viabilizando análises consistentes em percepção multimodal sem exigir *hardware* adicional de temporização.

2) Marcação Semântica: Para organizar os dados durante o armazenamento, a infraestrutura embarcada realiza a marcação semântica das mensagens. Para identificar os conjuntos sincronizados, insere-se no campo `frame_id` uma *string* identificadora seguindo o padrão `GRUPO:ORDEM`. O componente "GRUPO" refere-se ao radar utilizado como referência, enquanto "ORDEM" é um valor incremental que estabelece uma sequência temporal. A cada nova varredura detectada, o sistema aciona o *callback* que é gerado pelo filtro de tempo aproximado e copia o identificador do radar para os demais sensores temporalmente alinhados.

As câmeras seguem um agrupamento direto, com cada unidade associada a um radar específico. Já o LiDAR é sincronizado com todos os radares, copiando os grupos de todas as mensagens de radar que coincidirem com sua varredura. A lógica do programa pode ser vista na Figura 5. Os nós responsáveis por coletar os dados dos sensores publicam as mensagens em tópicos; outros nós sincronizadores agrupam essas mensagens e as republicam com seus identificadores, para depois serem armazenadas em um banco de dados SQL e extraídas para o dataset.

Todas as mensagens dos radares são preservadas, mesmo aquelas que não se alinham com outros sensores, assegurando a completude do dataset. A Fig. 5 ilustra o fluxo de sincronização e marcação semântica entre os sensores na arquitetura SenSync, destacando os *nós* de agrupamento, os canais de republicação e o armazenamento centralizado em banco SQL.

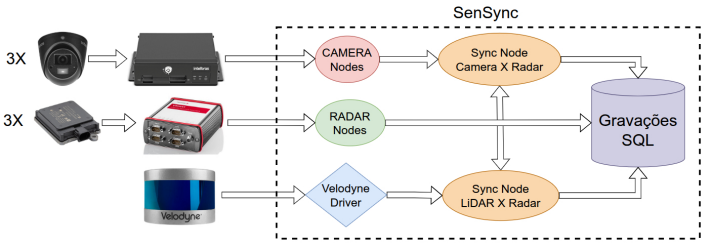

Fig. 5. Fluxo de sincronização e marcação semântica na arquitetura SenSync

IV. Resultados

Nesta seção são apresentados os resultados obtidos com a plataforma SENSYNC, destacando a avaliação da sincronização entre sensores, o desempenho da infraestrutura.

A. Ambiente de Coleta

Os testes iniciais foram realizados nas dependências da Universidade Federal de Pernambuco (UFPE) e em suas proximidades, sendo posteriormente estendidos para vias urbanas do Recife-PE. Essa abordagem garantiu a coleta de dados em diferentes tipos de ambientes, aumentando a diversidade dos cenários avaliados [9].

B. Taxa de Aquisição e Volume de Dados

Durante as gravações, a plataforma manteve estabilidade mesmo com múltiplos fluxos simultâneos. No total, foram coletados aproximadamente 12.4 GB de dados, valor correspondente à soma dos volumes individuais por sensor conforme são apresentados na Tab. I. A taxa média de gravação observada foi de cerca de 29 MB/s, o que corresponde a aproximadamente 1.74 GB por minuto.

TABLE I
Frequência de aquisição e volume de dados por sensor

Sensor	Frequência	Volume	Formato
Câmeras	até 30 Hz	10.4 GB Total	JPEG
Radares	∼13 Hz	18 MB Total	PCD
LiDAR	até 20 Hz	2 GB Total	PCD

Esses valores reforçam a capacidade da infraestrutura de rede e armazenamento da SENSYNC para lidar com cargas multimodais contínuas.

C. Análise de Defasagem Temporal

A avaliação da sincronização temporal foi realizada excluindo-se os dados da fase de inicialização, visando eliminar *outliers* decorrentes da estabilização do sistema. A análise consistiu em medir a diferença temporal entre as capturas do radar (sensor de referência) e as das câmeras e do LiDAR.

A defasagem da câmera, apresentada na Figura 6(b), demonstrou uma distribuição normal, com média de 0.8 ms e desvio padrão de 11.12 ms. Já a do LiDAR apresentou uma distribuição mais uniforme, conforme mostrado na Figura 6(a), com limite absoluto de defasagem em 25 ms.

Os valores estão dentro do esperado: a variação da câmera pode ser explicada pelo atraso (*delay*) no processamento de cada *frame*, enquanto o LiDAR e os radares possuem periodicidade quase constante, na qual a quantidade de pontos nas mensagens interfere pouco, com os limites bem definidos pela metade do seu período. Esses resultados indicam que a infraestrutura de sincronização baseada em *software* manteve defasagens médias abaixo de 15 ms, com valores mínimos próximos de 1 ms.

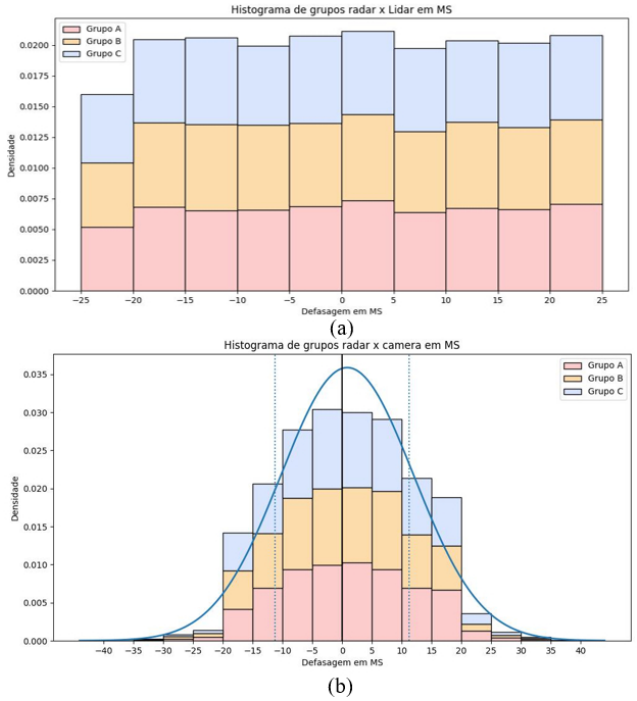

Fig. 6. Quantidade absoluta de defasagem entre os sensores. (a) radar e LiDAR. (b) Câmeras e radar

D. Armazenamento por ROS2 bag

O armazenamento dos dados foi concebido como parte essencial da infraestrutura, já que a plataforma SENSYNC visa não apenas processar fluxos em tempo real, mas também gerar *datasets* reprodutíveis para pesquisa. O perfil padrão de QoS foi ajustado conforme os nós produtores, e o cache desativado para que cada mensagem fosse registrada imediatamente. Os dados incluem os fluxos brutos (nuvens de pontos e imagens comprimidas), além de metadados de sincronização (frame_id), diagnósticos e registros de latência. Essa abordagem transforma o *rosbag* em uma unidade de análise completa, facilitando a reprodutibilidade e comparações entre sessões. Assim, o uso do `ros2 bag` foi incorporado à arquitetura como mecanismo de organização e padronização dos dados.

V. Conclusão

Este artigo apresentou uma arquitetura de aquisição e sincronização multimodal baseada exclusivamente em *software* no ROS 2, integrada a um veículo real equipado com câmeras, radares e um LiDAR. A técnica utiliza o radar como referência temporal e combina filas dinâmicas com o filtro *ApproximateTime* para alinhar sensores de diferentes frequências.

Os testes em vias urbanas do Recife–PE mostraram defasagens máximas abaixo de 25 ms e médias de 9–12 ms, valores adequados para aplicações de percepção veicular. A plataforma sustentou uma taxa de gravação de aproximadamente 29 Mb/s, resultando em cerca de 12 GB de dados sincronizados. Esses resultados confirmam que a sincronização em *software* é viável e reprodutível em condições reais.

A principal limitação observada é a variação temporal (*jitter*) provocada pelo sistema operacional e pelas diferentes características dos sensores. Futuramente, serão avaliadas técnicas para reduzir essa variação, como o uso do kernel PREEMPT_RT e otimizações de QoS, além da ampliação do *dataset*.

Agradecimentos

Os autores agradecem a FUNDEP Programa Mover - Linha V, pelo incentivo financeiro aos projetos FACEPE(APQ-1698-1.03/22) e SEGCOM (27192*80), VEHICLE_OTA (29271*10) também à FINEP (01.22.0157.00 - REF. 1157/21), assim como ao CNPq (304391/2021-2) e CAPES (88887.571378/2020-00).

References

[1] J. Gu, A. Lind, T. R. Chhetri, M. Bellone, and R. Sell, "End-to-end multimodal sensor dataset collection framework for autonomous vehicles," Sensors, vol. 23, no. 15, p. 6783, 2023.

[2] A. K. Tyagi and S. U. Aswathy, "Autonomous intelligent vehicles (AIV): Research statements, open issues, challenges and road for future," International Journal of Intelligent Networks, vol. 2, pp. 83–102, 2021.

[3] Y. Ye, Z. Nie, X. Liu, F. Xie, Z. Li, and P. Li, "ROS 2 real-time performance optimization and evaluation," Chinese Journal of Mechanical Engineering, vol. 36, art. no. 144, 2023.

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[5] K. Huang, B. Shi, X. Li, X. Li, S. Huang, and Y. Li, "Multi-modal sensor fusion for auto driving perception: A survey," arXiv preprint arXiv:2202.02703, 2022.

[6] X. Wu, H. Sun, R. Wu, and Z. Fang, "EverySync: An Open Hardware Time Synchronization Sensor Suite for Common Sensors in SLAM," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2024, pp. 12587–12593.

[7] I. P. Junior, L. P. Horstmann, and A. A. Fröhlich, "Enabling Time Synchronization with Hardware-in-the-Loop Integration on a Data-Driven Middleware for Autonomous Vehicles Simulations," in Proc. Workshop Latinoamericano Dependab. Segur. Sist. Veícul., 2024, pp. 5–8.

[8] H. Hu, J. Wu, and Z. Xiong, "A Soft Time Synchronization Framework for Multi-Sensors in Autonomous Localization and Navigation," in Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron. (AIM), 2018, pp. 694–699.

[9] Dataset SeSync, "Dataset SeSync [Online]," Google Drive, 2025. Available: https://drive.google.com/drive/folders/12Uwej7a3jy37YYJgQGBBiom9UeqL66kS. [Accessed: Oct. 2, 2025].