Development and Implementation of a Hands-On Robotic Course on Computer Vision for Autonomous Vehicles

Golnaz Habibi, Samuel Billerbeck, Daniel Vargas, Daniel Perez Melo, Evan McCulley
University of Oklahoma, golnaz, billerbeck, dvargas88, danielperez, evan.l.mcculley-1@ou.edu

FIGURE 1:
ARCPRO FLEET OF TEN 1/10TH SCALE CARS BUILT AT A LOW-COST FOR THE OFFERED COURSE IN COMPUTER VISION AND ROBOTICS

2025 IEEE Integrated STEM Education Conference (ISEC) | 979-8-3315-1343-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISEC64801.2025.11147426

***Abstract* – This paper introduces the development of a robotic course, Computer Vision for Autonomous Vehicle. Derived from VNAV course offered at MIT, this course is designed for senior undergraduate students and graduate students at the University of Oklahoma, where there are a few courses focused on robotics. Our course offers hands-on experience with robotic cars which have been built at low cost, allowing the students to implement fundamental materials in computer vision in a physical minicity environment built at the university. We integrated a popular visual Simultaneously Localization and Mapping algorithm, called Kimera, which allowed students to gain extensive experience with computer vision and visual SLAM in a low-cost 1/10th scale minicity, mimicking real-world features. We have provided the Docker version of VNAV course which makes the course broadly accessible to students and educators from different backgrounds in computer science and robotics. The final project is designed for students with different levels of experience in robotics and programming, which can be considered for K12 students in the future to introduce them to robotics/computer vision. While there are few teams and research labs at the University of Oklahoma focusing on robotics, this course bridges the gap by introducing a set of assignments in which the students get involved with data collection in a real-world environment and practicing different concepts of computer vision in a physical robotic platform. All the course materials, including the docker image, are open source, which can be utilized by the instructors and students around the world. We have also released the building material and instructions available at: https://github.com/airou-lab/CV_Autonomous_Vehicles.git**

Index Terms – robotics, software, Docker, computer vision, minicity, ROS, lab assignments, SLAM

INTRODUCTION

Robotics is a rapidly advancing field that integrates engineering, computer science, and artificial intelligence to design and develop machines capable of performing a wide range of tasks. With applications in autonomous navigation, transportation, healthcare, agriculture, and space exploration, robotics plays a crucial role in modern technological advancements.

FIGURE 2
1/10TH SCALE MINICITY LAYOUT USED IN THE OFFERED COURSE

Additionally, robotics serves as an effective tool for introducing STEM concepts to students at various educational levels. Among the key technologies driving robotics, computer vision has become particularly essential, enabling machines to intelligently perceive and interact with their environments. As robotics education evolves, there is a growing emphasis on integrating modern AI-driven perception techniques to enhance autonomous decision-making. In particular, deep learning-based convolutional neural networks (CNNs) and vision transformer (ViT) models have surpassed traditional object detection approaches, offering greater accuracy and robustness. Many recent developments in mobile object detection focus on

adapting these large models for deployment on autonomous systems, enabling real-time decision-making in dynamic environments. Models such as MobileViT V4 [1] and RepViT [2] exemplify this trend optimizing transformer-based architectures for efficiency on mobile platforms. As these models continue to expand in size and complexity, there is a growing need for modular, scalable approaches that allow researchers and students to experiment with state-of-the-art techniques in practical settings. At the University of Oklahoma, where undergraduate enrollment significantly exceeds graduate enrollment, students have had limited opportunities for hands-on robotics experience due to constraints such as high equipment costs, laboratory limitations, and the complexity of integrating multiple technical domains. To address these challenges, a new course has been developed, drawing inspiration from the VNAV course at MIT [3], but tailored specifically to undergraduate needs. This course emphasizes practical computer vision algorithm development, allowing students to implement and test computer vision techniques in real-world scenarios, with a particular focus on odometry and mapping as fundamental components. By combining theoretical foundations with practical implementation, students develop comprehensive skills to address real-world robotics challenges. This initiative not only contributes to Oklahoma's expanding technology sector but also prepares students for careers in robotics and automation. The course is designed to support academic development while equipping students with skills directly applicable to industry. As autonomous systems become more prevalent in fields such as manufacturing, agriculture, and logistics, there is increasing demand for engineers who can develop, integrate, and test real-time computer vision algorithms. By mirroring common workflows, such as sensor integration, algorithm optimization, and deployment on resource-constrained platforms, the course prepares students to contribute effectively to robotics teams after graduation. In this way, it serves as both a practical introduction to real-world engineering problems and a foundation for careers in emerging automation fields. Unlike the broader VNAV course, which covers all components of autonomous navigation, this course narrows its focus to developing practical computer vision algorithms specifically tailored to undergraduate studies. To facilitate hands-on learning, the course utilizes a Docker-based environment that minimizes prerequisites in Robot Operating System (ROS) [4] and Unix-based systems. Students gain experience with image processing, feature extraction, and object recognition within the context of autonomous navigation. The course is designed to run on either ROS 1 via Ubuntu 20.04 (ROS Noetic) or ROS2 humble [5] and is compatible with multiple operating systems, including Mac, Windows Subsystem for Linux 2 (WSL 2) [6], and Docker [7]. A final project further reinforces the concepts learned, enabling students to analyze visual mapping and localization while exploring the limitations of vision-based odometry under varying conditions such as lighting changes and vehicle speeds. Additionally, to bridge the educational gap in Oklahoma's robotics education, this course introduces an open-source repository containing educational resources, laboratory exercises, and programming examples. By providing hands-on experience in computer vision, the course equips students with skills applicable to autonomous vehicles, industrial robotics, and AI-driven automation, preparing them for real-world careers in robotics and automation.

Main Contribution:

The main contribution of this course is as follows:

- We provide low-cost robotic hardware along with an inexpensive controlled minicity environment [8], which provides a unique and equal testing bed for all students, to analyze the vision-based Simultaneous Localization and Mapping (SLAM) algorithm in real-world scenarios and study the strengths and weaknesses of such algorithms.
- We built Docker images for ROS1 and ROS2 [9] which make this course accessible for a broad range of students from little to no background in software engineering to a variety of students.
- The course covers a range of assignments such that students with different levels of experience in robotics and programming can benefit from them. Other than assignments on pre-recorded data, students experience collecting data from on-board sensors and applying computer vision algorithms on real robotic hardware.
- The result demonstrates the increasing students' abilities in problem solving and critical thinking, which increases when they gain research experience. In Fall 2024, over 50% of the students showed an interest in writing a research paper based on their findings on the final project which embarked on their research in robotics.

RELATED COURSES

This section reviews the existing literature and notable courses that merge robotics and computer vision**.** Courses that combine robotics and computer vision are becoming increasingly important in the academic landscape as both fields are critical for developing autonomous systems capable of navigating and interacting with the world. These interdisciplinary courses aim to equip students with the knowledge and practical skills necessary to design intelligent robots that use visual data for decision-making, perception, and task execution. Some of the related courses are summarized in Table 1. The rest of this section is focused on the related courses in detail.

Foundational Courses: Many universities offer introductory courses that teach the core principles of both robotics and computer vision. These include topics such as motion planning, control systems, visual SLAM, and object recognition. For example, courses like *CS 147: Introduction to Robotics and Computer Vision* at Stanford University [10] introduce students to the integration of robot control with

perception algorithms. Similarly, Carnegie Mellon University's *Robotics and Computer Vision* [11] course covers visual processing techniques necessary for autonomous robotics and computer vision, which are two of the most exciting and rapidly advancing fields in modern technology.

Table 1: Comparison of Courses across Different Universities

Course Name	University	Focus Area	Hands-on Experience	Open Source
Intro to Robotics [10]	Stanford	Integration of robot control and perception algorithms	Simulation	Yes
Computer Vision for Engineers [11]	Carnegie Melon	Autonomous robotics and computer vision	Simulation	No
Cognitive Robotics [13]	MIT	Reinforcement learning for robot decision-making	Real Hardware	Yes
Autonomous Robotics [14]	Georgia Tech	Interfacing with robots through ROS	Mixture	No
Robotics Specialization [15]	UPenn	Robot perception and control	Simulation	No
VNAV [3]	MIT	Computer-vision and perception using ROS	Simulation	Yes

Courses in the theory of robotics: robotics courses typically cover both theoretical and practical aspects of the field. The core areas include kinematics, dynamics, control systems, robot programming, sensors, and actuators. According to Siciliano et al. [12], foundational topics such as forward and inverse kinematics, path planning, and localization are critical in shaping students' understanding of robot motion and behavior. Many introductory courses, such as those offered by MIT [13] and Stanford [10], focus on mathematical modeling and control techniques.

Robotic Programming and Algorithms: Programming courses in robotics are designed to introduce students to different programming languages and software environments used for robot control. Popular languages include C++, Python, and MATLAB, often paired with robot operating systems (ROS). In the course offered by Georgia Tech's College of Computing [14], students learn how to interface with robots through ROS, developing skills in hardware abstraction and communication protocols.

Practical Applications and Case Studies: Hands-on projects and real-world applications are central to robotics education. Many courses emphasize integrating theoretical knowledge with practical challenges. The *Robotics Specialization* formerly on Coursera, created by the University of Pennsylvania [15], for instance, exemplified this approach by combining lectures on robot perception and control with practical assignments in which students programmed simulated robots to complete complex tasks such as obstacle avoidance and object manipulation. As robotics technology is applied across different domains, courses are beginning to integrate more specialized fields, such as medical robotics or industrial automation.

Emerging Trends and Innovations - Recent trends in robotics education emphasize interdisciplinary approaches. Courses are increasingly incorporating elements from AI, machine learning, and deep learning to advance robot autonomy. For example, *MIT's Advanced Robotics* [13] course includes modules on reinforcement learning for robot decision-making, allowing robots to improve their performance over time. Furthermore, the growing interest in autonomous systems in fields like autonomous vehicles and drones is reflected in new educational tracks that focus on autonomous navigation and the use of AI in decision-making.

This course is based on the VNAV course [3] that has been taught at MIT for several years now. The difference of the proposed course is to offer the course to a variety of students with little-to-no background in robotics and introduce them to computer vision concepts in a practical format. Also, students with different levels of skill in embedded systems can run experiments on different machines such as Ubuntu, Windows (WSL), and MacOS. We use a controlled minicity environment for robotic experience. While there are several minicity environments in the world, our minicity [6] is used for both educational and research purposes. We were inspired by $1/10^{th}$ minicity built at MIT [16]. While their work focuses on studying complex driving scenarios such as roadways with roundabouts, three-way intersections, and blind roads around buildings, we have designed our minicity with a detailed visual background, variety of buildings with varying heights and dimensions, making it suitable for testing and implementing vision-based mapping and navigation.

COURSE HARDWARE

I. ARCPro Robotic Cars

We have built ten robotic cars, called ARCPro, which we used on this course. Figure 1 shows the fleet of ArcPro vehicles. ARCPro is the successor to AirOU Race Car (ARC) with upgrades to the processor, chassis, and body. Figure 3 shows the ArcPro components. While ARC design is based on MuSHR robots [17], ARCPro is equipped with advanced processors and body design. The processor was upgraded from an NVIDIA Jetson Nano 4 GB in ARC to the Intel NUC i7 with 32 GB RAM. This significantly improved computational performance, allowing us to upgrade from offline processing only to onboard real-time processing. The chassis was also upgraded from a plastic frame to titanium alloy with dual-shock suspension to allow for more durability and ability to carry the increased weight of the new processor. The body frame was designed and 3D printed by our team and designed to incorporate quick and easy removal of all parts in the case of repair and maintenance, yet still able to hold the materials without the concern for parts slipping off

the body. The ARCPro also has a 50A VESC (Variable Electronic Speed Controller) from Flipsky to control the velocity of the cars remotely via direct connection to the NUC. Since the processor was upgraded, the power requirement also increased. The NUC is powered by a LiPO battery which is connected via a power converter to boost the voltage to 19V as required to power the NUC. We also tuned the car PID control parameters to adapt to the new chassis.

II. Bill of Materials and Building Instructions

The price to construct one ARCPro is ($1675), which is more cost-effective than other models that currently exist [18-19]. Please refer to the ARCPro project page for a more detailed breakdown of the prices. Building instructions and documentation are also provided in our GitHub.

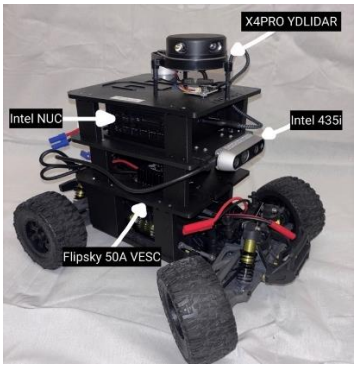

FIGURE 3
BREAKDOWN OF ARCPRO COMPONENTS

III. 1/10th scale Minicity

The AirOU minicity is a 1/10th scale simulated urban environment meant to act as a testing bed, which was built at the University of Oklahoma in Spring 2024. We were inspired by MIT minicity. However, our minicity is detail rich, allowing for extensive testing of computer vision applications and feature tracking. minicity is an experimental testbed where the students performed their experiments and recorded their data. It has also served as a testing bed for our own research when it was not in use for classroom purposes. Overall, the minicity is a low-cost test bed, at $1000 USD for a $16 \times 22\ ft^2$ city. Between a team of five with instructions, the construction takes less than four hours and will be easily transportable and easy to redesign the buildings if desired. Figure 2 shows the minicity layout.

COURSE SOFTWARE

The course software is categorized into two groups: software used for lab assignments and software installed in the car.

I. Software package: Docker Image, WSL compatibility

While we used VNAV lab materials, we built this course around Docker to simplify deployment and ensure cross-platform compatibility. Docker is a suite of tools that allows software to run in isolated containers, bundling each application with its dependencies. This avoids compatibility issues with the host OS and is significantly lighter than running full virtual machines. We created a Docker container preloaded with everything needed for ROS 1, and also used containers for MuSHR and for initial testing of Kimera-VIO [20], which was later deployed using its official ROS wrapper. These tools often require outdated Linux distributions and complex dependencies, making them difficult for students to install natively. Docker allowed students to work from their own systems, often Windows with WSL, without needing to dual boot or downgrade their OS. However, Docker introduces its own challenges. Students had to understand container concepts such as ephemeral environments, volume mounting, and image persistence. Attempts to hide these complexities through abstraction occasionally backfired, leading to errors that were difficult for students to debug. Despite CS students having more exposure to containers and Linux through other courses, both CS and non-CS students faced similar challenges when the abstraction failed. While prior knowledge of containers can be an advantage, students in CS generally did not have the advanced knowledge needed to reverse engineer the complex ROS container. Anecdotal observation also suggests that the illusion of familiarity may have offset some of the advantages possessed by students who had used simpler Docker environments in the past. Despite this, the benefit was clear: Docker made advanced tools like Kimera and MuSHR accessible in a reproducible and portable way. Furthermore, the failure of the abstraction was not due to a weakness of Docker, but rather because no realistic solution can address the true problem, that being heterogenous, unmanaged student devices. Future iterations of the course will continue to use Docker, but most likely on a cloud platform where the underlying compute can be more tightly controlled. These future containers will be streamlined through base images provided by the Open Source Robotics Foundation [21]. We can then extend these with course-specific packages using Dockerfiles, which improves transparency and allows students to further customize their environments. These containers, including the ROS 2 version, are created and maintained by the University of Oklahoma's Computer Science IT staff [9].

III. Remote Desktop Control for Remotely Control the Car

In this course, we used xRDP [22], a free and open-source implementation of the Remote Desktop Protocol (RDP), to access the Ubuntu OS on embedded robots. xRDP was chosen for its ease of setup—installation is usually sufficient—and because RDP clients like Windows Remote Desktop Connection are widely available. Alternatives such as VNC, X11 forwarding, or SSH tunneling require more

setup and often lack compatibility with GUI defaults. Using a desktop environment simplified access and tool configuration. Many applications in the course assume a local GUI, and a remote desktop preserved expected behaviors without requiring complex setup. This was especially helpful for students less comfortable with command-line tools. It also enabled direct interaction with camera feeds, simulations, and GUI-based sensors that are harder to manage through CLI alone. To streamline connectivity, each robot was assigned a static IP and DNS name, allowing students to connect using hostnames instead of retrieving dynamic IPs from local displays. We also automated Wi-Fi setup, reducing network configuration steps. This approach has limitations. Desktop environments consume more memory and are unsuitable for low-resource systems. Linux restricts remote sessions, blocking common actions like shutdown and Bluetooth pairing—even for users with local permissions. These restrictions are difficult to bypass without adding complexity. Also, only one user can connect via RDP at a time, which sometimes disrupted group collaboration. Although not ideal for production or long-term deployments, remote desktops offered a fast, low-overhead solution that aligned well with the course's teaching goals. By leveraging GUI defaults and reducing configuration burdens, this approach allowed students to begin working quickly, focusing on robotics development rather than setup issues. Ultimately, an optimized solution should not rely on a remote desktop, but when other tools do not suffice, or a simple solution is needed due to time constraints or lack of technical expertise, it can be highly effective.

II. Deploy Visual SLAM software on the car.

A variation of Kimera, Kimera Visual Inertial Odometry (Kimera-VIO), was selected as a SLAM framework for the final project. Since students learned about Kimera SLAM and its components during the class lecture and previous lab assignments, we considered running Kimera on the car as the final project, where students will have real-world experience in SLAM. We consider Intel NUC i7 as an on-board processor for real-time processing of Kimera. Kimera-VIO was selected for its effective integration of stereo vision and inertial sensor data, delivering robust real-time localization while carefully balancing the trade-offs between computational complexity, efficiency, and performance. Its design achieves a practical compromise by providing sufficient accuracy without overwhelming the processor, which is crucial for real-world SLAM applications. The system utilized an Intel RealSense D435i camera, combining stereo vision and its built-in IMU for visual-inertial odometry. To optimize performance, the IR sensor on the camera was deactivated, eliminating potential interference in our controlled environment. Students visualized the real-time trajectory in Rviz, which allowed for immediate assessment of the SLAM system's performance. Testing was conducted in the "minicity," a lab facility designed to replicate a scaled-down urban environment. This provided a realistic yet controlled scenario to evaluate the system. The localization accuracy was validated against a ground truth based on precise measurements of the minicity layout. Additionally, the car motion was controlled by the VESC which was connected to the Intel NUC. These two communicated through the ROS wrapper of the MuSHR package. By customizing the controller part of MuSHR, a speed limit was set on the autonomous vehicles to ensure smooth motion, prevent collisions, and improve SLAM accuracy. Abrupt movements could introduce noise or inaccuracies, which this measure successfully mitigated.

III. Camera Calibration packages: Kalibr + intel SDK

The camera was carefully calibrated prior to being handed over to the students. Using Intel's Python script provided for the D435i, the IMU was calibrated by keeping the device steady in six predefined positions to zero the gyroscope and accelerometer readings. This process provided intrinsic parameters such as accelerometer sensitivity, bias, and off-axis correction. Additionally, a three-hour static ROS bags were recorded with the cameras in a stationary position to characterize the IMU's noise. The extrinsic calibration between the camera and the IMU was performed using the well-known library Kalibr [23]. A ROS bag with synchronized image and IMU data was recorded, during which the camera was moved in various directions while tracking an AprilGrid target. Kalibr processed this data to compute the spatial relationship between the IMU/camera.

COURSE CURRICULUM

The class was offered in Fall 2022 for the first time at the University of Oklahoma, and it was initially offered to graduate students only. We followed a similar curriculum as VNAV course in the first year of the course. The class gradually included more hands-on robotics lab assignments such that the minicity and MuSHR-based racecars (*i.e., ARC*) were used for final projects in Fall 2023 offering. As the course was established in Fall 2024, it was offered for both graduate and undergraduate students. While the original VNAV course focused on the entire navigation pipeline, we offered a course with concentration on computer vision. Table 2 shows the course schedule. The lectures follow VNAV lectures, so we avoid repeating them here. Please refer to VNAV paper for more detail. In summary, the lectures still cover the entire navigation pipeline to provide a big picture of autonomous navigation and where the computer vision component. However, the course assignments mainly focused on visual geometry, visual odometry, and mapping. The difference between this course and the VNAV course is the assignments, focusing on evaluating computer vision methods on real dataset collected

by the students in a 1/10th scale minicity. The racecars are equipped with RealSense RGB-D cameras with built-in IMU. To achieve an accurate result, camera-IMU calibration is necessary. Students are provided step-by-step instructions for data-collection as well as camera-IMU calibration. Project assignments can be done in a group of 1 to 3 students. Students also develop their experience in teamwork throughout the semester. The course has seven main lab assignments and one final project. Lab assignments are briefly listed in Table 2. The final project was mandatory for the graduate section of the course, and optional for the undergraduate section for extra credit. This project focused on understanding Kimera, a visual inertial odometry (VIO) algorithm, in-depth and running the Kimera package on the car. While the Kimera package has already been setup by the course staff, the goal was to improve the critical thinking skills in students, and their reasoning about different outcomes, whether success or failure, and how they can ideate novel approaches to avoid failure or improve upon it. Students also have this option to present their study and experiment in front of their scholars and professors in an annual presentation forum which motivates them to present their work in a compelling way.

TABLE 2: Lab assignments

Lab	Lecture	Timeline
1	Introduction to C++, Docker, GitHub	WEEK 1
2	Introduction to ROS	WEEK2
3	Computer vision: feature matching	WEEK3-4
4	3D pose-estimation of car in minicity, Camera-IMU calibration	WEEK5-6
5	Pose graph optimization, GTSAM	WEEK 7-8
6	Loop closure detection in minicity Object detection using YOLO	WEEK9-10
7	Final exam: SLAM algorithms (Euroc dataset)	WEEK 11
final	Final project (real-time SLAM in minicity)	WEEK12-16

COURSE ASSESSMENTS:

The Fall 2024 offering included an undergraduate section, and it is an elective course in the school of computer science at the University of Oklahoma. The course was composed of 28 students, with disciplines comprising 70% Computer Science, 2% Environmental Science, 10% Electrical Engineering, 8% Data Science, 10% Mechanical Engineering. In this lab, students drove the car along a predefined, green-marked path, ensuring it passed through key landmark points (red circles), with expected loop closures at three major intersections (black circles). The goal was to implement DBoW2, a Bag of Words-based place recognition framework, to detect when the car revisited known locations. The detection process involved replaying the recorded rosbag in RVIZ, which visualized the trajectory and image frames. A Python script extracted and indexed images from the rosbag to prepare them for feature matching. The DBoW2 pipeline then compared newly recorded images to previously seen locations, identifying loop closures. Students fine-tuned score thresholds to balance detection sensitivity and false positives, ensuring robust place recognition. Figure 4 shows the loops in the minicity.

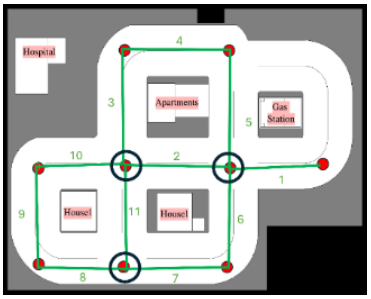

FIGURE 4
RESULT FROM KIMERA VIO WHEN IT WAS DRIVEN IN THE MINICITY UNDER DIFFERENT CONDITIONS: (TOP-LEFT): LOW SPEED, LIGHT WAS OFF; (TOP-RIGHT): HIGH SPEED, LIGHT WAS OFF; (BOTTOM-LEFT): LOW SPEED, LIGHT WAS ON; (BOTTOM-RIGHT): HIGH SPEED, LIGHT WAS ON.

I. Final Project: Kimera-VIO in Minicity

The final project aimed to provide students with practical experience on implementing and analyzing Kimera VIO in real time. Students were tasked with driving the car along the middle line of the minicity road. Figure 2 shows the yellow lane where the cars were supposed to drive on. The student also recorded images and IMU data while running Kimera in real time to estimate the car's trajectory. Afterwards, they compared the estimated trajectory to the ground-truth provided to them (i.e., tape measured yellow lane). A car's travel distance and direction were calculated relative to the corner of minicity, with measurements recorded in centimeters. These values were compared to the output from the car's Kimera VIO framework for the final project. Figure 6 has an example run with lights on and fast speed to show the overlapping with manual ground truth. This method of manual ground truth setup is suitable for a low-cost implementation of this project without expensive GPS-based, motion capture systems, or ceiling mounted cameras to create a ground truth. Figure 5 illustrates the Kimera VIO system in operation, with the video feed from the car's camera and the generated map being displayed from a rosbag file capturing real-time data as the system processes the car's movement, updates its pose estimation.

II. The effect of ambient light and car speed in Kimera performance

Part of the assignment was also to understand the Kimera component as well as evaluating its performance under varying conditions such as car speeds or lighting. Students conducted four separate runs to evaluate the performance of the Kimera VIO system under different conditions. The first run, with the lights off and the car moving slowly, resulted in the poorest performance. While there was some overall coherency in pose estimation, the loop closures were largely inaccurate, and the distances were completely off. The

limited lighting significantly impaired the system's ability to use visual odometry effectively, forcing it to rely more heavily on IMU and motor encoder data, which led to inaccuracies in the loop closures. Ground truth data was established by measuring one corner of the car course, which served as a fixed reference point.

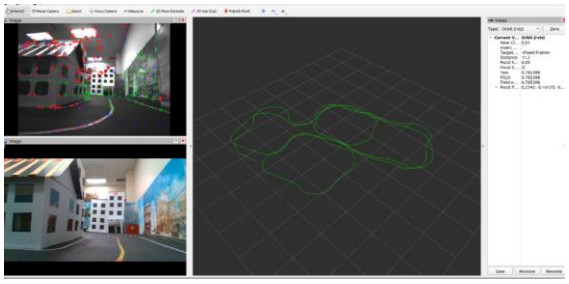

FIGURE 5

LAB 4: ROS VISUALIZATION (RVIZ) TOOL WHEN KIMERA IS EXECUTED IN MINICITY. THE CAR WAS DRIVEN IN THE MINICITY USING ON-BOARD PROCESSING. TOP-LEFT: SHOWS THE EXTRACTED AND MATCHED/UNMATCHED VISUAL FEATURES. BOTTOM LEFT: THE CAMERA VIEW OF THE CAR. RIGHT: THE RESULT OF THE KIMERA VIO, WHICH ESTIMATES THE CAR TRAJECTORY WHEN THE CAR IS DRIVEN IN MINICITY.

The second run, with lights on and the car moving faster, showed considerable improvement. The increased speed allowed the system to incorporate more input from the IMU and motor encoders, which led to more coherent loop closures and some improvement in the distances. However, while the results were better, they were still not optimal, indicating that the system's reliance on visual odometry was still not fully leveraged in this scenario. Much more performance was achieved in the third run, with lights on and the car moving slowly. In this case, the system produced highly accurate pose estimation and loop closures, with minimal errors in trajectory shape. The improvement compared to the light-off scenarios was due to more visual information being available with the lights being on. Thus, the Kimera VIO system was able to operate properly and there was much less reliance on the imu and motor encoders to keep coherence of pose estimation. However, the final run, lights on and the car moving quickly yielded the most accurate results overall. The synergy between visual odometry, IMU, and motor encoder data ensured that the loop closures were almost perfectly overlapped and highly congruent with ground truth data. This highlights the importance of combining these sensor inputs, particularly in dynamic environments with fast movement, as it significantly improved both pose estimation and loop closure performance. Figure 7 shows this experiment. To improve detection reliability, students incorporated RANSAC-based geometric verification, filtering out incorrect matches by analyzing feature correspondences between frames. This step ensured that loop closures were validated using geometric consistency, reducing reliance on raw BoW scores. The finaldeliverables included a list of detected loop closures, a comparison of results before and after geometric verification, and a visualization of the estimated trajectory with detected loop closures in RVIZ. This lab provided hands-on experience in place recognition, emphasizing the importance of loop closure for correcting localization errors, reducing drift, and improving long-term mapping accuracy.

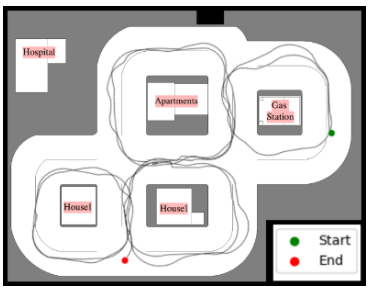

FIGURE 6

RESULT FROM KIMERA POSE ESTIMATION (I.E., TRAJECTORY OF THE CAR), OVERLAYED WITH THE GROUND TRUTH IN THE MINICITY

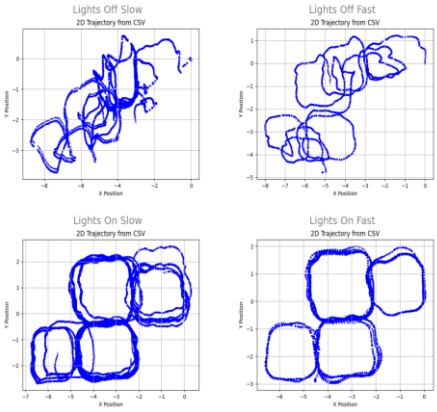

FIGURE 7

RESULT FROM KIMERA VIO WHEN IT WAS DRIVEN IN THE MINICITY UNDER DIFFERENT CONDITIONS: (TOP-LEFT): LOW SPEED, LIGHT WAS OFF; (TOP-RIGHT): HIGH SPEED, LIGHT WAS OFF; (BOTTOM-LEFT): LOW SPEED, LIGHT WAS ON; (BOTTOM-RIGHT): HIGH SPEED, LIGHT WAS ON

COURSE ASSESSMENT AND STUDENT FEEDBACK

Two sets of student surveys were conducted to collect student feedback. The course has been offered Fall 2022 - Fall 2024. Figure 8 shows the result of the 5 students' feedback taking the course. This result shows that 80% of the students (4 out of 5) found this course improved their skill in problem solving, critical thinking, and research experience.

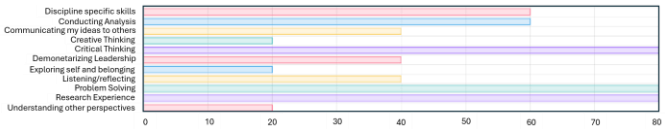

FIGURE 8

ANONYMOUS STUDENT SURVEY ON THE COURSE OFFERED IN FALL 2024

In the second survey, all 28 students participated. 83 percent of the students mentioned that this class increased their interest and engagement in computer vision. Among graduate students who took the final project, 86.6 percent of them found that the final project was very interesting and

rewarding. Sixty percent of the students submitted a research paper as a continuation of their final project, demonstrating the success of the course in motivating students to pursue research in robotics and computer vision. Since there are few robotics courses offered at the University of Oklahoma, this unique course offers core concepts from computer vision that students with different backgrounds and disciplines can take. The original VNAV course mainly serves as a graduate-level and senior undergraduate-level course for the students with prior experience in robotics. This latest version simplified the labs compared to the original VNAV course by providing a strong code base to the students which made it suitable for students with little or no robotics background experience.

COURSE CHALLENGES

This course was offered for students with different backgrounds in engineering, such as computer science, Mechanical Engineering, and Electrical and Computer Engineering. To make sure the course is well suited for students with different backgrounds, the course was simplified and more generalized compared to the VNAV course [3], such that some of the mathematical concepts in non-linear optimization were eliminated and considered as optional topics. Also, the lab related to Trajectory Optimization was considered an optional project.

CONCLUSION

This paper introduces a new course with an emphasis on hands-on robotic experience to understand complex computer vision material in a more interactive and experimental way. Using Docker for all the assignments makes this course manageable for students with a variety of backgrounds ranging from computer science, electrical engineering, to aerospace and mechanical engineering. The final project is designed such that students do not have to modify any prior software. They focus solely on running one of the Visual-SLAM algorithms on the robotic car and determining their strengths and limitations. The final provides an inclusive experience for all students to analyze the mapping and localization performance of a visual-based method without involving coding a complex algorithm from scratch. During the final project, students studied the Kimera paper [21] and during the implementation, they could see the clear connection between theory and practice. By varying environmental and dynamic conditions, students were able to analyze and criticize the problem. The final project is designed for students with different levels of experience in robotics and programming, which can be considered for K12 students in the future to introduce them to robotics.

ACKNOWLEDGMENT

We would like to thank Phong Nguyen, Alexander Cruz, Luyu Niu, and Christian Sink for their help in building the cars and collecting data. We would also like to thank the Gallogly College of Engineering at OU for financial support.

REFERENCES

[1] Mehta, Sachin, and Mohammad Rastegari. "Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer." *arXiv preprint arXiv:2110.02178* (2021).
[2] Wang, Ao, et al. "Repvit: Revisiting mobile cnn from vit perspective." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.
[3] Carlone, Luca, et al. "Visual navigation for autonomous vehicles: An open-source hands-on robotics course at mit." *2022 IEEE Integrated STEM Education Conference (ISEC)*. IEEE, 2022. Available at: https://ocw.mit.edu/courses/16-485-visual-navigation-for-autonomous-vehicles-vnav-fall-2020/
[4] Stanford Artificial Intelligence Laboratory et al., 2018. *Robotic Operating System*, Available at: https://www.ros.org.
[5] Macenski, Steven, et al. "Robot operating system 2: Design, architecture, and uses in the wild." *Science robotics* 7.66 (2022): eabm6074.
[6] WSL's developers, "WSL", learn.microsoft.com/en-us/windows/wsl/about, 2025
[7] Docker's developers, "Dockers", docker.com, 2025
[8] Vargas, Daniel, et al. "Design and Implementation of Smart Infrastructures and Connected Vehicles in A Mini-city Platform." *ITSC (*2024).
[9] Billerbeck, Samuel , ROS Docker for CS4733/5733, https://hub.docker.com/r/oucompsci/ros, 2025
[10] Khatib, Oussama, Introduction to Robotics, Stanford University
[11] Shimada, Kenji, Computer Vision for Engineers, Carnegie Mellon University (2017)
[12] Sciavicco, Lorenzo, et al. *Robotics: modelling, planning and control*. Springer, 2010.
[13] Williams, B. C., (2016), Cognitive Robotics, MIT, Available at: https://ocw.mit.edu/courses/16-412j-cognitive-robotics-spring-2016/pages/syllabus/
[14] Pradalier, Cédric, Autonomous Robotics, Georgia Tech in Europe https://europe.gatech.edu/sites/default/files/2023-09/CS%207630.pdf
[15] Lee, Daniel, et al, Robotics Specialization. University of Pennsylvania,
[16] Buckman, Noam, et al. "Evaluating autonomous urban perception and planning in a 1/10th scale minicity." *Sensors* 22.18 (2022): 6793.
[17] Srinivasa, Siddhartha S., et al. "MuSHR: A low-cost, open-source robotic racecar for education and research." *arXiv preprint arXiv:1908.08031* (2019).
[18] MIT race car developers, "racecar," 2015. https://racecar.mit.edu/
[19] Zheng, Billy, et al. "RoboRacer," *Roboracer.ai*, 2025. https: //roboracer.ai//index.html
[20] Open-Source Robotics Foundation, "OSRF Docker Images", github.com/osrf/docker_images, 2025
[21] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," arXiv:1910.02490 [cs], Mar. 2020 S. S. Srinivasa et al.,
[22] xRDP's developers, "xrdp - an open-source RDP server" github.com/neutrinolabs/xrdp 2025.
[23] Kalibr developers, "Kalibr", https://github.com/ethz-asl/kalibr.git

AUTHOR INFORMATION

Golnaz Habibi, Assistant Professor, School of Computer Science, University of Oklahoma
Samuel Billerbeck, System Administrator, School of Computer Science, University of Oklahoma
Daniel Vargas, Ph.D. student, Department of Aerospace and Mechanical Engineering, University of Oklahoma
Daniel Perez, Ph.D. student, Data Science and Analytics Institute, University of Oklahoma
Evan McCulley, Master's student, Department of Aerospace and Mechanical Engineering, University of Oklahoma