LiDAR-Based Environmental Object Classification System

Design Document

Team: sdmay24-31
Client: Ahmad Nazar
Advisers: Mohamed Y. Selim
Team Dr. Members/Roles:
Ella Rekow
Sachin Patel
Zachary Schmalz
Anuraag Pujari
Ryan Sand
Daniel Rosenhamer

Team Email: sdmay24-31@iastate.edu
Team Website: https://sdmay24-31.sd.ece.iastate.edu/

Revised: 12/03/2023 v2.0.1

Executive Summary

Development Standards & Practices Used

- Requirement Analysis:
    - Clearly define the objectives and requirements of the deep learning model.
    - Ample accuracy and response time for detections 3d point cloud
- Version Control:
    - Use a version control system (e.g., Git) to manage code changes and collaborate with a team.
- Code Organization:
    - Follow a modular and organized code structure.
    - Adhere to coding standards and style guides.
- Documentation:
    - Provide comprehensive documentation for code, algorithms, and data preprocessing steps.
    - Document assumptions, limitations, and dependencies.
- Code Reviews:
    - Conduct regular code reviews to identify issues, improve code quality, and share knowledge within the team.
- Model Validation and Evaluation:
    - Implement cross-validation techniques to assess model performance.
    - Use appropriate metrics for evaluating the model's accuracy, precision, recall, etc.
- Hyperparameter Tuning:
    - Systematically tune hyperparameters to optimize model performance.
    - Leverage tools like Grid Search or Random Search for hyperparameter optimization.

- Monitoring and Logging:
    - Implement logging mechanisms to track model training progress.
    - Set up monitoring for model performance in production.

Summary of Requirements

- Introduce and train a machine learning model for LiDAR
    - Modify or create an algorithm to detect objects with greater accuracy than current public solutions
- Address the issue of non-standardization in LiDAR data with a training model.
    - Research varying LiDAR data sets
    - Develop a method to match the sets efficiently
    - Ensure compatibility between different LiDAR models
    - Develop an optimized solution that works across a variety of LiDAR devices.
- Publish our data and information via academic paper
    - Write and review academic paper to showcase our data collection
    - Describe the modified or created algorithm
    - Post our data collected at Iowa State University for public use

Applicable Courses from Iowa State University Curriculum

- Com S 228: Introduction to Data Structures
- Com S 252: Linux Operating System Essentials
- Com S 311: Introduction to Design and Analysis of Algorithms
- Com S 321: Introduction to Computer Architecture and Machine Level Programming
- Com S 363: Introduction to Database Management Systems
- Com S/S E 309: Software Development Practices
- S E 317: Introduction to Software Testing
- Cpr E 288: Embedded Systems Introduction
- Cpr E 388: Embedded Systems II: Mobile Platforms
- Cpr E/SE 329: Software Project Management
- ENGL 314: Technical Communication

New Skills/Knowledge acquired that was not taught in courses

- Machine Learning
    - Deep Learning
    - Training with accurate data additions
    - Ability to develop ML models for LiDAR data enhancement
- Working with Data Sets
    - Understanding good/clean data
    - Assess LiDAR data quality and identify discrepancies
- LiDAR setup and management skills
    - Initialization and configuration
    - ROS (Robot Operating System)
    - Data Collection Execution
    - Experience with LiDAR data visualization and manipulation software
    - Knowledge of techniques for merging LiDAR data from multiple sources or models.
    - Understanding of data registration and alignment methods

Table of Contents

List of figures/tables/symbols/definitions

Figures:

Tables:

1 Team, Problem Statement, Requirements, and Engineering Standards

1.1 Team Members

Ella Rekow

Sachin Patel

Zachary Schmalz

Anuraag Pujari

Ryan Sand

Daniel Rosenhamer

1.2 Required Skill Sets for Your Project

LiDAR Technology Expertise:

- We need In-depth knowledge of LiDAR technology, including its principles, components, and applications.
- Familiarity with different LiDAR models and manufacturers.
- Knowledge of pros and cons of different LiDARs (3D compared to 360 degree LiDAR)

Data Processing and Analysis:

- Proficiency in data processing and analysis techniques specific to LiDAR data.
- Ability to manipulate and interpret LiDAR point cloud data.
- We need to be able to use Python to translate data and interpret the original types of data recorded

Programming and Software Development:

- Experience with LiDAR data visualization and manipulation software.

LiDAR Data Fusion:

- Knowledge of techniques for merging LiDAR data from multiple sources or models.
- Understanding of data registration and alignment methods.

Machine Learning and AI:

- Familiarity with machine learning algorithms for LiDAR data classification and feature extraction.
- Ability to develop AI models for LiDAR data enhancement.

LiDAR Calibration:

- Proficiency in calibrating LiDAR sensors to ensure accurate data capture.
- Experience in calibrating multiple LiDAR models for consistency.

Data Quality Assurance:

- Skill in assessing LiDAR data quality and identifying discrepancies.
- Knowledge of data validation and error correction techniques.

Cross-Compatibility Testing:

- Ability to test LiDAR solutions across different LiDAR models to ensure compatibility.
- Experience with optimizing solutions for various LiDAR setups.

Quality Assurance and Testing:

- Capability to perform rigorous testing and quality assurance so that the optimized solution works reliably across different LiDAR devices.

Organization and Documentation Skills

- Ability to accurately describe processes and designs to ensure they are documented correctly
- Capability to manage time and prioritize tasks so that team members can be efficient

Team Management and Leadership

- Knowledge of how to work with other team members and lead based on identified strengths and weaknesses

Communication and Collaboration

- Ability to communicate clearly and effectively with team members
- Experience planning and discussing designs and ideas with other team members

1.3 Skill Sets covered by the Team

LiDAR Technology Experience:

- Daniel Rosenhamer - LiDAR implementation for object detection (CprE 288)
- Sachin Patel - 2D LiDAR obstacle detection for autonomous robot traversal
- Ryan Sand - 2D LiDAR path creation in Autonomous Robotics

Data Processing and Analysis:

- Anuraag Pujari - DS 201,202 and internship as data engineer
- Ryan Sand - data display and transforming training at RSM

Programming and Software Development:

- Zach Schmalz - Various COM S/SE courses as well as Web/Mobile internship at Genova Technologies
- Sachin Patel - COM S courses, three software engineering internships, personal projects
- Ryan Sand - Several COMS and SE courses and internships

Machine Learning and AI:

- Anuraag Pujari - DS 202, Undergraduate research assistant for a machine learning project
- Ella Rekow - Personal research into the application and workings of neural networks.
- Sachin Patel - Currently taking COM S 474

Sensor Integration:

- Ella Rekow - Integrated sensors to utilize in FIRST Robotics Challenge in high school
- Ryan Sand - research into 2D LiDAR and Inertial Measurement Unit sensor integration for Robotics Club
- Daniel Rosenhamer - Experience implementing sonar, infrared, and LiDAR sensors
- Sachin Patel - Robotics sensor integration (Cardinal Space Mining Club)

Cross-Compatibility Testing:

- Ryan Sand - SE 317
- Zach Schmalz - SE 317 and industry regression testing for internship at Genova Tech
- Ella Rekow - SE 317 and industry level testing at Netsmart

Quality Assurance and Testing:

- Ryan Sand - SE 317 and work at RSM
- Ella Rekow - SE 317 and industry level testing at Netsmart
- Daniel Rosenhamer - Testing experience at John Deere

Organization and Documentation Skills:

- Daniel Rosenhamer - Actively use Git, GitLab, Confluence, and Jira for personal projects, university projects, and work projects.
- Ryan Sand - Git and DevOps, as well as experience as the Documentation Manager on my High School FIRST Tech Challenge team
- Sachin Patel - Git, Trello, ticket managers
- Zach Schmalz - Git, Atlassian products for internships, GitLab for university projects
- Ella Rekow - Atlassian products, Documentation for information technology group at Iowa State, software documentation for numerous classes

Team Management and Leadership:

- Anuraag Pujari - IE 470
- Daniel Rosenhamer - Programming lead on high school FIRST Robotics Challenge team. Team lead in CprE 288 and ComS/SE 309.
- Ryan Sand - Lead of Robotics Autonomous Snowplow team
- Ella Rekow - Lead learning community for Women in Tech, lead programmer in robotics team, project manager for intern project
- Sachin Patel - Software and electrical lead of Cardinal Space Mining Club for 2 years

Communication and Collaboration:

- Zach Schmalz - SE 309/other group project course, and working in a small group at Genova
- Ryan Sand - group projects in coursework and worked in small groups in internships
- Ella Rekow - Numerous class projects, Managing student workers at the Iowa State Solution Center, Managing learning community for Women in Tech
- Daniel Rosenhamer - Team experience through Com S/SE 309, SE 329, CprE 288, CprE 381, and four internships on various development teams.
- Sachin Patel - team projects in clubs, courses, and internships
- Anuraag Pujari- lots of group projects through courses at Iowa State

1.4 Project Management Style Adopted by the team

Our team has opted for a hybrid approach, combining elements of both AGILE and waterfall development methodologies for our project. This choice aligns well with the nature of our solution, which revolves around a research topic. In practice, this means we will engage in iterative development, allowing us to make incremental progress and adapt as needed. However, certain aspects of the project will require a more traditional, linear approach, where we must fully complete specific tasks before moving on to the next stage. We will use Trello to manage, organize, and track our tasks efficiently. Additionally, GitLab will serve as our platform for issue tracking and version control, ensuring seamless collaboration and code management throughout development.

1.5 Initial Project Management Roles

- Ella is responsible for team organization and/or management, client interaction, and as keeper of Documentation/Secretary.
- Dan will be responsible for managing the communication mediums, LiDAR subject matter expert
- Anuraag will be the acting Data Manager and architect
- Sachin will be the Deep Learning Model subject matter expert
- Ryan will be the Data Collection Lead and will assist with the organization
- Zach will be the regression testing lead as well as the QA manager
- Note these are the initial assignments and may be modified throughout the project's progression to better fit the needs of our client, group, and solution.

2 Introduction

2.1 Problem Statement

There is a lack of 3D LiDAR object classification machine learning models that work with multiple LiDAR data formats; are scalable in terms of the model size, number of object classes, and the ease of adding features; and are efficient in real-time applications.

2.2 Requirements & Constraints

Functional requirements:

- (1) Development of an object classification training model for LiDAR sensors
    - Object detection/classification accuracy of >75% within 500ms
- (2) Creation of a labeled dataset suitable for object detection
    - Ensure comprehensive labeling of all different object types
- (3) Compatibility assurance with various LiDAR sensor data types
    - Should work with at least 3 data types, including: .lvx, .ply, .las

Resource requirements:

- (4) Sufficient computational resources for storing the collected data, training, and testing the model
    - Graphics card required of RTX 3060 or better
        - Focus on ensuring high-quality data processing and model training
    - 16GB Ram required
        - Requires a minimum of 16GB RAM for efficient operation and good refresh rates
        - Aim to facilitate smooth data processing and real-time model responses

UI requirements:

- (5) Published dataset is hosted on GitHub in an easily accessible/navigable manner
    - Host the dataset on GitHub with an intuitive README
    - Ensure that users can easily retrieve the dataset
    - The dataset has clear and unambiguous details of the model

Performance requirements:

- (6) Necessary that our model can classify objects with >75% accuracy
- (7) Model should achieve identification within a 500 ms response time

Legal requirements:

- (8) Adherence to data privacy and ethical standards
    - Safeguard and maintain the confidentiality of personally identifiable information
- (9) Restrictions on collecting sensitive information such as face details and license plates and compliance with IEEE standards

Maintainability requirements:

- (10) Necessitates ongoing model maintenance and documentation for future reference and collaboration
- (11) Create the model in a way that it is scalable so that it has the ability to work well with wider ranges of objects and larger datasets
- (12) Well documented so people can have an easier time using it

Testing requirements:

- (13) Rigorous testing and validation of the object classification system is also essential. Testing is covered more in the testing-specific section of the document

2.3 Engineering Standards

- ISO/IEC 20889:2018 - Preserve personally identifiable information
- IEEE 1588-2008 (PTPv2), PPS (Pulse Per Second)
- IEEE 1451 family of standards
    - These standards collectively create a framework for smart transducers and sensors.
- Agile development process
    - Will utilize Agile to facilitate workflow and promote effective collaboration and documentation

2.4 Intended Users and Uses

The outcomes of this project are poised to cater to a diverse spectrum of industries, ranging from research specializing in visual sensing to security systems in need of heightened sensor technology resilience. It extends its impact to individuals exploring innovative applications of obstacle detection and avoidance, which has many applications in the field of robotics for path planning. Moreover, LiDAR object detection is useful for deriving the number of objects, for example, the number of trees in a forest or students in a classroom. The project's ripple effect extends to improving autonomous vehicles for identifying obstacles and avoiding collisions. There are also many drone applications for surveying land and identifying key objects for which this project is useful.

3 Project Plan

3.1 Task Decomposition

TASKS:

- Task 1 - Research Machine Learning
    - Research various LiDAR technologies for project assessment(Req 3, 10)
        - Research different types of LiDARs to determine if livox-mid 40 satisfies the needs to develop a model.
    - Evaluate different machine learning models with similar applications such as camera detection models (Req 1, 2, 5, 6, 7, 12, 13)
    - Explore and review machine learning models similar in scope and application to the project's requirements. (Req 1, 2, 5, 6, 7)
- Task 2 - Collect data with LiDAR and camera
    - Take scans with the LiDAR to collect data to test with machine learning model (Req 4, 9, 10)
    - Discover online sources that contain different types of LiDAR data to consider for usage against our machine learning model (Req 3, 10)
        - Online datasets can also be used as reference against our own data for how it should be formatted, processed, etc.
    - Take a high diversity set of data, capturing all different units we would like our model to identify (cars, pedestrians, cyclists, buses) (Req 4)
        - Diversity refers to how we are able to collect data in different ways by using different angles for the point of view, like setting the LiDAR up on a rooftop. This also refers to how the traffic level of data collected is important. Some locations will have few pedestrians/cars, while others have many. This is something that must be taken into consideration when collecting data
- Task 2 - Process Data
    - Encode and format data to prepare for processing by neural network (Req 2, 3, 5)
    - Label data by classifying and labeling objects within the LiDAR data to use for training of machine learning model(Req 2, 5)

- Task 3 - Develop and Train Model
    - Research different neural networks to consider for the development of our own model (Req 1, 5)
    - Explore data compression options (Req 10)
    - Configure and train model to meet requirements of a quality classification model (Req 1, 2, 5, 6, 7, 12, 13)
- Task 4 - Validate Model
    - Verify the milestones have been met (Req 1, 2, 6, 7, 13)
    - Check over and finish documentation (Req 12)
    - Revisit previous tasks if necessary

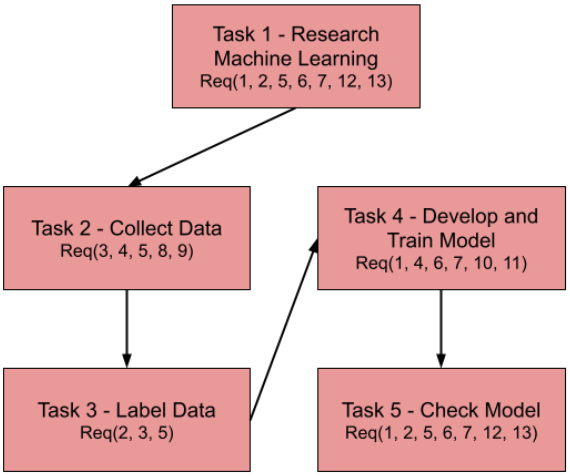


Figure 1: Task breakdown.

3.2 Project Management/Tracking Procedures

Our team has opted for a hybrid approach, combining elements of both AGILE and waterfall development methodologies for our project. This choice aligns well with the nature of our solution, which revolves around a research topic. In practice, this means we will engage in iterative development, allowing us to make incremental progress and adapt as needed. However, certain aspects of the project will require a more traditional, linear approach, where we must fully complete specific tasks before moving on to the next stage. This hybrid approach allows us to benefit from AGILE's flexibility and responsiveness, ensuring we can incorporate feedback and adjust our course based on evolving requirements. Simultaneously, the waterfall elements provide structure and a clear roadmap for certain aspects of the project, particularly those that involve well-defined and sequential tasks.

As for tracking progress throughout the course of this and the next semester, our team will use Trello to manage, organize, and track tasks efficiently. Trello's visual board system will enable us to have a clear overview of our project's status, easily assign and monitor tasks, and ensure everyone is on the same page regarding ongoing activities. In addition to Trello, GitLab will be employed as our platform for issue tracking and version control. GitLab's version control capabilities will help us manage our codebase effectively, allowing multiple team members to collaborate seamlessly while keeping track of changes. The issue-tracking feature will assist in identifying, prioritizing, and resolving any problems or tasks that arise during development.

This combination of Trello and GitLab will contribute to a well-organized and transparent project management process, facilitating efficient collaboration and ensuring that our team stays on track to achieve our project goals.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

METRICS:

1. **Number and size of quality data sets**
2. **Model accuracy percentage**
3. **Real-Time Model detection speed in milliseconds**
4. **Precision Recall**
5. **Confusion Matrix**

MILESTONES:

1. Task 1 - Research Machine Learning
    - 1.1. Develop a basic understanding of machine learning
        - **1.1.1. Review multiple academic papers and other online sources such as GitHub repositories with client**
        - **1.1.2. Understand various methods to resolve errors in predictions from precision-recall and confusion matrix metrics**
    - 1.2. Explore machine learning frameworks and libraries
        - **1.2.1. Create a prototype using TensorFlow**
        - **1.2.2. Identify at least one existing closely related model**

2. Task 2 - Collect Data
    2.1. Take scans with the LiDAR and camera
        2.1.1. The LiDAR scans will be completed once per month in different locations
        2.1.2. We will use 75 - 100 GBs of diverse data sets
    2.2. Discover online sources that contain different types of LiDAR data
        2.2.1. We will discover online sources of 20 additional data sets
    2.3. Take a high diversity set of data, switching up the locations, angles, and traffic levels
        2.3.1. The data sets will be narrowed down to 15 diverse sets to be used, including our own
3. Task 3 - Process Data
    3.1. Narrow down to usable, quality datasets
        3.1.1. There will be 15 diverse datasets to be used
    3.2. Segment and transform LiDAR data based on areas of interest
        3.2.1. Find areas of interest being the different classifications and different scenarios
    3.3. Label datasets manually in MATLAB
        The manual labels will become ground truth data due to being completely correct
4. Task 4 - Develop and Train Model
    4.1. Research different neural networks
        4.1.1. We will discover 10 research papers that describe the many neural networking possibilities
    4.2. Build an improved model based on OpenPCDet
    4.3. Configure model to meet requirements of a quality classification model
        4.3.1. The model configuration will provide the basis for the model to track objects with 75%+ accuracy and detect objects in 500ms

5. Task 5 - Validate Model

5.1. Verify the milestones have been met

5.1.1. The model will be trained to track objects with 75%+ accuracy and detect objects in 500ms

5.1.2. Confirm high coverage of scenarios

3.4 Project Timeline/Schedule

Task	Duration	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15
Research Machine Learning	100h															
Collect data with LiDAR and camera	60h															
Process Data	105h															
Develop and Train Model	90h															
Validate Model	90h															

Task	Duration	Week 16	Week 17	Week 18	Week 19	Week 20	Week 21	Week 22	Week 23	Week 24	Week 25	Week 26	Week 27	Week 28	Week 29
Research Machine Learning	100h														
Collect data with LiDAR and camera	60h														
Process Data	105h														
Develop and Train Model	90h														
Validate Model	90h														

Figure 2: Gantt chart breakdown of the project. The task of researching machine learning has a duration of 100 hours over weeks 1-10. The task of collecting LiDAR and camera data is 60 hours over weeks 10-15. The task of processing the data has a duration of 105 hours over weeks 14-20. The task of developing the model and training the model has a duration of 90 hours over weeks 20-25. And finally, the task of validating the model has a duration of 90 hours over weeks 25-29.

Our schedule plan contains our five main tasks over the time allotted to our project. We plan on wrapping up our data collection phase of the project by the end of week six and will be working concurrently on developing the model during that time. The model will be configured by the end of week 11, and we will begin to label our data sets as soon as we finish collecting them in week 7. At the start of week 14, our group will begin to train our model using the data and configuration choices made in the previous two tasks. Finally, once that task is done in week 20, we will proceed to the final phase of our project, which involves testing our model. We left plenty of time for this task to help ensure we can revisit previous tasks if needed to meet our milestones during our tests.

3.5 Risks And Risk Management/Mitigation

RISKS:

- Task 1 - Research Machine Learning

(0.1) While researching machine learning, important issues include practical problems arising from resource intensiveness, which necessitates large computational resources and related financial and environmental costs. Sensitive data processing must be done carefully due to security concerns, which include vulnerability to adversarial attacks and privacy breaches. The absence of universal standards in the rapidly evolving ML field contributes to reproducibility challenges, while ethical responsibilities necessitate vigilance against unintended consequences. Continuous learning, regulatory compliance, and human factors, such as effective collaboration, round out the landscape. In addressing privacy concerns, our project is committed to ensuring that LiDAR scanning will not involve identifying individuals through recognizable features or personal data. Rigorous measures will be implemented to prioritize privacy and uphold ethical standards in LiDAR technology. Despite these challenges, proactive risk management, ethical considerations, and collaborative efforts within the research community are pivotal for steering ML development responsibly.

- Task 2 - Collect Data

(0.5) While we currently possess a certain amount of data, we need to expand our dataset through additional data collection efforts. Acknowledging a potential risk associated with circumstances that might impede our ability to record sufficient data is imperative. This risk becomes particularly salient during the colder months when there tend to be fewer individuals walking around in large groups. Moreover, the prospect of our sole LiDAR system encountering technical issues poses an additional challenge. To effectively mitigate these concerns, a proactive strategy involves meticulous planning to ensure the acquisition of at least a few sets of our data. This foresighted approach becomes crucial in circumventing potential disruptions. Furthermore, to diversify our dataset sources, we can enhance our reliance on research to identify and incorporate other publicly available datasets, thus fortifying our data pool against unforeseen challenges.

- Task 3 - Label Data Set

(0.1) The current step presents minimal risks, primarily attributed to the fact that we already possess the required data and the task at hand involves manual labeling. A relatively low inherent risk level characterizes this process as it allows for direct visual detection of any potential inaccuracies during the labeling process. The capability to detect errors in real-time positions us advantageously, enabling prompt corrective actions and minimizing any potential loss of time. This meticulous approach underscores our commitment to precision in data labeling, ensuring that the labeled dataset meets the required standards. The real-time error detection mechanism serves as a built-in quality control measure, further enhancing the robustness of this step within the overall project workflow.

- Task 4 - Develop Model

(0.5) The primary risk inherent in developing the model task lies in the potential expenditure of time on configuring the model in a manner that fails to meet our accuracy and time-based benchmarks. It is essential to delve into the intricacies of this risk by acknowledging that achieving the desired model configuration might not materialize seamlessly in the initial attempts. This inherent uncertainty is duly recognized and incorporated into our project schedule. The acknowledgment that the perfect model creation is not an immediate expectation underscores our preparedness for an iterative process, allowing for multiple iterations and adjustments as needed to align with our defined benchmarks. This nuanced understanding of the risk underscores our commitment to a flexible and adaptive approach, ensuring that the iterative nature of model development is factored into our project timeline. The potential risk in this step revolves around incorrectly training the model, stemming from misconfigurations identified in Task 2 or the utilization of flawed datasets from public sources or our own data collection efforts. It's essential to delve deeper into this risk by recognizing that while not considered massive, it can significantly impact the accuracy and performance of the model. As previously indicated, our project schedule has been designed with the expectation of iterative refinement, acknowledging the likelihood of needing multiple iterations to reach our targeted milestones. This risk mitigation strategy inherently allows flexibility and adaptation during the model training. To further bolster our risk mitigation efforts, the meticulous approach taken in Task 2 becomes crucial. By dedicating time to finding and recording accurate datasets, we proactively minimize the risk associated with the usage of faulty data. This preemptive strategy is a foundational measure to enhance the quality and reliability of the datasets employed during model training, aligning with our commitment to achieving the desired benchmarks.

- Task 5 - Validate Model

(0.1) The risk associated with this task is relatively low, primarily centered on verifying whether the model performs according to the predefined specifications. Any deviations or issues encountered during this stage are more likely to be attributed to potential shortcomings in previous tasks. It's crucial to recognize that the only conceivable challenge lies in the possibility of conducting the tests incorrectly, leading to a potentially underestimated final model accuracy. Our proactive strategy to mitigate risk involves a thorough testing approach. By adhering to our planned methodology of testing the data in multiple ways, we aim to ensure a robust evaluation process. This multifaceted testing strategy aligns with our commitment to precision and safeguards against inaccuracies stemming from a singular testing approach. The emphasis on thorough testing underscores our dedication to delivering a model that meets and potentially exceeds our envisioned accuracy benchmarks.

3.6 Personnel Effort Requirements

Task	Hour Estimate	Explanation
Task 1 - Research Machine Learning	100 hours	We researched different LiDAR technologies to determine the best for our project. After that, we looked into different machine-learning models with similar applications and looked through them.
Task 2 - Collect data with LiDAR and camera	60 hours	Use the LIDAR to gather a reasonable amount of data points from different locations. We expect this to take the least time, even though we must monitor the LiDAR as it collects data.
Task 3 - Process Data	105 hours	Before creating a strong machine learning model, we concentrated on enhancing and reducing the datasets to ensure they were high-quality and useful. Choosing datasets that most accurately reflected the situations and settings pertinent to our goals required a careful curation procedure. Then, we started working on the difficult task of preprocessing LiDAR data, using segmentation methods to draw attention to particular regions of interest that were vital to our investigation. It was essential to convert this data into a voxel format to improve its viability for further machine-learning procedures. Using MATLAB, the dataset labeling process was carried out manually with meticulous attention to detail. A strong basis for the model training stage was established by manual labeling, which guaranteed precision and applicability in identifying and annotating objects within the dataset.

Task 4- Develop and Train Model	90 hours	In this task, we will develop a model that we deem suitable and then train the model created with the labeled data set. This task presents a large learning curve.
Task 5 - Validate Model	90 hours	We will run the model to ensure it fits our accuracy requirements, using the test and actual data in a 20-80 split. Most of the time allocated here is if we need to retune our model after our tests to ensure we reach our milestones.

Table 1: Personal Effort breakdown of each of the 5 tasks: Research Machine Learning, Collect data with LiDAR and camera, Process Data, Develop and train model, and Validate model

3.7 Other Resource Requirements

This section includes tools, libraries, and other various resources not addressed above to be utilized in our project.

- Cloud Compare: CloudCompare is a 3D point cloud (and triangular mesh) processing software. It was originally designed to compare two dense 3D point clouds (such as the ones acquired with a laser scanner) or between a point cloud and a triangular mesh. It relies on a specific octree structure dedicated to this task. Afterward, it has been extended to a more generic point cloud processing software, including many advanced algorithms (registration, resampling, color/normal/scalar fields handling, statistics computation, sensor management, interactive or automatic segmentation, display enhancement, etc.). This tool will be utilized specifically in our project to visualize the point cloud we gather from the Livox Mid 40.

- DeepSense 6G Data: DeepSense 6G is a real-world multi-modal dataset that comprises coexisting multi-modal sensing and communication data, such as mmWave wireless communication, Camera, GPS data, LiDAR, and Radar, collected in realistic wireless environments. We will use this data when necessary to add to our already gathered data to ensure we have a good labeled data set to input.

- Google Suite: This tool includes Google Drive, docs, slides, and more. We have utilized this to create detailed documentation updated in real-time for this project.

- Gitlab: GitLab is a DevOps software package that can develop, secure, and operate software. This platform will be GitLab, a DevOps software package that can develop, secure, and operate software. We will use this tool to upload and update our code for this project.

- 3D Detection & Tracking Viewer: This project was developed to view 3D object detection and tracking results. It supports rendering 3D bounding boxes as car models and rendering boxes on images.

- Kitti Vision Benchmark Suite: Kitti contains a suite of vision tasks built using an autonomous driving platform. The full benchmark contains many tasks, such as stereo, optical flow, visual odometry, etc. This dataset contains the object detection dataset, including the monocular images and bounding boxes. We are using Kitti to reformat the data to make sure we can use multiple different LiDAR data inputs in our model

- Livox Mid-40: The Livox Mid-40 LiDAR sensor is incredibly cost-effective. It detects objects as far as 260 meters away [1] and uses an advanced non-repetitive scanning pattern to deliver highly accurate details in the FOV. A compact body enables users to easily embed units into existing designs for greater flexibility and performance. The Mid-40 has been mass-produced and is ready to ship immediately to facilitate uses in autonomous driving, robotics, mapping, security, and other areas, from small-batch testing to large-scale applications. The Livox LiDAR is what we will be using to gather all of the data on Iowa State's campus.

- Livox ROS Driver: livox_ros_driver is a new ROS package specially used to connect LiDAR products produced by Livox. The driver can be run under Ubuntu 14.04/16.04/18.04 operating system with ROS environment (indigo, kinetic, melodic) installed. Tested hardware platforms that can run livox_ros_driver include: Intel x86 CPU platforms and some ARM64 hardware platforms (such as Nvidia TX2 / Xavier, etc.).

- Linux Ubuntu 14.04/16.04/18.04: Operating system with ROS environment (indigo, kinetic, melodic) installed

- Livox Detection: Combined with the advantages of HAP, this detector can achieve better perception performance compared with the Horizon. Another improvement is adopting an anchor-free method inspired by CenterPoint to make the detector more flexible in dealing with multiple datasets.

- Livox Viewer: Livox Viewer is a computer software designed for Livox LiDAR sensors and Livox Hub. Users can check real-time point cloud data of all the Livox LiDAR sensors connected to a computer and can easily view, record, and save the cloud data for offline or further use. This is how we will initially visualize the data that we gather before we convert it.

- MATLAB: MATLAB is a programming and numeric computing platform used by millions of engineers and scientists to analyze data, develop algorithms, and create models.

- Lidar Toolbox: Lidar Toolbox provides algorithms, functions, and apps for designing, analyzing, and testing lidar processing systems. You can perform object detection and tracking, semantic segmentation, shape fitting, lidar registration, and obstacle detection. The toolbox provides workflows and an app for lidar-camera cross-calibration.

- Point Pillar: PointPillars is a method for 3-D object detection using 2-D convolutional layers. PointPillars network has a learnable encoder that uses PointNets to learn a representation of point clouds organized in pillars (vertical columns). The network then runs a 2-D convolutional neural network (CNN) to produce network predictions, decodes the predictions, and generates 3-D bounding boxes for different object classes, such as cars, trucks, and pedestrians.

- OBS: OBS Studio is a free and open-source, cross-platform screencasting and streaming app.
- OpenTopography: OpenTopography facilitates access to topography data, tools, and resources to advance our understanding of the Earth's surface, vegetation, and built environment. We provide web-based access to high-resolution (meter to sub-meter scale), regional, and global (2-90 m resolution) topography data acquired with lidar, radar, and photogrammetry technologies. The data are co-located with on-demand processing tools to generate derivatives and visualizations that are particularly useful for Earth-Science applications.
- OpenPCDet: OpenPCDet is a clear, simple, self-contained open-source project for LiDAR-based 3D object detection. This project utilizes a Point-Voxel Region-based Convolutional Neural Networks (PV-RCNN) [5]
- Power source: Specifically a standby uninterruptible power supply (UPS) with 600VA / 360W battery backup
- PyLas: Python library for lidar LAS/LAZ IO. LAS (and its compressed counterpart LAZ) is a popular format for lidar point clouds and full waveform. Pylas reads and writes these formats and provides a Python API via Numpy Arrays.
- SciKit-Learn: scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms, including support-vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
- TensorFlow: TensorFlow is an open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but focuses on training and inference of deep neural networks.
- Keras: Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library

4 Design

4.1 Design Content

Our project's design content is centered around deciding on and implementing the best options for a neural network that labels objects from input data collected by a LiDAR system. The goal of the project is to provide an effective deep learning model to label the data that is taken at Iowa State University that can be applied to other data sets taken from different LiDAR systems as well.

4.2 Design Complexity

Our project showcases a significant level of technical intricacy, evident through its multifaceted nature, involving various interconnected systems with the primary objective of real-time object identification utilizing a Lidar sensor. While we have chosen the Livox Mid-40 sensor for our specific application, our overarching goal is to compile an extensive dataset to serve as the cornerstone for training deep learning models dedicated to Lidar-based object recognition, regardless of the sensor in use.

The technical complexity of our project is further emphasized by the unique challenges it tackles in an uncharted area within the field. Existing literature, as cited by the IEEE Sensors Council [2], underscores the sparse nature of Lidar point cloud data and the associated hurdles in network processing. This necessitates exploring innovative approaches and solutions, making a valuable contribution to the field's body of knowledge.

The scarcity of readily available models, as mentioned in the IEEE Sensors Council report, signifies that our project is at the forefront of research. Our foray into deep learning on point sets, as also acknowledged by Stanford University, underscores our dedication to advancing knowledge in this domain.

Finally, one of the most apparent reasons why this project is complex is the fact that we are dealing with unordered 3D point cloud data, as opposed to ordered datasets. This adds complexity because we need to account for how the points are located in three-dimensional space without relying on the data being preliminarily sorted into sections resembling 2D arrays. Our algorithm will need to function without advanced knowledge of where each data point in the point cloud is in the order of points to be processed.

Taken together, these factors demonstrate that our project contains sufficient complexity. Our collaboration with a Ph.D. candidate to complete this project serves as a testament to its advanced nature, cementing its position as a technically intricate undertaking.

4.3 Modern Engineering Tools

1. **Sensor Technology:** We are using a Livox Mid-40 LiDAR sensor to capture various 3D data for our project. The LiDAR sensor will be used in parallel with a camera in order to accurately label the objects recorded by the LiDAR. The data recorded from the LiDAR is one of the most important components of our project. It will be used to train our object classification model.

2. **Data Visualization and Processing Tools:** Our project utilizes various tools to visualize the data being collected, including Livox Viewer, Cloud Compare, PyLas, and OpenPyLivox. Each tool allows us to visualize the data in a 3D platform and interact with settings to manipulate the data shown. This will help us process the data before labeling it.
3. **Data Labeling:** Once the data has been recorded, we will need to label it before training our machine learning model with it. The data will be labeled in two different ways, which will produce two different sets of labeled LiDAR data. The first labeled data set will come from manually labeling our data using the Lidar Toolbox in Matlab. The second labeled data set will come from automatically labeling our data using the YOLO V4 Network.
4. **Deep Learning Frameworks:** The two labeled data sets will be fed into our deep learning models. We will be using TensorFlow and Keras as our deep learning models. These models combine the labeled data and produce a deep 3D visual data set. This will then be processed again before it is a completely labeled set.

4.4 Design Context

Area	Description	Examples
Public health, safety, and welfare	This project can be able to provide assistance to safety through the usage of LiDAR in several different use cases, the main one being vehicles and detection systems when it comes to detecting pedestrians or other vehicles on the road. The deep learning network could also be tweaked to identify other objects and could help remove human workers from dangerous scenarios.	Increase road safety by detecting vehicles more accurately in systems such as stoplight car detection or even autonomous vehicles Could possibly identify if an accident occurred Has potential to monitor dangerous locations without human supervision, which would keep human operators safer while ensuring privacy
Global, cultural, and social	Our project does not impact these categories in a substantial way, however some privacy concerns could be eased by implementing our project	Replacing regular video cameras with a LiDAR system could improve the privacy of the individuals being recorded by the system
Environmental	There are no outstanding environmental impacts compared to the normal environmental damage caused by the manufacturing of the LiDAR and related items. The laser itself is not dangerous because the LiDAR we are using is a class I laser. However, there may be environmental applications for the object detection algorithm, potentially for positive impact.	Detecting the number of trees or animals in a forest over time Observing erosion or other land alterations with concrete data

Economic	This project will be available publicly and may save companies work hours for developing something similar. The cost of the system is comparable to a security camera system and will likely provide consumers and companies with alternatives for regular cameras depending on their needs, as the software to detect objects is more readily available and easier to use. This is not a product we are selling, and so while it might slightly improve LiDAR sales, it will not be marketed.	The learning model could entice businesses or researchers to use LiDAR systems instead of others, such as a regular camera setup Because it will be made publicly available, the model could assist consumers with a lower budget because it will work with different LiDAR types

Table 2: Project Impact Overview including Public Health, Safety, and Welfare, Global, Cultural, and Social, Environmental and Economic impacts.

4.5 Prior Work/Solutions

While research on the use of Lidar as a data method for deep learning is available, as evidenced by papers such as the IEEE Sensor Council's "Deep 3D Object Detection Networks Using LiDAR Data: A Review" from 2021 and "BirdNet: a 3D Object Detection Framework from LiDAR Information," presented at the 21st International Conference on Intelligent Transportation Systems (ITSC) in 2018[3], limited exploration in this field is due to the high cost and inconsistency associated with Lidar technology. The abstract of this IEEE paper mentioned earlier acknowledges this challenge: "Recently, deep neural networks have been developed to extract powerful object features from sensor data. However, the sparsity of Lidar point cloud data poses challenges for network processing." This highlights that even prominent engineering communities recognize the hurdles in this concept.

Nonetheless, a paper titled "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation"[5] presents systems that can take in a voxel or raw point cloud to classify information, offering a broader solution to the issue. Additionally, Livox has a repository on GitHub, Livox Detection-simu, with the goal of being a real-time identification model. It does, noticeably, have a number of major issues and is no longer maintained. However, challenges remain, particularly when dealing with Lidar data, given the inconsistency in the format transmitted by the sensor.

4.6 Design Decisions

Our team has chosen to use the Lidar that is provided to us by our client. We reached this decision because we were comparing different LiDARs to one another and felt no need to buy a different one when the one we used could do everything we needed. Our team has identified multiple areas on campus that can be considered high-activity areas where we would want to gather data. Depending on the specific area, we have decided on a point where we would also place the Lidar for the best results. Next, we need to decide on the amount of data we need from these areas. We do not have a specific number on the size of data that we are planning to collect because we will need to collect the data and determine how useful it will be to our model. If we need more data, we can attempt to locate usable datasets from open-source online resources. However, this is a backup as we would like to gather the data ourselves to ensure its relevance and quality. The last thing we have is our overall system design, which is how our data will be processed into the model and which machine learning model to use.

We have decided to manually label the data using MATLAB and Optimized Point Pillar labeling. We decided against creating our own model from scratch and have decided to use the OpenPCDet object detection algorithm to help object recognition in our data. Then, we will run the trained model on our testing data and validate the results. We have decided to take 80% of the data to use for training and 20% for testing. We have also decided that the model should have an accuracy of 75% or greater to be deemed successful.

4.7 Proposed Design

Our team has worked with our client to determine the best path forwards while researching and learning more about the skills required to complete the project. We have reached the conclusion that the future design listed below in section 4.7.2 is the most robust plan out of the alternatives we discussed. We reached this conclusion after creating Design 0 in section 4.7.1 and testing it using MATLAB for labeling and determining if the OpenPCDet algorithm is a better alternative than creating a new model from scratch. We will likely need to continue to create more design iterations throughout the next semester as we learn more about the intricacies of our project.

4.7.1 Design 0 (Initial Design): Design Visual and Description

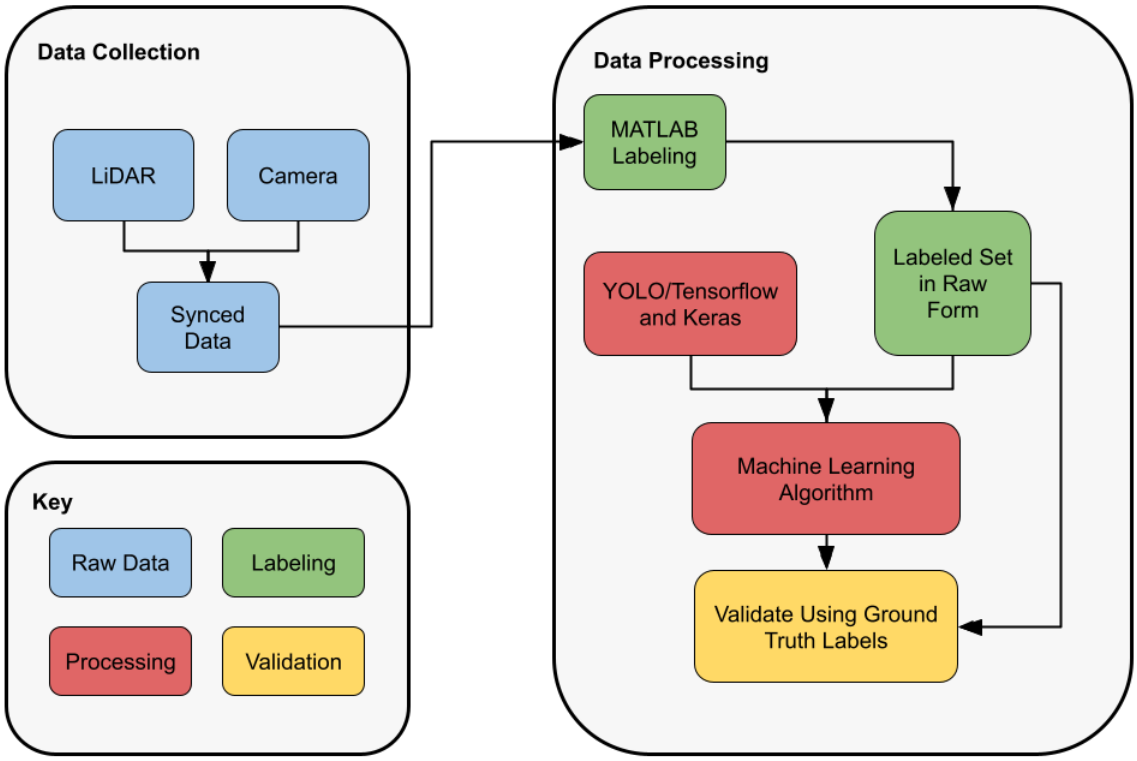


Figure 3: Initial design iteration of system. Data collection includes LiDAR and Camera data being synced. That is then fed into our Data Processing pipeline of MATLAB labeling to create a raw labeled data set. That also will be used for validation as ground truth labels. There will also be YOLO/Tensorflow and keras impacting the Machine Learning Algorithm that will be validated using the ground truth labels.

Our initial design was decided upon after consulting with our client and discussing the project's possibilities with him. This process took several meetings to finalize this design and all of its components. We have yet to have the opportunity to test the entirety of it, especially because this pipeline showcases the entire process our data will need to be sent through. However, we tested the data collection and the beginning of the data processing and labeling sections using MATLAB and OpenPCDet.

The first portion of our diagram is the Data Collection portion. Before we reach the first two components in this portion, we need to decide on where we want to collect data. In this step, we researched and brainstormed different locations for data collection across the Iowa State University campus. We considered the time of day and different angles possible to ensure that the data we collected would be helpful for both the testing and training of our learning model by finding a location with many objects to identify, such as cars and pedestrians. This helps the project meet many requirements, in particular, requirement 5 (External data will be collected around campus as needed) and must satisfy requirement 9 (Restrictions on collecting sensitive information such as face details and license plates and compliance with IEEE standards).

The first set of components in our block diagram is the recording set of modules. Once we decided on the locations where we wished to collect data, we needed to physically collect it using a combination of LiDAR data to train and camera footage to help us label the LiDAR data more accurately. Depending on the location and time, the amount of data we took at each location varied. However, the length averaged around an hour. We then took that data and combined it together so that the camera picture could overlap the LiDAR data. This is the first big step in satisfying requirement 2 (Creation of a labeled dataset suitable for object detection).

Our second set of modules following the data recording components is the labeling section. We will use a mixture of manual and automatic labeling to obtain a raw set of labeled data from the LiDAR and camera overlay. In our initial design, we looked into ways to completely automate the labeling process and planned to fall back on manually labeling the data in MATLAB if we could not find a method that works with our dataset. Like the previous section of components, the labeling group of modules satisfies requirement 2 (Creation of a labeled dataset suitable for object detection).

The next component group comprises the processing and training portion of our block diagram. In our initial design, we planned to use tools like the You Only Look Once (YOLO) algorithm and the Tensorflow library to create the best model we could for our dataset. This process would provide us with our final labeled dataset. This set of modules meets requirements 1 (Development of an object classification training model for LiDAR sensors), 2 (Creation of a labeled dataset suitable for object detection), and 3 (Compatibility assurance with various LiDAR sensor data types).

Our final step is the testing step, where the team will manually check the trained data against some data that we left specifically for the checking step (20% for checking, 80% for training). This meets requirement 13 (Rigorous testing and validation of the object classification system are also essential) and will need to meet the milestone set for the project to succeed. If we do not meet the milestone of 75%+ accuracy, the team will discuss and move back up the pipeline to a point further back and complete the steps again until our model is successful.

Functionality

The project will operate by allowing a user to input their LiDAR data into the neural network and receive the correctly labeled data based on what they want to be labeled. This will be as seamless as possible, although Design 0 shown above only showcases how we will create the neural network, not how the deep learning algorithm will be accessed and implemented in a user-friendly way. Our project's initial goal is to get the network created, and we will add extra features, such as a better way to manipulate or access the data, if we reach that goal.

This planned implementation is designed to meet all of our functional requirements. These include having the algorithm detect objects correctly with an accuracy greater than 75% within 500ms (1). It also will be scalable because of the use of the OpenPCDet model which will make it easy to add new objects in the future to the system if necessary (2). The addition of the data converting module from Design 0 to Design 1 will help ensure that our model can receive input in a variety of LiDAR data types, including .lvx, .ply, and .las (3). We will begin implementing this design next semester and make more design iterations as necessary to ensure we meet the requirements of our project.

4.7.2 Design 1 (Design Iteration)

As we look towards the future of this project, we have made some changes to the first iteration of our design that we will be implementing. These changes came from lessons learned after prototyping part of the initial design using MATLAB and OpenPCDet. The changes aim to make the project more feasible and accurate to the project requirements.

Design Visual and Description

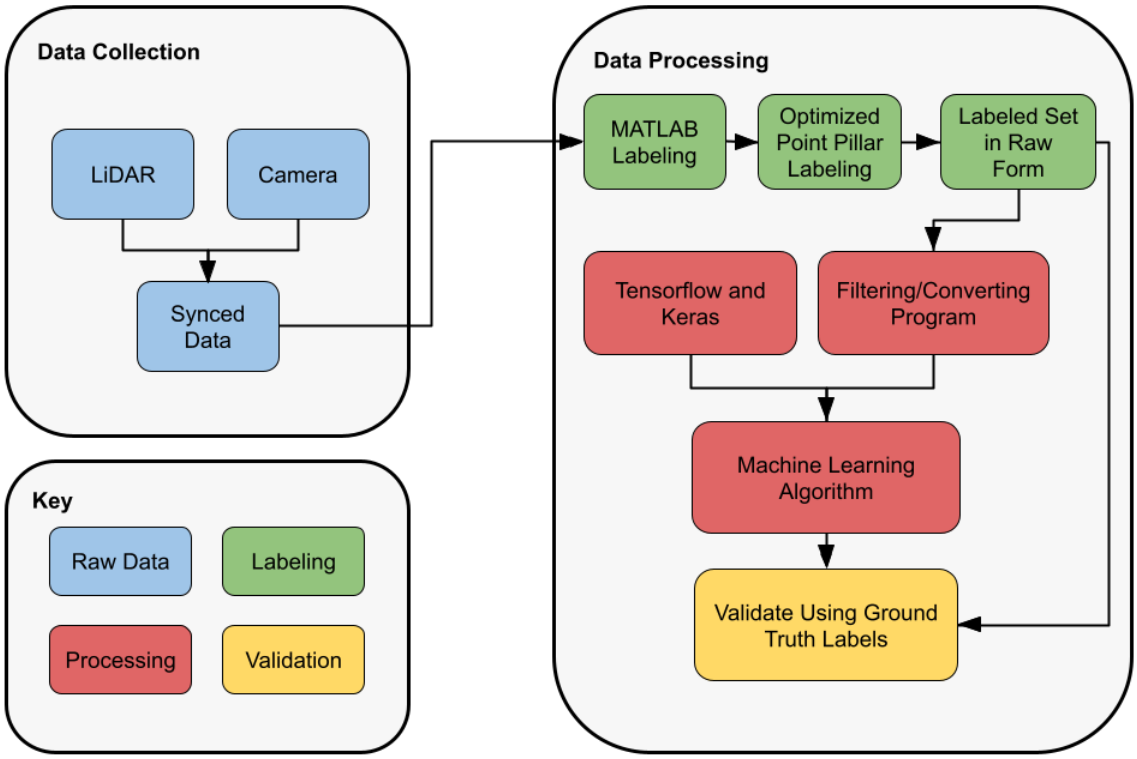


Figure 4: First design iteration of the system after the initial. Data collection includes LiDAR and Camera data being synced. That is then fed into our Data Processing pipeline of MATLAB labeling being put into Point Pillar to create a raw labeled data set. That also will be used for validation as ground truth labels. There will also be Tensorflow and Keras impacting the Machine Learning Algorithm that will be validated using the ground truth labels.

This design shows what we plan to change in Design 1 after our lessons learned from Design 0. The first change in this design is the introduction of the Optimized Point Pillar detection algorithm to assist us in labeling. We learned that the MATLAB process of labeling was prohibitively time intensive, and are planning on bolstering it with the Point Pillar labeling to create our ground truth values, which we will use to validate the algorithm's accuracy. The next change in the data processing portion of our design is the addition of a program or process we will create to attempt to convert or generalize the data being input from the labeling modules. This will help us ensure that the model can use as many different data types as possible and be as accessible as possible. It will also help provide a way for our data to be cleaned, if necessary, before using it in the model to avoid any unexpected issues. The final change is a decision to not use YOLO as a reference to create our own algorithm and instead modify the generic algorithm OpenPCDet to fit our needs. This decision was reached because of the complexity of attempting to create a model from scratch, being deemed unnecessary for the scope and eventual end result of our project. The requirements coverage is nearly identical to our Design 0. However, requirement 3 (Compatibility assurance with various LiDAR sensor data types) is much more directly addressed and likely improved upon as we are now looking to translate between proprietary LiDAR formats instead of being required to feed the data in standard formats such as PLY.

4.8 Technology Considerations

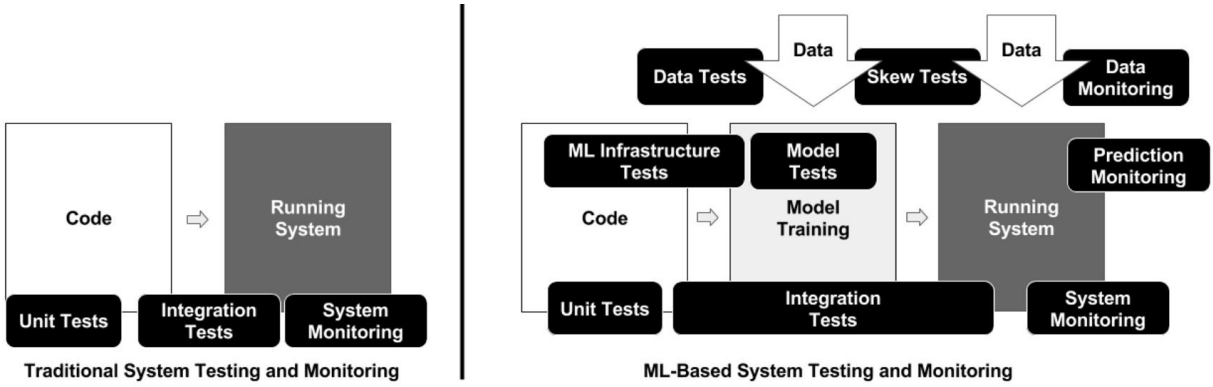
We have already made several decisions that balance different technological options for our project. The first decision was to continue to use the Livox Mid40 LiDAR our client had previously used instead of others, such as a Velodyne LiDAR model. While the decision was partly financial based, it was also reached because we believed that this LiDAR, being a 3D LiDAR with a range of 300 meters, would be the best suited for our project when compared to others with lesser ranges or full 360-degree capture capabilities. This is because we aim to detect and identify objects in certain locations, which requires 3D point capture. We also believed that a 360-degree LiDAR was not justified for the price of our use cases. Another decision we made was the decision to pivot from attempting to create our own machine learning model for our project to modifying the generic OpenPCDet used in the Livox Detection system. We discovered OpenPCDet because of our use of a Livox product and discovered it is a good base for our modifications. This choice will make the project more efficient because we do not need to spend time completely creating a new model. However, the downside is that we might not be able to make as precise of a model for our specific needs because we cannot form every part of it. We believe that the benefits of efficiency outweigh the negatives and will be able to determine if this is correct once we can begin fully implementing the design.

4.9 Design Analysis

Because of the complexity of our project, we were unable to fully implement the design in our first semester. However, we were able to test the data collection portion of our design, as well as part of the data processing portion. Because of those tests, we were able to make some decisions regarding how we want to proceed in the future. We believe that the parts of our design that we were able to test showcased promise for our final design and deliverable. There were a few complications with the data collection process, but our team collected sufficient data to conclude that the process outlined by our client would work as we scale our project upwards. We will need to wait until we can fully test the labeling and learning algorithm modifications to analyze our design completely, but we appear to be on track to produce our deliverable while meeting our requirements.

5 Testing

In machine learning, deploying object classification models demands a rigorous and well-structured testing strategy to ensure both production readiness and the reduction of technical debt. Drawing inspiration from the insightful paper, "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction" by Eric Breck et al. at Google [4], our approach integrates requirements and design with a comprehensive testing framework. This document outlines the overarching testing strategy, emphasizing the critical connection between system requirements, design considerations, and the adoption of specific testing instruments. The image found in the paper below shows the complex testing system of a machine learning model compared to a traditional software system. As we navigate the landscape of testing for deep learning models, particularly those focused on object classification, we recognize unique challenges that arise due to the inherent complexity of neural network architectures and the dynamic nature of real-world data. Our testing strategy addresses these challenges head-on, leveraging key insights from the ML Test Score rubric to guide the development of tests that span data quality, model robustness, monitoring, and governance. By aligning our testing efforts with these critical facets, we aim to establish a resilient testing framework that ensures our object classification deep learning model's



reliability, adaptability, and interpretability in diverse production scenarios.

Figure 5[4]: Machine Learning (ML) systems demand thorough testing and ongoing monitoring. A crucial distinction lies in the fact that, unlike manually coded systems (on the left), the behavior of ML-based systems is not easily predetermined. Instead, it hinges on dynamic aspects of the data and diverse choices made during model configuration.

5.1 Unit Testing

Traditional unit testing methodologies face unique challenges when applied to machine learning models due to the fundamentally probabilistic and data-driven nature of these models. Unlike conventional software, where deterministic inputs yield predictable outputs, machine learning models learn from data patterns, making their behavior dependent on the specific dataset they are trained on. Traditional unit tests, designed for deterministic code, struggle to encapsulate the inherent uncertainty and variability in machine learning predictions. Moreover, machine learning models often operate in high-dimensional spaces, making exhaustively testing all possible inputs impractical. Additionally, datasets' dynamic and evolving nature can introduce variability over time, further complicating the establishment of fixed test cases. Consequently, unit testing for machine learning requires innovative approaches that consider the probabilistic nature of predictions and adapt to the dynamic nature of data-driven models.

By creating a feature expectation schema, we can create a baseline to compare the results of our model. Similar to a unit test where expected outcomes are defined and compared, this method encodes intuitions about the data into schemas that can be automatically checked. For instance, height expectations for an adult human or word frequency distributions in English text are schema rules. These schemas serve as a set of criteria against which input data during both training and serving phases can be tested. The process involves constructing schemas by calculating statistics from training data, adjusting them based on domain knowledge, and refining them iteratively. Visualization tools like Facets can aid in analyzing data to inform schema creation, and invariants can be automatically inferred from the system's behavior, contributing to a robust unit testing framework for the machine learning algorithm.

The concept of data invariants in training and serving inputs is closely related to the idea that analyzing and comparing datasets is crucial for detecting problems in machine learning systems. While monitoring the internal behavior of a learned model can be challenging, examining the transparency of input data serves as the primary means to identify issues, especially when the world undergoes changes that may confuse the ML system. The previously constructed schema in the "Data 1" test measures whether the incoming data adheres to the expected schema. Significant divergences between the data and the schema trigger alerts, providing an early warning system for potential problems. Fine-tuning alerting thresholds is emphasized to balance false positives and false negatives, ensuring that alerts are useful and actionable in maintaining the system's reliability and performance.

Another “unit test” style we can apply is ensuring that our model reproduces the same results when retrained with the same information. The described method aligns with a possible unit test approach for a deterministic object classification model. The emphasis on reproducibility in training, where training on the same data should ideally yield identical models, simplifies reasoning about the system and aids in audibility and debugging. This determinism is particularly advantageous for diff-testing, ensuring that changes in the feature generation code, for instance, can be validated by verifying that the old and new code lead to the same model. The method acknowledges challenges in achieving complete determinism in model training, especially in non-convex methods like deep learning or random forests, due to factors such as random number generation and initialization order. The suggested solution involves seeding to handle random number generation but also highlights the importance of addressing initialization order, especially in multi-threaded or distributed systems. Additionally, the method suggests ensembling models to

mitigate non-determinism and enhance the robustness of the unit test, providing a more reliable and consistent evaluation of the deterministic object classification model.

However, the most traditional type of unit testing we will be applying is with the model specification code. Creating unit tests with model specification code for machine learning models involves defining a set of tests that validate the expected behavior and performance of the model according to its specifications, like typical software unit tests. This process typically begins with establishing clear and comprehensive specifications for the model, including input requirements, expected output, and performance metrics. Unit tests are then crafted to evaluate the model's correctness and reliability against these specifications. This may include testing the model's predictions on synthetic or predefined datasets, assessing its response to edge cases, and verifying that it meets specified accuracy or other performance criteria. The test suite should cover various aspects of the model's functionality, such as feature handling, training convergence, and robustness to variations in input data. By integrating model specifications directly into the unit testing framework, developers can systematically ensure that the machine learning model aligns with the intended requirements and performs reliably across different scenarios.

5.2 Interface Testing

Most of our projects will be tested using the unit test plan outlined above. However, there are still a few connections between different stages of the neural network and the data. This interface testing will be focused on smooth and effective interactions between the neural network and data inputs. The interface should be user-friendly for labeling and capturing data, facilitating easy navigation and adaptability for automation or semi-automation.

The current scope of our project does not include a sophisticated User Interface for our machine-learning algorithm. As a result, our only interface testing will be how effectively the data is transferred between the different stages of the neural network. In the future, if we are ahead of schedule enough for the project scope to be increased, we would need to add additional unit tests and test the software responsible for using our algorithm more traditionally.

5.3 Integration Testing

The critical path regarding our design would likely be training our machine learning model to effectively identify the objects within the Lidar data. Another path that could be seen as critical would be the quality of the training data. The training data must be of decent quality because the ability of the machine learning model depends on it. Potential issues of overfitting or underfitting the objects within some data should be assessed.

We will spend most of our time on the critical path mentioned above, which includes transferring the data from its collection method in our testing of a Webcam and the Livox Mid-40 LiDAR provided to us to the last stage of the machine learning process.

We will manually test the data at the end of the pipeline and use tools to perform sanity checks on the data at each process stage. This can be accomplished because the data will be checked against models that it will need to match to be allowed to proceed to the next stage of the pipeline. We will also propose limits on the age of the data to ensure that the data is not contaminated.

The manual tests can be conducted using Matlab to observe the data. This can help us use the Webcam data to sync the data streams up to determine if the machine learning model missed any data in the pipeline. We can also either make our checker or use a tool to automate checking if each data points to correct dates and is formed in a model that matches what the neural network would take in. These checks will help ensure that the meta-level requirements of our data are met.

Finally, we can use a small sample of data to closely monitor and debug any portion of the network that is behaving oddly. This is useful to determine how widespread a problem is in the network and will help us focus on individual steps in the process if we need to. We can use the tool Tensorflow to help us accomplish this and other forms of debugging and testing.

5.4 System Testing

System testing in a machine-learning object classification system is a comprehensive evaluation process to ensure the model's reliability, robustness, and effectiveness in real-world scenarios. This testing phase goes beyond individual components and examines the system as a whole. In the context of an object classification system, key metrics such as precision-recall and confusion matrices are integral to this evaluation. Precision-recall metrics provide insights into the model's accuracy and completeness, while confusion matrices offer a detailed breakdown of the model's predictions across different classes. Including all beneficial features in the testing process ensures a thorough assessment of the model's capability to handle diverse input variations. Moreover, confirming that all hyperparameters have been tuned optimally contributes to the model's stability and performance. Before deployment, system testing also involves validating the quality of the model to ensure it meets predefined standards. Crucially, the assessment extends to monitoring resource usage to confirm the absence of leaks, ensuring the model operates efficiently over time. In essence, system testing in this context serves as a crucial step in verifying the readiness and reliability of the machine learning object classification system before it is deployed for practical use.

Precision and recall in the context of a deep learning model can be considered as metrics for evaluating the model's performance on a system level, making them part of a system test. In a deep learning classification system, precision and recall provide insights into the model's ability to correctly identify and retrieve relevant instances within a dataset. Precision is the ratio of true positive predictions to the total number of instances predicted as positive by the model. It measures the accuracy of the model when it predicts a positive outcome. On the other hand, recall, or sensitivity, is the ratio of true positive predictions to the total number of actual positive instances in the dataset. It assesses the model's capability to identify all relevant instances. In the context of a system test, precision and recall help gauge the overall effectiveness of the deep learning model in correctly classifying instances, capturing both the accuracy and completeness of its predictions. System tests for deep learning models often involve evaluating these metrics on comprehensive datasets, covering a range of scenarios and input variations, to ensure the model's robustness and reliability in real-world applications. By examining precision and recall, system tests provide a holistic assessment of the model's performance, contributing to the understanding of its behavior beyond individual training or validation samples.

A Confusion Matrix in a deep learning model serves as a system test by providing a comprehensive and quantitative evaluation of the model's performance across different classes. It goes beyond individual predictions, offering a holistic view of how well the model categorizes instances into various classes and identifies potential areas of improvement. The matrix captures true positives,

true negatives, false positives, and false negatives, enabling a nuanced analysis of the model's strengths and weaknesses. By examining precision, recall, and accuracy metrics derived from the Confusion Matrix, one gains insights into the model's ability to correctly classify instances and its susceptibility to errors. This holistic assessment is crucial in validating the overall efficacy and reliability of the deep learning model, making the Confusion Matrix a valuable tool in the broader context of system testing for machine learning applications.

In the context of a deep learning model, considering all features in the system test becomes imperative for a comprehensive evaluation of its performance. Including all available features allows for an in-depth examination of the model's ability to leverage the entire spectrum of information within the input data. This approach ensures that the model's predictive capabilities are tested across diverse dimensions and that it can generalize well to different aspects of the problem domain. It provides a holistic assessment of the model's robustness, uncovering its capacity to discern relevant patterns and relationships from the multitude of input features. By incorporating all features in the system test, one can validate the model's adaptability and effectiveness, enhancing confidence in its real-world applicability and ensuring that it leverages the full richness of the available data for optimal performance.

Tuning hyperparameters is a critical step in optimizing the performance of a machine-learning model. It involves adjusting configuration settings that are not learned from the data but impact the model's learning process. When all hyperparameters have been tuned, it signifies that the model has been systematically fine-tuned to achieve the best possible performance based on the chosen criteria, often improving accuracy, generalization, or convergence. This process involves experimentation with different parameter values, guided by techniques such as grid search or random search, to find the optimal combination. A well-tuned model will more likely generalize effectively to new, unseen data and perform optimally in real-world scenarios. Therefore, stating that all hyperparameters have been tuned reflects a thorough optimization effort to enhance the model's predictive power and efficiency.

In the context of a deep learning model, affirming "no resource leaking" in a system test underscores the model's efficient management of computational resources throughout its lifecycle. This assurance signifies that the model appropriately allocates and deallocates memory and other system resources, preventing issues such as memory leaks that could lead to performance degradation over time. Effective resource management is crucial in maintaining the stability and reliability of the deep learning model, ensuring that it operates efficiently and consistently, particularly when dealing with large datasets and complex neural network architectures. A system test that verifies the absence of resource leaks validates the model's overall robustness and its ability to handle computational resources responsibly, contributing to the model's sustainability and reliable performance in production environments.

5.5 Regression Testing

For regression testing, we need to make sure that the deep learning model continues to accomplish manually labeling objects within our recorded LiDAR data. As we develop our deep learning model, it will be important to designate a few datasets that can be used for object detection and, whenever we make changes to the model, ensure that the model still detects objects that it did in past versions. This is driven by the requirement to develop the best algorithm to detect objects accurately and quickly.

Regression testing in the context of deep learning models for object classification systems is a crucial quality assurance process. As these models evolve and undergo modifications, it becomes imperative to ensure that any updates or changes do not inadvertently introduce new errors or regressions. Given the complexity of deep learning architectures and the sensitivity of object classification tasks, regression testing becomes a pivotal strategy to maintain the model's reliability and accuracy over time. This testing approach involves systematically validating that the model continues to perform as expected, preventing the emergence of unintended consequences in its predictions.

Regression testing for a deep learning model in object classification systems is validated by ensuring that the model has not experienced a regression in prediction quality on served data. This means that after updates, modifications, or enhancements to the model, the predictions on real-world served data remain consistent and accurate. The absence of regression in prediction quality assures stakeholders that the changes made to the model have not adversely affected its ability to correctly classify objects, providing confidence in the model's continued reliability. This testing process involves comparing the current predictions with previously established benchmarks or ground truth data, enabling the identification of any deviations that may signal a regression in the model's performance. Such rigorous regression testing is essential for deploying and maintaining robust object classification systems, particularly in dynamic environments where data patterns and model requirements may evolve over time

5.6 Acceptance Testing

Our client has specified that the model should have an accuracy of at least 75% on the testing data we've collected. The model should also identify objects in less than 500 ms. The four classes that the model will identify are pedestrians, bicycles, cars, and buses.We will record and demonstrate our deep learning model's ability to detect objects regularly with our client, making time for it during our meetings. Once again, because of how important but uniform our unit tests are, most of our project's success will be determined by how well those perform on the scale that we need them to.

Many of our requirements either directly or indirectly reference the main requirement of 75% accuracy in identifying objects using our trained machine learning model. This success rate includes ensuring that the objects are identified fairly. If possible, we will split our testing into different data slices to test if our predictions change based on outside factors in grouped data.

Our functional requirements are all very dependent on the accuracy percentage being correct. Some of these requirements include the "Development of an object classification training model for LiDAR sensors," "Creation of a labeled dataset suitable for object detection," and "Implementation of an object classification system." Our main performance requirement is also in this category and will be completed if the percentages are met in our system tests.

Our last functional requirement, which involves ensuring our model works with various LiDAR sensors, is accomplished if we achieve the Integration Test in Tensorflow to ensure that the incoming data fits the requirements of having the correct fields. We would need to translate certain fields if necessary. Our first resource requirement also falls under this test, which states we must encompass access to multiple LiDAR sensors.

5.7 Results

Our testing process, inspired by the ML Test Score rubric and adapted to the unique challenges of object classification deep learning models, has yielded comprehensive results ensuring compliance with requirements. The unit testing phase focused on the probabilistic and data-driven nature of machine learning models, employing feature expectation schemas to establish baseline criteria for model performance. By systematically creating and adjusting these schemas, we ensured that our model's predictions aligned with expected outcomes, akin to traditional unit testing in deterministic software. The interface testing phase, while limited due to the absence of a sophisticated user interface, prioritized the effective transfer of data between the neural network stages. Integration testing honed in on critical paths, particularly the quality of training data and the training process. Manual tests, facilitated by tools like Matlab and Tensorflow, confirmed the smooth flow of data through each processing stage, addressing challenges related to overfitting, underfitting, and data quality.

System testing embraced precision, recall, and confusion matrices as key metrics, providing a holistic evaluation of the model's performance on diverse datasets. All features were systematically incorporated, ensuring a thorough assessment of the model's adaptability and generalization across varied input variations. The confirmation of tuned hyperparameters and the absence of resource leaks affirmed the model's stability and efficient resource management. Regression testing was pivotal in ensuring that model updates did not introduce regressions in prediction quality on served data. The absence of such regressions validated the model's reliability over time, aligning with the goals outlined in the ML Test Score rubric.

In summary, our testing strategy, influenced by the ML Test Score rubric, successfully navigated the intricacies of object classification deep learning models. From unit testing to system testing, each phase addressed specific challenges and requirements, ensuring that our design aligns with intended specifications. Precision, recall, and other system-level metrics provided a comprehensive view of the model's effectiveness, while regression testing guaranteed its continued reliability. By systematically applying these testing methodologies, we can confidently conclude that our object classification deep learning model is as intended, meeting the specified requirements and demonstrating robust performance across various scenarios.

6 Implementation

As mentioned in section 4.7.2, we have a robust plan to implement and continue to iterate on in the future as we work on the project. Our first step will be to continue to collect more data sets. We have a good base at the moment but will need more to fully implement and train our modified OpenPCDet model. We want to ensure that we continue to collect data sets in varying locations with enough pedestrians and vehicles for our model to be trained on and verified.

We will then sync the data we take with our Livox Mid40 LiDAR and webcam and begin labeling it. We are planning on using both MATLAB and Optimized Point Pillar to label our datasets to provide the labels we can use to validate our model later on. Our next step is to iterate through the model modification process using OpenPDCDet as a base and tweaking it as we discover what changes need to be made to improve the generic model for our use cases.

Finally, we will validate the model multiple times using our manually labeled ground truth values to ensure that the modifications we made to the model are accurate. We want to make sure our model is in the desired accuracy of at least 75% within the given 500ms scan.

As we move into the future and away from strictly designing and prototyping to implementing our project, we plan on beginning with the above design and modifying it as we discover better ways to achieve our desired final deliverable.

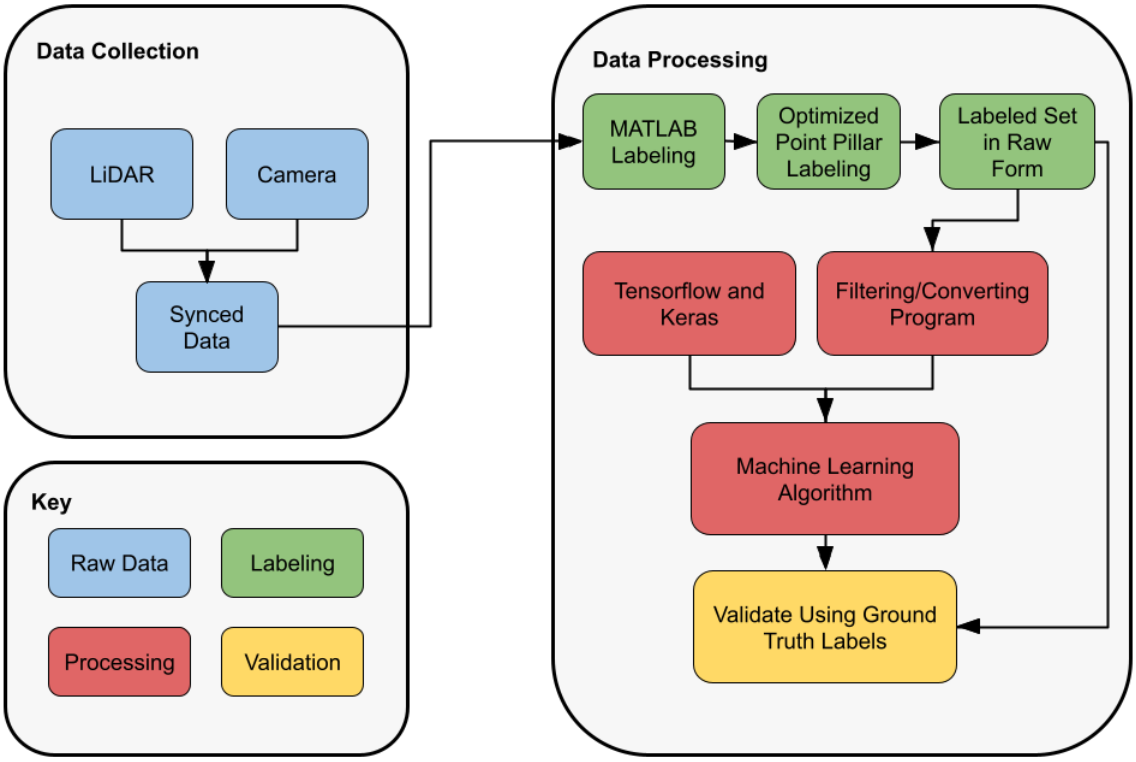


Figure 6: First design iteration of the system after the initial. Data collection includes LiDAR and Camera data being synced. That is then fed into our Data Processing pipeline of MATLAB labeling being put into Point Pillar to create a raw labeled data set. That also will be used for validation as ground truth labels. There will also be Tensorflow and Keras impacting the Machine Learning Algorithm that will be validated using the ground truth labels.

7 Professionalism

7.1 Areas of Responsibility

Area Of Responsibility	Definition	NSPE Canon	SE Code of Ethics
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	Ensure that software is developed and maintained with high quality, meeting professional standards and avoiding deceptive practices.
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	Strive to deliver software solutions that provide value to stakeholders and maintain financial responsibility in project costs.
Communication Honesty	Reports work truthfully, without deception, and are understandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	Communicate software-related information truthfully and objectively, ensuring transparency and avoiding deceptive communication.
Health, Safety, and Well-Being	Minimize risks to the safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	Prioritize the safety, health, and well-being of users and stakeholders in software development and use.
Property Ownership	Respect the property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	Acknowledge and respect intellectual property rights and confidentiality in software development.
Sustainability	Protect the environment and natural resources locally and globally.		Develop software solutions with consideration for environmental impact and long-term sustainability.

Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully.	Develop software that contributes positively to society, following ethical practices and legal obligations.

Table 3: NSPE Table with Software Engineering code of ethics applied to each area of responsibility.

For each area of responsibility, the SE Code of Ethics is integrated to emphasize how it complements or extends the NSPE Canon. The sustainability aspect is included without a direct NSPE Canon counterpart since it's more aligned with software development practices.

7.2 Project Specific Professional Responsibility Areas

- Work Competence
    - Applicability: Highly applicable. Developing a reliable object classification model requires a high level of professional competence in data collection, machine learning, and model development.
    - Team Performance: Medium. The team is likely to face challenges in understanding and implementing deep learning models, but with a dedicated approach, competence can be achieved.
- Financial Responsibility
    - Applicability: Highly applicable. The project involves using resources, including the LiDAR system and computational resources.
    - Team Performance: High. The team has access to a LiDAR system and is conscious of potential resource needs, as indicated by considering purchasing an additional LiDAR.
- Communication Honesty
    - Applicability: Highly applicable. Clear and transparent communication is essential for presenting project results, especially when dealing with diverse stakeholders.
    - Team Performance: High. Using Trello and GitLab indicates a commitment to effective communication and documentation.
- Health, Safety, Well-Being
    - Applicability: Applicable. While not a primary concern, maintaining safety during data collection and adhering to ethical standards is important.
    - Team Performance: Medium. The team must ensure safety during LiDAR scans and be aware of potential risks associated with data collection.
- Property Ownership

    - Applicability: Applicable. Respecting intellectual property rights and confidentiality is crucial, especially when dealing with diverse datasets.
    - Team Performance: High. The consideration of restrictions on collecting sensitive information aligns with property ownership responsibilities.
- Sustainability
    - Applicability: Partially applicable. While not explicitly mentioned in the context of software development, considering environmental impact and long-term sustainability is relevant.
    - Team Performance: Low. The project may not explicitly address sustainability concerns in the current context.
- Social Responsibility
    - Applicability: Highly applicable. The project aims to benefit various groups, including researchers, security enthusiasts, engineers, and the automobile industry.
    - Team Performance: High. The diverse intended users and uses indicate a commitment to societal benefit, aligning with social responsibility.

7.3 Most Applicable Professional Responsibility Area

The most applicable professional responsibility area for the LiDAR sensor project is "Work Competence." Given the intricate nature of the project, involving the development of an object classification training model for LiDAR sensors and the creation of a labeled dataset suitable for object detection, a high level of professional competence is essential. The team needs to navigate challenges associated with LiDAR sensor development, address data inconsistencies, and ensure the cross-compatibility of the object classification system. This responsibility encompasses the need for expertise in machine learning, neural networks, and data processing. As the team delves into unfamiliar territory, particularly in the realm of deep learning models, the application of professional competence becomes paramount for the successful execution of the project. The team's ability to understand, implement, and continuously refine complex technologies will significantly influence the project's outcomes, making "Work Competence" the most pivotal professional responsibility area for ensuring the project's success.

8 Closing Material

8.1 Discussion

The main task we've done this semester is collecting LiDAR data. Although we still haven't reached the amount of data we are required to collect, the amount we have collected contains a good variety of classes the machine learning model will be tasked to identify (cars, buses, pedestrians, and bicycles) and will work well as training and testing data for our eventual model.

8.2 Conclusion

The team has engaged in exploratory efforts with various LiDAR software tools to facilitate data collection and visualization. The Livox viewer, tailored for our Livox Mid-40 LiDAR, has been employed for data collection and visualization. Additionally, we have conducted experiments using Cloud Compare, a widely recognized LiDAR data visualization tool. In pursuing effective machine learning approaches, we delved into methodologies such as YOLO and Tensorflow.

We collected over 6 GBs of LiDAR data throughout this semester, and we are starting to manually label the classes of objects we wish the model to identify with the MATLAB LiDAR labeler. It has been harder to collect more LiDAR data towards the end of the semester because of the cold. If we were to do this differently, we would have started collecting data much earlier in the semester since more people and bicyclists are around campus.

Our overarching objective is to develop a proficient machine learning model to accurately classify objects within LiDAR point cloud data. To achieve this, our strategic approach involves creating a robust dataset featuring numerous instances of the targeted object classes, including cars, buses, pedestrians, and bicycles. We have yet to start implementing the machine learning model. However, we have researched different methods to help us with our application and will start developing them early next semester. Another constraint our team ran into was that our lack of knowledge of machine learning caused us to spend significant time learning the basics.

8.3 References

[1] "Livox Mid-360 User Manual v1.2," Jun. 2023.

[2] J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "BirdNet: A 3D Object Detection Framework from LiDAR Information," in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018.

[3] Y. Wu, Y. Wang, S. Zhang, and H. Ogai, "Deep 3D object detection networks using LiDAR data: A review," IEEE Sens. J., vol. 21, no. 2, pp. 1152–1171, 2021.

[4] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML test score: A rubric for ML production readiness and technical debt reduction," in 2017 IEEE International Conference on Big Data (Big Data), 2017.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," arXiv [cs.CV], 2016.

8.4 Team Contract

Team Members:

1) Ella Rekow
2) Sachin Patel
3) Zachary Schmalz
4) Anuraag Pujari
5) Ryan Sand
6) Daniel Rosenhamer

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:

Our team will conduct regular face-to-face meetings every Monday at 3:30 for discussions and updates. Additionally, we scheduled our T.A. meetings at 4:30 on Zoom every Monday to facilitate further communication and collaboration. These structured meetings are crucial in keeping our team organized, informed, and aligned on our goals and tasks.

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

We will primarily use text messages for quick and efficient communication for scheduling and small updates. However, for more substantial and critical conversations that require archiving and easy reference, we will utilize Discord to ensure that important information is readily accessible and well-documented. These communication strategies cater to the specific nature of the information being conveyed and enhance our team's efficiency and clarity.

3. Decision-making policy (e.g., consensus, majority vote):

Our decision-making policy is based on a majority vote. In the event of a tie, we will resolve it through discussions with relevant stakeholders, such as our advisor, TA, client, or the appropriate party involved. This approach ensures that decisions are made collectively when possible and that any deadlocks are effectively addressed through consultation with those who can provide guidance or additional perspective.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Meeting minutes and important documents will be shared and archived using Google Drive for easy access and collaboration. For formal communication and discussions, we will rely on Discord, ensuring that all important conversations are well-documented and accessible to team members. Code will be managed and version-controlled through Gitlab, allowing us to track changes, collaborate on development, and maintain an organized codebase. These tools and platforms are chosen to streamline our record-keeping processes and enhance our overall team efficiency.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

Individuals are expected to attend and participate in all team meetings and be punctual and engaged. If running late or unable to attend, providing at least an hour's notice is required to ensure smooth communication and collaboration.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Team members are expected to communicate openly, understand their tasks, and take ownership of assignments. They should plan effectively, meet deadlines, and collaborate with teammates. The team leader will provide guidance and support in meeting expectations and organizing the team.

3. Expected level of communication with other team members:

We will maintain open, transparent communication with team members and provide regular updates when timely responses are required. Team members will share relevant information, seek clarification when needed, and offer constructive feedback to other members. We will collaborate on tasks and address challenges as a group.

4. Expected level of commitment to team decisions and tasks:

The expected level of commitment to team decisions and tasks includes being dedicated to the project, proactively engaging with other team members, and using a collaborative approach. Team members are encouraged to actively contribute to discussions, provide input, and express concerns during decision-making. Once decisions are made, team members are expected to commit fully to the agreed-upon courses of action, demonstrating accountability, timely execution, and a willingness to support fellow team members. This commitment fosters a sense of shared ownership, leading to a successful project and better morale in the team.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

- Ella is responsible for team organization and/or management, client interaction, and acting as keeper of Documentation/Secretary
- Dan will be responsible for managing the communication mediums, LiDAR subject matter expert
- Anuraag will be the acting Data Manager and architect
- Sachin will be the Deep Learning Model subject matter expert
- Ryan will be the Data Collection Lead and will assist with organization
- Zach will be the regression testing lead as well as QA manager

Note these are the initial assignments and may be modified throughout the project's progression to better fit the needs of our client, group, and solution.

2. Strategies for supporting and guiding the work of all team members:

Our strategies to support the team include guiding team members by setting clear expectations, offering regular feedback, recognizing strengths and allowing those with them to use them, fostering open communication to ensure no one is out of the loop, and facilitating collaborative problem-solving to make the most out of our team. We will identify which areas team members are

stronger or weaker in and will take the time to share our knowledge and expertise in different areas. However, we will only force people to work on a particular part of the project based on their prior experience.

3. Strategies for recognizing the contributions of all team members:

The team will recognize team members by ensuring that in our presentations and group discussions, we credit the team members responsible for making progress in our project in the correct proportions and areas. Our presentations will have a visible and accurate recognition slide or page that is as comprehensive as the team agrees it to be. We will encourage everyone to have a voice in the group and give everyone a chance to provide input into discussions and decisions.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

Zach: I have hands-on experience working in agile teams on various projects during my academic courses and internships. I've used tools like Git, GitLab, and Jira to manage projects. My internship involved web and mobile development, where I focused on visual enhancements and bug fixes. I conducted thorough regression testing and documented issues to maintain project quality. I became proficient in handling large codebases, deploying websites with Docker, and working with frontend technologies like HTML, CSS, Swift (IOS), and Kotlin (Android). I've worked with Python, Angular, SQL, and AWS Elastic Beanstalk on the backend. I learned C, C++, Springboot, Hibernate, and Websockets in my academic courses, enhancing my technical skills. Throughout these experiences, I've developed strong teamwork and communication skills, consistently contributing to successful project outcomes.

Dan: I led the programming of my robotics team in high school, which taught me how to interface motor controllers, interact with servos, and communicate over WiFi. Through several internships and college, I've learned how to manage projects with Git, GitLab, Jira, and Rally. In these internships, I developed the front of programs with XML, React, HTML, and CSS and the backend of programs with Java, Node.js, Firebase, AWS (Terraform, S3, and a few other services within AWS), and various Java backend libraries (for APIs and WebSockets). I've used various platforms to develop projects, including Unity, Android Studio, and Visual Studio services. Additionally, I've implemented embedded technologies, including UART, Lidar, Sonar, PWMs, ADC, and CAN.

Sachin: I've gained many skills with the projects I've created throughout high school and college. I've learned programming languages such as Java, C, C++, C#, Python, and Javascript. I've developed with frameworks and libraries such as React.js, Node.js, Django, and Spring Boot. I've had three internships where I've learned project management and agile methodologies. I was also the software and electrical lead of the Cardinal Space Mining Club, where I learned to use 2D LiDARs for obstacle detection and plotting.

Ella: I bring a versatile skill set encompassing multiple programming languages, web development technologies, and automation tools. My commitment to quality and accuracy ensures that projects are completed to high standards. I excel in team collaboration, effectively communicating with internal and external stakeholders. I am pursuing a Bachelor of Science in Software Engineering at Iowa State University and have practical experience as a Front-End Developer at Netsmart and as a Junior System Admin and ServiceNow Admin at Iowa State University. In addition to my

professional roles, I actively contribute to the community, offering code instruction, developing open-source projects like a Discord bot, designing websites, and engaging in various learning experiences, including data analytics, reactive web design, and machine learning. My well-rounded skill set, dedication to self-improvement, and commitment to community engagement make me a valuable asset to any software engineering team.

Anuraag: Throughout my course load at Iowa State, I have taken many classes that have led me to gain certain skills. These skills include programming in Java, Python, and C. On top of that, I was part of an undergraduate research project that had to do with neural networks, which analyzed certain attacks on LSTM. Also, I am a data science minor and worked with Amazon S3 through an internship last summer. Back in high school, I was the business lead but also worked a bit with build and programming, so I do have some experience.

Ryan: My skill set is likely not the most technical of the group; however, I have good organizational and documentation skills. I have experience in Java, C, JavaScript, Python, and Microsoft Azure. I have worked in multiple Agile environments and have experience working in various team settings. I am working as the head of the Robotics Club at Iowa State University's Snowplow Team, which uses a 2D SICK LiDAR to plan a path through obstacles during our competition. I have a working knowledge of Linux, Windows, Git systems, and metric tracking systems. My experience in Robotics, both in high school and at Iowa State University, has given me ample opportunities to practice my soft skills, which help me present and engage with clients or team members well. As a detail-oriented individual, I will bring a practical perspective to the team and help us pay attention to parts of our project.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

We will always ensure that all team members can speak their minds in group discussions and that their contributions are not brushed aside in the moment or later. Everyone will have an important role in the group, which will help them feel ownership of the project so that the group as a whole is the driving force for our project. Our strategies include leadership leading by example, having a clear set of roles for each member to avoid clashes, and having a simple but strong voting system that we adhere to.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment obstructs their opportunity or ability to contribute?)

To address collaboration or inclusion issues, we will establish clear procedures for team members to communicate concerns, such as in our group chat or Discord server. We will encourage open dialogue and offer multiple channels for feedback. When a team member indicates that the team environment hinders their ability to contribute, we will promptly acknowledge their concerns and investigate the issues. This investigation could involve our TA or the professor if it is a behavioral issue or the client in the case of a decision issue. We want to ensure everyone is treated fairly and has the best chance to influence the project positively.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

Improve LiDAR Point of View (POV) Awareness:

Objective: Enhance our understanding of LiDAR data by improving the point of view (POV) awareness.

Key Result: Develop a more comprehensive approach to interpreting LiDAR data, accounting for variations in POV.

Action Steps:

a. Conduct a comprehensive analysis of current LiDAR data interpretation methods.

b. Identify common issues and challenges related to LiDAR point of view discrepancies.

c. Develop and implement strategies to enhance our awareness of LiDAR POV variations.

d. Continuously test and refine our methods to ensure improved accuracy and reliability.

Address the Lack of Standardization:

Objective: Mitigate the issue of standardization while working with diverse LiDAR systems.

Key Result: Establish an optimized approach that functions effectively across various LiDAR devices.

Action Steps:

a. Assess the existing disparities in LiDAR data output among different devices.

b. Identify commonalities and patterns that can lead to an optimized solution.

c. Collaborate with industry experts and colleagues to gather insights and best practices.

d. Develop and implement a flexible solution that adapts to various LiDAR systems without standardization.

e. Continuously monitor and update the solution to ensure compatibility with numerous LiDAR devices.

Emphasize Flexibility Over Standardization:

Objective: Shift our focus from standardization to flexibility in handling LiDAR data.

Key Result: Develop an adaptable solution that accommodates the inherent diversity of LiDAR systems.

Action Steps:

a. Educate the team on the advantages of flexibility over rigid standardization.

b. Encourage a mindset shift towards adaptable problem-solving approaches.

c. Establish clear guidelines for flexibly handling LiDAR data.

d. Foster a culture of continuous learning and adaptation within the team.

e. Share success stories and lessons learned to reinforce the benefits of flexibility in LiDAR data interpretation.

By pursuing these goals, we aim to enhance our LiDAR data interpretation capabilities, address the issue of standardization, and ultimately provide more reliable solutions that can accommodate a variety of LiDAR devices while retaining flexibility in our approach.

2. Strategies for planning and assigning individual and teamwork:

Strategies for effective planning and task assignment involve clear goal setting, skill assessment, balanced workload distribution, regular communication, and fostering accountability. During our weekly meetings, we will discuss and update the team on how each member's tasks are progressing. Once we have data from these meetings, we can adjust and rearrange tasks based on how quickly and smoothly they are completed. We will remain flexible and allow the majority vote to decide disagreements about who is responsible for which task. Keeping a clear and regular communication channel for this strategy to work. If we succeed in communicating in this way, we will be able to ensure that the tasks are shuffled to the best group of team members for the job.

3. Strategies for keeping on task:

Some strategies for task focus include prioritization, time blocking, clear goal setting, and accountability partners. We want to help the team be the most productive it can be while giving each member plenty of freedom. We will prioritize each task and communicate that clearly to the entire group so that everyone knows what they should work on first. The team will also create understandable and realistic goals to help keep the productivity on track. Finally, there will be more than one person working on each subsection of our project to provide accountability for each other if someone's productivity drops off.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

We want to keep our infraction handling simple but effective. Suppose a team member brings an issue to the team's attention. In that case, the first response is an informal acknowledgment of a person violating obligation (aka messaging/letting them know in some meetings that they aren't meeting expectations).

2. What will your team do if the infractions continue?

If there are continued infractions (after two informal reminders), a more formal confrontation could involve our advisors or the TA. We do not want to drag out any negative feelings and want to take care of any issues so that the team can go back to functioning productively.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) Ella Rekow	DATE 8/31/2023
2) Sachin Patel	DATE 8/31/2023
3) Zachary Schmalz	DATE 8/31/2023
4) Anuraag Pujari	DATE 8/31/2023
5) Ryan Sand	DATE 9/05/2023
6) Daniel Rosenhamer	DATE 8/31/2023

Appendix

Deep Learning Model: A deep learning model is an artificial neural network with multiple layers (deep architecture) that enables automatic learning of hierarchical representations from data, facilitating the extraction of complex features and patterns.

Git: Git is a distributed version control system that enables collaborative software development by tracking changes to source code during the development process. It allows multiple developers to work on a project simultaneously, maintaining a history of changes and facilitating collaboration through features such as branching and merging.

GitLab: A platform utilized for issue tracking and version control in the project's development process.

Keras: Keras is an open-source high-level neural network API written in Python. It serves as an interface for building, training, and deploying artificial neural networks, simplifying the process of developing deep learning models. Keras is often used in conjunction with other deep learning libraries, such as TensorFlow or Theano, and provides a user-friendly and modular approach to constructing neural networks.

LiDAR: a detection system that works on the principle of radar, but uses light from a laser.

Livox Mid-40: A LiDAR sensor developed by Livox, a company specializing in LiDAR technology. The Livox Mid-40 LiDAR sensor is known for its compact design, high-performance capabilities, and cost-effectiveness. It is commonly used in various applications, including robotics, autonomous vehicles, and industrial automation, where precise and real-time 3D mapping is required.

Livox Viewer: Livox Viewer is a computer software designed for Livox LiDAR sensors and Livox Hub. Users can check real-time point cloud data of all the Livox LiDAR sensors connected to a computer and can easily view, record, and save the cloud data for offline or further use

Machine Learning: Machine learning is a subset of artificial intelligence that involves the development of algorithms and statistical models that enable computer systems to improve their performance on a specific task over time without being explicitly programmed. It relies on analyzing patterns and data to make predictions, decisions, or identify trends, allowing machines to learn from experience and adapt to new information. Machine learning encompasses various techniques, including supervised learning, unsupervised learning, and reinforcement learning, and it finds applications in areas such as image and speech recognition, natural language processing, and predictive analytics.

MatLab: MATLAB, short for "Matrix Laboratory," is a high-level programming language and interactive environment primarily designed for numerical computing, data analysis, and visualization. It is widely used in academia, industry, and research for tasks such as mathematical modeling, simulation, and algorithm development.

OpenPyLivox: Python3 driver for Livox lidar sensors

Point Cloud: A set of data points in a 3D coordinate system—commonly known as the XYZ axes. Each point represents a single spatial measurement on the object's surface. Taken together, a point cloud represents the entire external surface of an object.

PyLas: A Python library used as a way of reading LAS/LAZ in Python

ROS (Robot Operating System): A set of software libraries and tools that help you build robot applications

TensorFlow: TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across various tasks but focuses on training and inference of deep neural networks.

Trello: A popular project management tool the team uses to organize and track their work.

YOLO: "You Only Look Once" is a machine learning algorithm developed for real-time object detection. It operates with a single forward pass through a neural network, simultaneously predicting bounding boxes and class probabilities within a grid system, making it highly efficient for various applications such as surveillance, autonomous vehicles, and robotics.

OpenPCDet: OpenPCDet is a clear, simple, self-contained open-source project for LiDAR-based 3D object detection used in Livox Detection V2.0.