Department of Electrical Engineering and Computer Science

Howard University

Washington, DC 20059

EECE 404: Senior Design II

Spring 2024

Final Report

By:

Robert Jones, Alayen Pratt, Ahmad Abdur-Rahman, Ayron Fears, Karci Gibson

Instructor: Charles Kim, Ph.D.

Date Assigned: 4/3/2024 Date Submitted: 4/23/2024

Summary/Abstract:

This project aims to improve autonomous vehicle design with ORB-SLAM2. ORB-SLAM2 is a versatile and accurate Visual SLAM solution able to compute in real-time the camera trajectory and a sparse 3D reconstruction of the scene in a wide variety of environments. This was planned to be implemented onto a Raspberry Pi 4B with more hardware support from the Cyclone V architecture-based FPGA. This hardware design optimizes accuracy and precision, and with this system, it achieves rapid localized mapping critical for applications such as commercial driving and real-time navigation. To complete this design we needed to gather the required materials: Intel RealSense D435i camera, Arduino MKR Vidor 4000, Raspberry Pi 4 Model B, SunFounder PiCar-V Kit, Circuit Wiring, 12V Battery Pack, and Ultrasonic Sensors. There are some environmental, socio-cultural, and compliance constraints that we took into consideration when drafting our solution design, like U.S. Roadways, distrust in AI, and traffic laws. After this, we started our Top Solution design. We finalized our model with the Intel Realsense Camera at the front of the vehicular frame. The battery pack was placed and screwed on the bottom of the frame, while the hardware components were fitted and stacked appropriately on the top of the frame. Also, the ultrasonic sensors are attached to the sides of the frame. A picture of the design is shown below in the report. This design optimally reassured that the live feed from the camera was being processed by the Raspberry Pi 4B and Arduino.

Our agile workflow involved three sprints, each lasting three to four weeks, to complete our project. The first sprint focused on hardware retrieval and assembling the RC car, the second showcased camera usage, and the final sprint confirmed the car's functionality. Weekly in-lab meetings and Zoom meetings were used to discuss progress and agendas. The project involves assembling a car frame with a Raspberry Pi, Arduino, and Intel camera. The Arduino FPGA is housed on an extra platform. The software foundations were established using CMake, and the robot operating system (ROS) was used for control. Data was collected from ORB-SLAM2, and in ROS we attempted an autonomous navigation stack with a transformation offset tree, sensory data, odometry data, a base controller script, and a map server. A Graph Neural Network was created to solve the Bundle Adjustment problem. The team attempted to develop an autonomous vehicle using hardware design principles to improve processing time in software-centric approaches. We propose an FPGA with a GNN to optimize 3D scene reconstruction maps and an RGB-D camera for depth information. The remote control prototype serves as a proof of concept for autonomous vehicle design, with future research focusing on total development, image processing, and real-world simulations.

Problem Statement:

We aim to improve autonomous vehicle design with ORB-SLAM2 on a Raspberry Pi 4B and a Cyclone V architecture-based FPGA. Our implementation significantly improves processing, execution timing, and SWAP-related tradeoffs; hence prioritizing human safety. By emphasizing hardware-centric design, we optimize precision, efficiency, and navigation, overcoming limitations imposed by software-based approaches. Utilizing hardware-driven principles like pipelining, our system achieves rapid localized mapping critical for applications such as commercial driving and real-time navigation.

Design Requirements:

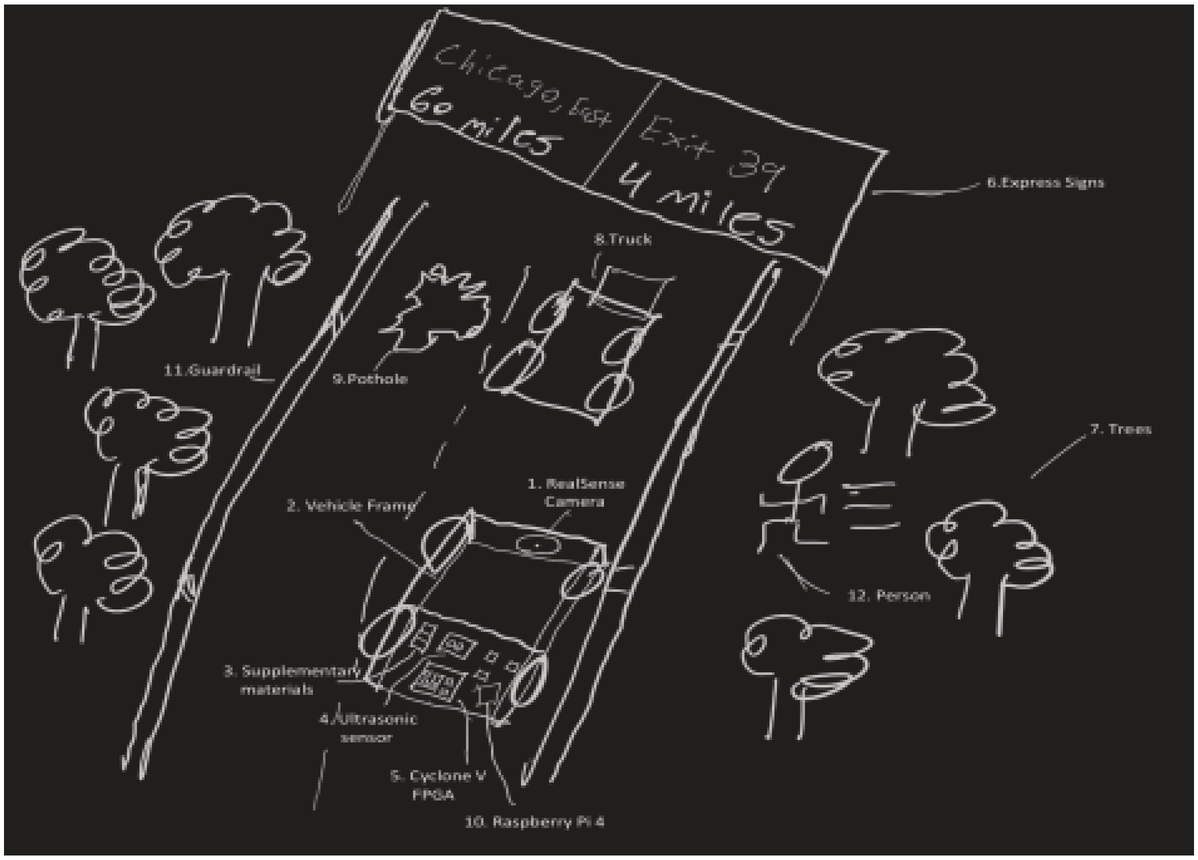
Design Requirement Form			
Date:	10/1/2023		
Project Name/Title:	SLAM		
Team Advisor	Dr. Seabron		
Project's Goal/Scope	Implementation of ORB-SLAM2 algorithm on an autonomous vehicle		
Team Members	Alayen, Ayron, Robert, Karci, Ahmad		
4-sentence problem statement	We plan to introduce a new standardization of autonomous vehicular design using the ORB-SLAM2 algorithm on a Raspberry Pi and FPGA. Our implementation will drastically improve processing and executable timing in addition to SWAP-C-related trade-offs and overall human safety. It will further address the need for a hardware-centric design to optimize the precision, efficiency, and navigation that is constrained by software-based designs. Our approach will be able to map its environment at the necessary speed for commercial driving because of its hardware-driven design, which utilizes principles such as pipelining in its operation.		
Requirements	**Items**	**Weight**	**Quantity**
1. Product Specification	Intel Realsense D435i Camera	Operating Range: (Min-Max) ~.3m - 3m Weight: Approx. 60-65 grams	1
	Arduino MKR Vidor 4000	Weight: 43.5 grams	1

	Raspberry Pi 4 Model B 4GB Ram	Weight: 46 grams	1
	Sunfounder PiCar-V Kit	Weight: 839 grams	1
	Circuitry wiring	Weight: 3-10 grams	1
	12V Battery Pack	Weight: 0.5-2 pounds	1
	Ultrasonic Sensor	Operating Range: (Min-Max) ~ 13ft	2
2. Constraints	**Environmental Constraints**		U.S. Roadways, Emissions, Object/Hazard Recognition
	Socio-Cultural Constraints		Social distrust in autonomous technology, Differing views on vehicle design preferences
	Compliance (Rules, Regulations, and Standards)		National Highway Traffic Safety Administration, Lack of human interpretation, Variation in regional traffic laws

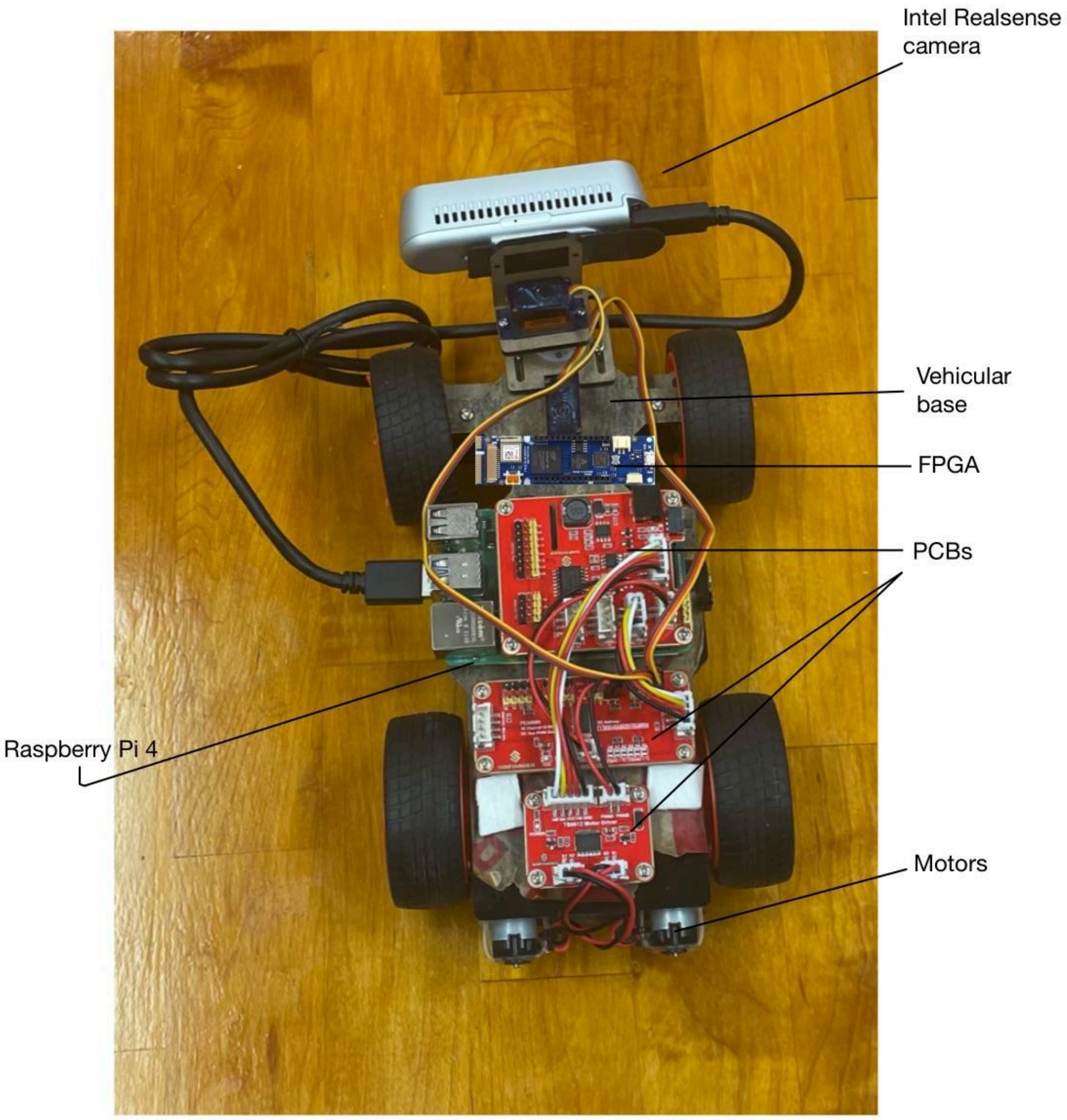
Top Solution Design :

Here we have our model which displays the Intel Realsense D435i camera (1) on the top of the vehicular frame (2) this will record/capture objects on the camera in 1080p in addition to the IR radar depth imaging. These images are then processed in the Raspberry Pi 4B (10) using the ORB-SLAM2 library and FPGA-based GNN to produce the most accurate rendition of a 3D scene map. The camera gyroscope and accelerometer are then used to map the vehicle’s location and spatial status via the Cyclone V FPGA (5). The ultrasonic sensors (4) are also connected to the Raspberry Pi(10) and powered by supplementary material such as the 12V Li-Ion Battery. They will emit a chirp that will detect the distance of nearby objects and send this data to the Raspberry Pi(10) for navigation. The Intel Realsense D435i camera captures static objects such as the guardrail (11), pothole (9), trees (7), and expressway signs (6), these images are then processed and used as a guide to keep the vehicular frame (2) positioned within its lane. Together, the Intel Realsense D435i camera and the ultrasonic sensors (4) work collaboratively to determine the vehicle's position and distance from nearby objects such as the person (12) and the truck (8). They quickly send this data back to the Cyclone

V FPGA to ensure enough processing and response time to avoid collision.



Component level



Agile Workflow:

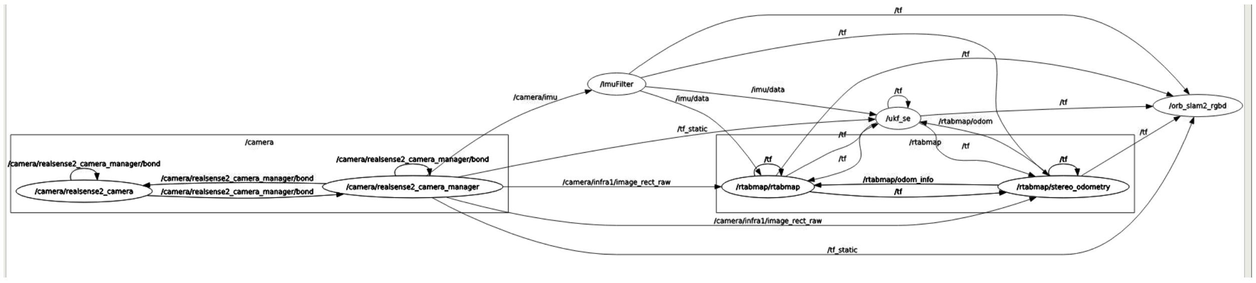
Our agile workflow consisted of three different sprints that we split into three to four-week increments between each sprint for the allotted tasks to ensure the completion of our project. The initial sprint was focused on retrieving the hardware for our products and assembling the RC car. The following sprint highlighted the successful usage of the camera along with the car and lastly, the final sprint tested and confirmed the car was fully operational with all the targeted functionality. To complete these sprints, we used a weekly implementation plan of in-lab meetings twice a week with each subsystem team having specific days and times. Also, we hold a Zoom meeting at the end of each week with all team members to discuss the upcoming agenda of the following sprints and progression from the previous ones.

To complete the incremented set of goals, we constructed weekly developmental tasks that would allow us to progress through each sprint as efficiently as possible. For the initial sprint, our main priority was the physical assembly and ensuring the car's base would fit all the desired components. Our first weekly task was to acquire the hardware parts necessary for the completion of the project which was a fairly easy task. Once acquired, the hardware subsystem team developed a layout design of the placement of each component on the vehicular base that would best utilize the limited spacing available . By the end of the three week increment allotted for sprint one, we were able to successfully complete the physical assembly and ensure that each component would be properly spaced for optimal performance and accommodate necessary wiring. The following sprint was an integral portion of our project that was primarily focused on the software subsection of the project. The main goal for sprint two was to ensure the connectivity and functionality of the Intel realsense camera with the car. Throughout the first two weeks assigned for this sprint, our team first established that the camera was operational and achieved the desired tasks so that we could move on to testing the data points it collected. Our team conducted several tests of the camera's data points to verify their effectiveness for optimal navigation. During the final week of sprint two, establishing the USB connection of the Raspberry Pi with the camera was the last step necessary for us to progress to the final sprint. The concluding set of our weekly tasks began with ensuring that the connection between the Raspberry Pi and the FPGA was secure and that all the components functioned properly as we anticipated. Once this was verified, the team was able to fully test the our 3-D reconstruction pipeline and confirm that the vehicle is able to accurately map its own environment autonomously at a highly efficient rate. The final weekly development task was to check the vehicle's ability to navigate and respond to changes in surroundings and adaptability to environmental chaos. To do so, we conducted several test touch and operation performance analysis on the vehicle before completing our final sprint.

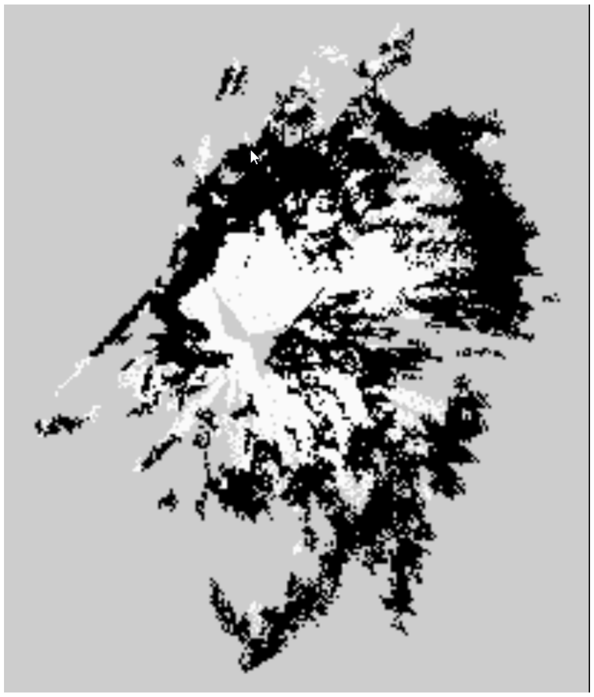
Project Implementation:

Since the car frame came as a kit complete with motors, servos, PCBs, screws, and tools, the electrical and mechanical tasks for the project were already planned out. The frame assembly with the Raspberry Pi, Arduino, and Intel camera finished quickly. Due to limited space available on the PiCar-V frame for all our experiment's components, we created an extra platform fixed to the top of the car which houses the Arduino FPGA. We moved on to establishing the software foundations for our experiment, creating a User-Interface with the camera and testing the official ORB-SLAM2 library on our Raspberry Pi 4B. Most libraries/dependencies for the Intel camera and ORB-SLAM2 were built with CMake, an open-source platform for software packaging and build automation. When building ORB-SLAM2, we encountered dependency version conflicts based on our C++ compiler and Operating System (Raspbian OS 10). Certain commands and operators were not recognized and had to be replaced. To process the Intel D435i camera's RGB and Depth input we used CMake to make an executable wrapper for linking the camera to ORB-SLAM2. We tested the camera's depth input from a laptop by taking videos of numerous objects such as water bottles, chairs, and books. The camera is powered and tethered to the car utilizing the Pi 4B's USB 3.0 port. This enabled the camera to transmit 720p images to the Pi 4B quickly. The Arduino FPGA is connected to the Pi 4B using the Inter-Integrated Circuit (I2C) communication protocol. It involves creating a serial bus interface using Serial Data (SDA) and Serial Clock (SCL) ports from both devices in a sequential circuit.

To establish control of the car systems, we used the Robot Operating System (ROS) and created scripts to format user input from the keyboard and output calculated velocity controls from the Pi to the PWM and moter controllers via an independent I2C bus. Due to the ease of use when integrating external packages in ROS, we also used it to interface with ORB-SLAM2 and the D435i camera. We experienced numerous issues with the camera on ROS, such as dependency conflicts, sensor drift, and random loss of transmission. Through calibration and rebuilding packages, we fixed most of the issues with the camera and began gathering data from ORB-SLAM2. We extracted point clouds, a set of 3D map points in space; and camera/keyframe pose trajectory, 3D map points and rotation coordinates based on camera origin. With this configuration, we were able to map our environment while driving remotely from a computer efficiently. We started working toward autonomous navigation by satisfying all data requirements from ROS: a transformation offset tree between the camera and car frame, sensory data from the camera, odometry data from the camera, a base controller script in Python, and a map server from ORB-SLAM2.

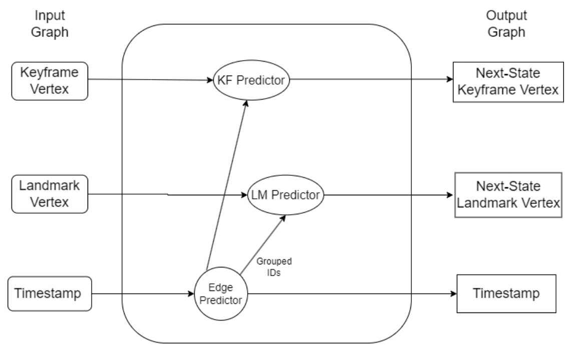

/camera
/camera/realsense2_camera_manager/bond
/camera/realsense2_camera
/camera/realsense2_camera_manager/bond
/camera/realsense2_camera_manager/bond
/camera/realsense2_camera_manager/bond
/camera/realsense2_camera_manager
/camera/imu
/ImuFilter
/tf
/tf
/tf
/imu/data
/imu/data
/tf_static
/tf
/ukf_se
/rtabmap/odom
/tf
/orb_slam2_rgbd
/rtabmap
/tf
/tf
/tf
/tf
/tf
/camera/infra1/image_rect_raw
/rtabmap/rtabmap
/rtabmap/odom_info
/tf
/rtabmap/stereo_odometry
/camera/infra1/image_rect_raw
/tf_static

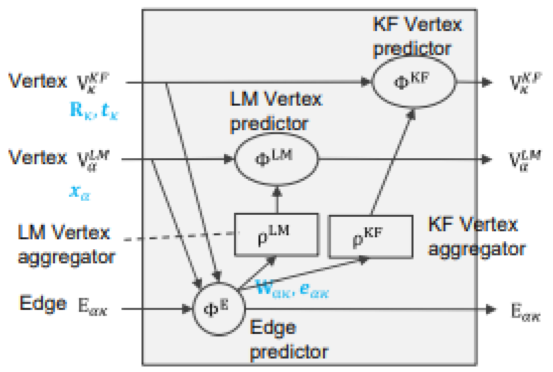
ROS data flow for ORB-SLAM2



Generated 2D grid map

As we approached the semester's end, we focused on implementing a GNN for solving the Bundle Adjustment problem. We tried 3 different Python Neural Network libraries to make our Graph Neural Network. We tried Pytorch, Graph-Nets, and Tensorflow-GNN. We used Tensorflow-GNN, a new library built by Google's DeepMind team. We used TF-GNN because it had the most comprehensive and accessible set of tools for creating a custom dataset despite being released this year. We constructed our data into a graph structure with keyframes and point clouds represented as nodes and the edges being IDs that map a single keyframe to ~500 map points in a point cloud. In the future, we will work on expanding the data representations in our graph. There is an approach we want to replicate that uses aggregator functions to calculate other parameters that could be useful for the bundle adjustment problem. We created a starter network that used dense layers and the relu function to calculate the next most statistically probable state of a single map point coordinate ('x') using the other 9 coordinates. Unfortunately, we have run out of time to complete our implementation of the GNN Bundle Adjustment as a part of the project and this report. While we plan to finish over the summer, we have made strides in completing the last major portions of the project.





Current graph structure **Objective graph structure**

Conclusion:

Our approach to emerging technology entailed utilizing hardware design principles to innovate a rapidly saturating field. The emergence of artificial intelligence has led to the introduction of many technological advancements, including computer vision and autonomous systems. One of these advancements, autonomous systems, incorporates integrated systems to accomplish tasks typically done by humans, such as utilizing a forklift or performing maintenance. Autonomous driving is one of the more complex tasks, and many attempts have been made to standardize and commercialize it. Many of the approaches utilize software-centric designs to accomplish this. These designs have far too much processing time to respond in near real-time, let alone real-time, which would be necessary to drive on the road with other drivers safely. To improve this bottleneck issue in processing, we approached our design utilizing hardware design principles, specifically, using an FPGA with a GNN trained to solve the Bundle Adjustment problem to optimize slam-generated 3D scene reconstruction maps faster than the Levenberg-Marquadt method. Our design used an RGB-D camera to capture depth information in addition to the live camera feed.

Using a Raspberry Pi 4B as our primary processor proved inefficient in processing the images in the ORB-SLAM2 library. Hence, our approach using an FPGA would substantially reduce the bottleneck in map processing. Throughout the sprints, we developed the vehicular base, connected the kit's PCBs and Raspberry Pi 4B, and developed ROS packages to establish remote control of the vehicle. We planned to use the Vivado High-level synthesis tool to convert a developed graph neural network from C++ into VHDL to increase the processing speed of the map points from the SLAM algorithm. Time constraints and problems with synthesizing the generated VHDL using Quartus halted the optimization of the pin allocation of the FPGA.

However, our car was able to be remote-controlled. The autonomous node was developed to enable the vehicle to move autonomously in software and work on optimizing the VHDL is still being done and will be done entirely in the coming weeks. Furthermore, we acquired a 12V 5A power supply to ensure all components are supported regarding power consumption and can still navigate as an independent vehicle. Our vehicle is well-researched and will serve as a well-produced proof of concept prototype as an autonomous vehicle that utilizes hardware to improve the insufficiencies of the software-driven approaches to an autonomous vehicle using commercially available components. Future works and research include developing the vehicle in its totality, pipelining the processing of images, and testing the vehicle response time in real-world simulations to achieve the real-time reaction necessary to drive on national roadways safely. Our current prototype is a significant first step in developing the standard for autonomous vehicle design, which will undoubtedly emerge in the coming years. We intend to continue our work and constantly improve our design until it accomplishes the declared intent to develop the standard design of autonomous vehicles.

References:

Marquardt, Donald W. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, 1963, pp. 431–41. *JSTOR*, http://www.jstor.org/stable/2098941. Accessed 23 Apr. 2024.

LEVENBERG, KENNETH. "A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES." *Quarterly of Applied Mathematics*, vol. 2, no. 2, 1944, pp. 164–68. *JSTOR*, http://www.jstor.org/stable/43633451. Accessed 23 Apr. 2024.

Liu, Qiang, et al. "π-BA: Bundle Adjustment Hardware Accelerator Based on Distribution of 3D-Point Observations." *I.E.E.E. Transactions on Computers/IEEE Transactions on Computers*, Jan. 2020, p. 1. https://doi.org/10.1109/tc.2020.2984611.

Qin, Shuzhen, et al. "π-BA: Bundle Adjustment Acceleration on Embedded FPGAs with Co-observation Optimization." *IEEE*, Apr. 2019, https://doi.org/10.1109/fccm.2019.00024.

Nikolic, Janosch, et al. "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM." *IEEE*, May 2014, https://doi.org/10.1109/icra.2014.6906892.

C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013. https://ieeexplore.ieee.org/document/6696650

Tanaka, T., Sasagawa, Y., & Okatani, T. (2021, October). Learning to Bundle-adjust: A Graph Network Approach to Faster Optimization of Bundle Adjustment for Vehicular SLAM. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv48922.2021.00619

Ferludin, O., Eigenwillig, A., Blais, M., Zelle, D., Pfeifer, J., Sanchez-Gonzalez, A., Li, W. L. S., Abu-El-Haija, S., Battaglia, P., Bulut, N., Halcrow, J., Gonçalves, D. A. F. M., Gonnet, P., Jiang, L., Kothari, P., Lattanzi, S., Linhares, A., Mayer, B., Mirrokni, V., . . . Perozzi, B. (2022, July 7). TF-GNN: Graph Neural Networks in TensorFlow. arXiv.org. https://arxiv.org/abs/2207.03522