SenseRator 2.0

University of Central Florida

Spring 2024 - Fall 2024

Sponsor: URBANity Lab

Group L10:

Nicholas Karamitos

Allen Lin

Abdullah Zahran AL Hinaey

Kristian Michel

Alexandra Ramlogan Salgado

Ayman Arif

Table of Contents

Executive Summary

As urban landscapes face increased scrutiny for their livability, sustainability, and safety, new and innovative approaches to urban planning and assessment have become paramount and have ushered in the era of the "Smart City." The SenseRator is our approach to pioneer a new initiative in urban assessment. By leveraging the capabilities of the Jetson Nano and equipped camera sensors, our goal is to redefine how urban planners evaluate the ever changing urban environments by using computer vision technologies to determine a Pedestrian Flow & Safety Score for the differing regions of a city.

Originally the SenseRator's capabilities were directed to disaster response, using object detection algorithms and primarily RGB camera sensors to detect factors such as flooding. The previous team developed a Python based Graphical User Interface in which you would upload a video taken from the SenseRator, and then it would run its object detection over top of the imported video and display it alongside the running video. This step of the project showcased the basic capabilities of the SenseRator which allows us to move forward. With instructor feedback and brainstorming sessions, we decided to shift our attention.

Our team aims to redefine the SenseRator's goals, transitioning to a user-centric system offering open-source data access, in hopes of transforming urban planning practices. With the SenseRator 2.0, we will equip a vehicle with a RGB camera, then using RGB-Camera based object detection, the SenseRator will drive a region of a city and detect then count selected features that make that region more walkable and safe. Once this is done, the data will be displayed on a web-accessed, 2D map interface that

allows the user to click over each region of a city and view the associated Pedestrian Flow & Safety Score, a score determined by features that indicate walkability and safety. Interested users can view video footage of the area or view the object detection footage used to generate the PFS Score of the area.

General Content

Introduction

Identification Of The Project

The SenseRator 2.0 project aims to integrate the use of RGB cameras mounted onto the Arcimoto FUV to create an interactive web application in which the user can view PFS Scores and video footage of a region. These regions will be used to assess urban environments by focusing on walkability and safety. This will be done by object detection to determine the PFS of each region within a city. This project aims to transform urban planning, by making it a comprehensive, real-time data accessible, user-centric web application.

The object detection component of the SenseRator will utilize RGB cameras to identify and categorize urban features such as sidewalks, traffic lights, and crosswalks which are relevant to the city's livability and safety. The deployment of computer vision algorithms are necessary to discern various different elements which are crucial for the evaluation of the walkability of an area. The effectiveness of the detection directly influences the accuracy of the data gathered, forming the foundation for the scoring process.

The PFS Score development is a crucial part of the project by combining all of the data collected into a comprehensive index that will measure the walkability and safety of each region in a city. Algorithms that will calculate the walkability of a city will need to be created to fit our needs since walkability is a very broad term. The score on the map will then be updated

accordingly to ensure that the index reflects the current conditions of that neighborhood. The PFS will allow users to quickly compare different areas of a city and help determine which path they will go. It also serves as a metric that will show urban planners, city officials, and citizens the regions that could use some more development and improvements.

The SenseRator 2.0 will mark a significant improvement for urban planning and development by offering a transformative approach on how cities can be evaluated and improved. Through the use of the RGB cameras, this project will create an interactive web application that is both innovative and practical. The integration of all of the features the SenseRator holds alongside urban planners, government officials, and citizens will provide the tools needed to enhance urban livability, safety, and sustainability. The SenseRator 2.0 will pave the way for smarter, more livable cities all around the world.

Significance

The SenseRator 2.0 can impact several key areas of urban development and planning. It improves urban planning and management by providing urban planners and city officials with accurate, real-time data about many different parameters such as walkability and safety. This information will help when making crucial decisions on where to allocate resources, implementation of policies, and planning future developments. The ability to assess urban areas comprehensively allows changes to be made based on data which leads to much more effective urban management.

Allowing users to see the PFS provides the community with tools to understand and evaluate their surroundings. The accessibility this provides

allows greater community involvement in urban planning processes and allows citizens to advocate for necessary changes and improvements based on data. This ensures that the voices of the residents are heard and supports community engagement.

This project also shows the innovation of technology through the use of RGB cameras for real time object detection and analysis software. The SenseRator pushes the boundaries of traditional urban assessment techniques. This approach improves the quality and granularity of the data as well as setting a precedence for future projects relating to smart cities.

The scale of the SenseRator project can easily be increased, whether it will assist in the analysis of post disaster areas or adapting to different environments. As the original SenseRator was focused on disaster relief, the capabilities of the SenseRator 2.0 could be adapted to specialize in post disaster relief as well. This ensures that the benefits of this project can scale up to regional, national, and global levels.

By tracking walkability and safety, the SenseRator also contributes to micro mobility and safety goals. It can help track risks to pedestrian safety and regions with low pedestrian mobility. This will lead to healthier and safer urban environments over time. The SenseRator 2.0 not only serves as a leap forward in smart city development, but also serves as a model for future innovations for urban development.

Motivations

As my education in Computer Science progressed throughout my degree, I learned smaller bits and pieces of different potential fields I can enter

post-graduation. One of these routes was machine learning and artificial intelligence. After some surface level research, I decided that it may be an interesting route to take, and took multiple courses regarding artificial intelligence and machine learning. This included: Algorithms of Machine Learning, Robot Vision, and Artificial Intelligence. After completing these courses, I decided that it was a field I was interested in pursuing, and enrolled into Senior Design with the expectation I would participate in a project related to artificial intelligence.

SenseRator ended up being one of the more appealing projects that were pitched. I heavily believe in providing a public service and finding ways to help people, so SenseRator's initial scope resonated with my personal values. I also felt that my previous semester provided me a foundation to begin, that SenseRator would give me the opportunity to build upon and have experience for future opportunities.

I believe that this project has taught me a fair number of things that can be useful skills for the future. I have learned how to maintain communication, handle curve balls, and express my ideas in a safe, healthy environment. Everything I have learned, and anticipate to learn, have only increased my excitement for the future of this project, and determination to complete everything the team has set our mind to.

- Alexandra Ramlogan

Since I started learning about Computer Science in high school, I have had an interest in machine learning. My first experience with formally learning machine language was at UCF. Studying algorithms for machine learning at UCF helped me learn a lot about machine learning ideas such as supervised

learning, unsupervised learning, and reinforcement learning. I learned about many different kinds of algorithms such as decision trees, linear regression, logical regression, KNN (K-nearest neighbor), clustering, and PCA (Principal Component Analysis).

Using PyTorch, I got hands-on experience with training data sets and using machine learning algorithms to train those models. The most interesting experience of machine learning for me was training data sets based on images. It allowed me to visualize a different way of training data sets. Unfortunately, I did not get much time to use images to train data sets. As soon as this project was pitched to the class, I was very interested since it would allow me to work on training data based on images/video footage instead of numbers on paper.

This project will allow me to deepen my understanding about machine learning and more about how it can be used in the real world. The backend for the SenseRator project is in C++, so it will be interesting to learn machine learning in a language other than Python. The project also encompasses machine vision, which is another bonus. Learning about how computers are able to analyze images/videos captured from sensors and translate them into usable data sounds very interesting to me. I want to learn more about how the data collected from each piece of hardware, such as different kinds of sensors, will be interpreted through machine vision. I would also like to play around with the hardware if possible.

The SenseRator can be a very beneficial tool to use to gather data after natural disasters. It will help first responders know where the most critical disaster management areas are so they can send help to the most critical areas quickly. It will also let them know about the scale of the damage from

the disaster. Working on this project will help me gain hands-on experience while also contributing to disaster management efforts.

- Allen Lin

When I first began looking into the project, I was interested in the aspect of the disaster relief potential SenseRator had, as the previous senior design group designed SenseRator to recognize potholes as well as flooded areas to alert individuals that lived around the area.

However, as I looked into the previous team's design I noticed that they didn't implement the LiDAR, to little or no regard which made me want to experiment with the LiDAR more. I also saw that they made a video playback feature which also made me interested in joining the project because I wanted to see how it worked, and I wanted to see if it can be improved for real-time processing.

The previous team also seemed to change their ideas a lot through the project such as moving from a C++ codebase to a python codebase in the middle of the run, as well as building the GUI from Unity engine, which made red flags pop inside my mind, I didn't like what Unity was doing with their software and I wasn't well versed in both of the languages but I did want to learn about them both so that was one of the reasons I chose the project.

I was also interested in adding my own touch to the project and building upon SenseRator which made me investigate the feasibility of my own ideas I wanted to add to the project, recognizing the homeless population as well as providing an incentive to clean up the environment, more on that later.

As it was getting closer to researching my ideas as well as learning new knowledge I've become more passionate about the project, because I like struggling to learn new things as it gives me motivation to continue because I have the ideology, I don't care you've beaten me once, I'm going to learn you, and I am most definitely going to solve you one way or another.

- Ayman

When I first began this project, I was a little nervous as I had never done anything related to machine learning before. What made it even more intimidating was the use of different advanced hardware tools such as the LiDAR that would be used in the project. However, at the same time I was really excited to be given the opportunity to learn about new tools and technologies within the field. I would be given the chance to work on a machine learning project that could impact a lot of people and give me the chance to learn a lot. Therefore my current motivation in this project is to learn as much as possible and use that knowledge to create a great ML model capable of evaluating the quality of life in Orlando's neighborhoods which city officials can use to further improve Orlando and perhaps one day other cities in America.

- Kristian

The Motivations that drive my enthusiasm for working on the Arcimoto SenseRator are drawn from my personal and academic experiences. Through my Computer Science degree, I've carefully attempted to search for my path of interest in all of the various opportunities and niche areas of the discipline, which led me into Machine Learning (ML), Artificial Intelligence

(AI), and Computer/Robot Vision (CV). This sub-discipline of Computer Science and programming in general, stand out to me as the area that will drive the future of technology. With that in mind and the knowledge I've gained from delving into these courses of study, putting my skills in ML, AI, and CV to the test and developing a real world tool, whose sole purpose is driven by a task defined simply as helping people in a time of need, is exactly where I want to be in my student career.

The primary objective of the Arcimoto SenseRator EV is to leverage the capabilities of LiDAR and camera sensors powered by CV in order to increase effectiveness in urban assessment and mapping for smart cities. By using object detection and methods of 2D and 3D mapping, It will assess the walkability, safety, and cleanliness of regions within a city. Having spent lots of time throughout my life in cities across the United States and Europe, I have always paid attention to what makes each place so unique and why certain places stand out more than the rest. It always comes down to those three factors as mentioned above. When cities are safe, clean, and walkable, they promote health, efficiency of day to day tasks, and a stronger sense of community. I believe that a strong initiative of urban planners moving forward in this country can help to create stronger and more efficient communities that promote health and sense of place as well as a clear design that is based around the human experience.

Where I believe I can contribute to this project most is in the Computer Vision tasks and also in the Machine Learning aspects of our work. I'm looking forward to learning new technologies and tools in order to help my teammates along the way.

- Nicholas Karamitos

I chose this project because I have always loved working on hardware projects. Unfortunately, when I started my degree in Computer Science at University of Central Florida , all my attention was focused on learning about software. This project presents an exciting opportunity to combine both disciplines – hardware and software – in a meaningful way. Another motivation is that this project uses Artificial Intelligence and machine learning to detect problems. At this time, most of the world is working on developing these technologies. By working on this project, I see myself gaining real experience and working on real projects.

From my point of view, this project has a lot of room to grow. There are many ways I can make it better. For example, I can make it do more tasks, like handle different situations. Additionally, there is room for improvement in the speed at which it detects problems. And I can make it more flexible, so it can work in different environments. I am really excited to see how I can contribute to make this project even better.

Another motivating factor to consider is that I'll be working on the second version of the project, which could potentially be easier since everything has already been set up before. This means I'll have more time to be creative and innovative with version two. Additionally, working on a second version provides us with a reliable source of information and guidance from the project sponsors, which can be incredibly valuable for me and my teammates. This opportunity allows us to build upon existing foundations and create something even better.

Another motivating factor for me is the potential impact this project could have on my career. I will be graduating within a year by gaining experience

and contributing to real-world projects like this one could significantly enhance my chances of landing a job at big tech companies. Many of these companies actively seek candidates with a decent understanding and experience in AI and ML, making this project an huge opportunity to develop and showcase my skills in these areas. By actively participating and making meaningful contributions, I am preparing myself for future career opportunities.

- Abdullah Zahran AL Hinaey

Broader Impacts of the SenseRator

Pedestrians

In terms of pedestrians the societal impact of the SenseRator stems from giving the ability to give a rating of a potential place of residence. Such that it would be able to give if the city of choice fulfills the needs of the individuals such as does it have enough benches and/or sidewalks for ease of transport. Would there be loud noisy streets across from their residence drilling into your eardrums day in and day out, or would it allow the ease of mind of relaxation.

Giving the pedestrians an ease of mind would allow them to develop a place of leisure. Another societal impact is that when society starts to think of the concerns of the individual then a harmonious trickling down effect can happen such as in the case that communal efforts can be prospered as when more QoL ventures are supported and financed the residents would be more likely to fulfill their daily needs as well as propagating engagement towards civic duties such as depositing and recycling to fulfill your part.

Cities

The SenseRator could potentially have a massive impact on cities, towns, and the like due to its unique goal and powerful features that leverage modern technology. City planners and politicians are eternally looking for ways to improve their city and thus increase the economy, revenue, cleanliness, and tourism of their city or town. All these factors can essentially be summarized in a simple phrase known as quality of life.

Quality of life can vary person to person in a city, but it's of an agreeable opinion that quality of life generally improves when technology improves. As a result, the SenseRator could potentially have a powerful impact on the future of Orlando's structure. The SenseRator is capable of taking images of neighborhoods and creating quality of life indicators for those neighborhoods. City officials can use this to be able to identify which areas in their city are in need of a quality of life upgrade. Thanks to this, the SenseRator will be able to impact many people and improve the quality of life in cities.

Legal, Ethical, and Privacy Issues

Navigating the complex world of laws and regulations is crucial for the success of our SenseRator project as we capture imagery across Orlando to assess neighborhood quality.

Protecting Personal Data

- Local Privacy Regulations: We need to ensure that in the city of Orlando we aren't going against any local privacy laws that prevent us from being able to film or will get us in trouble.

- Strict adherence to specific privacy laws in Florida: Likewise, we need to make sure we aren't breaking Florida laws related to privacy. Florida just recently enacted the Florida Digital Bill of Rights act to protect people's information.

- Significant resource allocation to deeply understand these legal frameworks - We will spend time reading up on modern laws related to the work the SenseRator does to ensure that no law is broken and that all protocols are safely and efficiently followed.

- Ensuring all personal data from images is handled with the utmost responsibility and ethics - We will ensure that none of this data will be personally leaked or used for any other purpose other than the purpose of the SenseRator which is to improve the urban life of Orlando.

Public vs. Private Space Distinctions

- Careful identification of public spaces where photography is permissible - We will make sure not to take images of places that are available for public view such as someone's backyard or inside of a commercial building.

- Detailed analysis and review of images taken in private spaces to avoid legal issues - We will personally and thoroughly review each image to make sure there’s no image taken of private places.

- Licensing Compliance - We will make sure we aren’t breaking any copyright or licensing compliances with our image data.

- Rigorous compliance with the YOLOv8 model’s licensing terms - We will make sure to strictly follow the YOLO model’s licensing terms so as to avoid any legal issues regarding our usage of the model.

- Regular audits of our practices to ensure they stay within the bounds of the law - We will make sure throughout the project we are perpetually following the bounds of the law.

Adhering to Surveillance Laws

- Transparent Communication with the Public - We will make sure to communicate with any related public individuals about the practice that we are conducting near them.

- Implementation of a transparent communication strategy to clear up any potential public misconceptions - We will make sure to have clear signage to make sure we will not have any public misconception about the SenseRator or the SenseRator car.

- Detailed public explanations of the objectives and methods behind our data collection efforts - We will make sure to have detailed

explanations of everything we are doing to ensure that anybody can access the secrets of the SenseRator.

Ethical Considerations

- Core Ethical Practices - We will make sure to continually follow ethical procedures while conducting model training for the SenseRator using our highly detailed and advanced data.

- Unwavering commitment to maintaining high ethical standards across all project facets - We will make sure to not violate any laws or morals throughout the engagement of this project and to make sure everything goes well.

- Consent and Public Interaction - We will make sure that we have appropriate consent about the images we've taken legally so that we are not in any danger or trouble.

- Proactive engagement with the community to secure informed consent - If such a situation arises wherein public engagement causes questioning, we will continue to have updated and modern information regarding our practices.

- Bias-Free Technological Application - We will make sure that the application of our technology is free from bias relating to creed, color, or other biases.

- Continuous refinement of the YOLO model to ensure it operates without bias - We will make sure our model operates without discrimination of any group of people.

- Regular updates and reviews of our AI protocols to increase decision-making fairness and accuracy - We will make sure that our AI does not have an unfair treatment of people or places.

Efforts to Minimize Bias

- Use of Diverse Data Sources - We will make sure to use different types of data sources to make sure we can catalog everything efficiently and morally.

- Employing a varied mix of data sources to train our model and reduce biases - We're attempting to reduce the amount of bias in our model to make sure certain areas aren't treated differently based on incorrect readings

- Commitment to a fair analysis that mirrors the true diversity of Orlando's neighborhoods v - We will collect data to reflect Orlando's neighborhoods to give accurate quality of life scores.

- Regular Algorithm Evaluations - We will make sure to continually evaluate our algorithms to ensure that we do not have any silly mistakes within our code.

- Routine audits of our algorithms to identify any biases - We will manually check the weights of our model to reduce the possibility of biases

- Immediate modifications to our algorithms to ensure fairness and maintain objectivity - We will make sure our algorithms don't accidentally produce discrimination among the diverse residents of the city of Orlando.

Privacy as a Foundational Element

- Purposeful Data Collection - We will make sure that the data we collect will be with intention, targeting specific streets and aspects that highlight the true city of Orlando.

- Precise data collection aimed specifically at assessing neighborhood qualities - We will make sure the data we collect will be specifically tailored to only assessing neighborhood qualities.

- Implementation of strict internal policies to clearly outline the purpose and limits of data collection - We will make sure to outline the limits of what we can discover.

- Utilization of state-of-the-art encryption and secure data storage methods - We will make sure our data is securely stored on the cloud, free from interference by hackers and the like who wish to steal data and harm people

- Adherence to rigorous data retention policies to manage data lifecycle and prevent unauthorized access - We will ensure that we will have policies in place to ensure that data is secure and safe for everyone.

- Respect and Care in Data Usage - We will make sure to respect the data that we retrieve and maintain its quality and security.

- Strict bans on using data for profiling based on demographic or personal traits - We will make sure that the data will not be used to distinguish or judge different types of people.

- Extra measures of caution when operating near sensitive areas such as schools and hospitals - We will make sure not to disrupt important public facilities during the process of our data collection.

Project Characterization

Initial Project Ideas

Ayman's Idea

The first idea I have begun researching was the feasibility to recognize homelessness and build some sort of model that would accurately depict them as well as creating an alert system that would have alerted NGOs or NPOs about high density population to build shelters around those areas and increasing more volunteer involvement in those specified areas. I was excited to take this project as it meant I could suggest this idea and could probably also get the greenlight to build the project based upon it. So I've made presentation slides showing how it is a niche topic as well as a crisis in Orlando and how it could be built and elaborated upon by future developments.

The H.A.S.™ would monitor dense populations of homelessness and from there would transmit it to local NPOs and NGOs by a secure network transmission and from there the NPOs and/or NGOs can decide where they can build homeless shelters as well as developing other accommodations for them. There was work being done on this however, there were drawbacks such as labeling data not being available during the project scope as well as there's an ethics problem at play in developing such a project which eventually led to its dissolution as the project progressed.

Furthermore, another idea that I had and pushed forward for consideration about the project scope is a litter detection system to help clean the environment as orlando is desperately in need of cleaning as just walking

across the campus you'll see more trash there than you saw moments before, and you might ask, okay why don't you pick it up and the answer to that is even if you do the wind blows it over and/or animals venture around stealing for their needs. Before you ask, covered trash cans won't solve the need of human carelessness as if it's one's habit to not follow rules then they'll surely more likely not want to if more regulation is imposed.

What I proposed is using YOLO's implementation of litter detection and creating an add-on to SenseRator being a simple claw that would do the work that others won't. While also adding an incentive for people to participate as there would be a reward program if you were to volunteer your time to gather the litter as all you had to do was take a free ride on an EV.

Finally, the last idea I proposed was monitoring and labeling endangered species and habitation. You might be asking how this can be done with LiDAR anyways as it doesn't provide color or intensity and/or depth of objects. That is correct and you pointed to a major point in terms of intensity and depth that can be used to see if habitat loss is dangerously high due to deforestation or low due to recovery efforts which would be gathered by adding drone capabilities to the SenseRator to monitor from a "Bird's-Eye-View".

However, there are issues with the design as most drones aren't capable to carry the weight of a sophisticated LiDAR system as well as there could be aerial zoning laws that are punishable by flying over restricted zones which even though it is beneficial there's too much bureaucratic red tape to overcome which eventually led to its closure.

The process of developing ideas can be fun as you can voice your own opinion as well as gain valuable criticisms to improve your idea. However, you must also be prepared to have your ideas to be abandoned as the more you delve into the feasibility of your ideas and justification the more likely you find out that they'll end up in the purgatory of possibility and reality. These may seem like moot points, however as projects progress challenges are always encountered and sometimes you just have to move with the flow for the betterment of the whole of the project as what will be shown sooner than expected.

Allen's Idea

My initial idea for the project was fully transferring the current capabilities of the SenseRator onto a drone in order to provide more accessibility to areas that the SenseRator vehicle cannot reach. This is because drones would be able to get a better overall view of a post disaster areas as well as accessing areas such as forests with its sensors.

The idea for this came from the currently existing Draganfly Sentinel-Ag and FlyGuys drone models. These drones have some capabilities of post natural disaster relief, but they are not specialized in it. The idea was to create a drone that fully focused on post natural disaster relief by scanning areas after a natural disaster and analyzing the data from those areas. When a pothole or a flood was detected, the software would notify emergency services to provide more relief and safety to that area.

The main issues with this idea are the lack of practical data and the weight constraints of a drone. There are not many datasets on post disaster areas and testing with the drone would be impractical as natural disasters do not

occur often. The carrying capacity of current surveying drones are approximately 2 lbs which is definitely not enough to be able to carry sensors and cameras. The constraints of this idea were too difficult to overcome and this idea ended up being scrapped.

Nicholas's Idea

Initially, when brainstorming how we could further continue the work of the SenseRator 1.0 team, whose primary objective was using the vehicle and its capabilities for post-disaster response, I tried to think in terms mainly of continuing this goal. The thought was "How can we increase the capabilities of the SenseRator to help people in disaster type scenarios such as Hurricanes, Flooding, etc?" Having been born and raised on the coast of Florida, this topic really piqued my interest because of experiencing and being affected by the various storms that haunt our state.

The first idea of mine was to equip the SenseRator 2.0 with not only Cameras and LiDAR, but also a thermal or infrared sensor in order to locate and notify city officials of potential coastal erosion caused by storms, or hidden and impending sinkholes. The SenseRator would be deployed periodically for regular surveys, and also used after significant storms, and would scan roads, beachside areas, and neighborhoods for cooler temperatures underneath the ground. Cooler temperatures would indicate the possibility of water under the ground and a lack of structured ground support, flagging that location for potential sinkholes or (in the case of being near the coast, ie. Flagler Beach, FL) erosion under the roads. Then using Machine Learning Algorithms for object detection and comparisons to previously recorded sinkholes and eroded coastlines, we could flag all potential finds on a map interface for city officials to further investigate. This

idea was plausible though we decided against it due to testing constraints in time and distance to coastal regions.

My second idea was to fall in line with UrbanITY Lab’s mission to further the research and development of smart cities. The SenseRator 2.0 could be reconfigured and retrained to detect and analyze traffic patterns and signs in real time in order to collect data for city officials to plan and re-assess areas of their city, which need updating to reduce bottlenecks in traffic. This idea was debated by the group and eventually we concluded that we could take some of these ideas of smart city research and create a more human centric approach to this project. We combined the ideas of traffic analysis and smart cities that some of each team member's ideas included in order to develop the current iteration and idea that we will accomplish, Urban Mapping and analysis for walkability, safety and cleanliness.

Abdullah’s Idea

I have several ideas that could significantly improve and advance the project. One idea I am particularly excited about is creating a product that can operate in many environments, including aerial environments using drones. This innovation could greatly increase the project's versatility and appeal to potential investors. By Using drone technology, we could expand the project's capabilities to monitor and detect issues in areas that are difficult for regular vehicles to access. For instance, deploying drones with sensors could allow us to gather data and identify problems remotely and efficiently.

Another idea that I believe would greatly benefit our project is to expose the AI model to the public by creating an application. This app would allow users

to flag areas with damages or issues, prompting our vehicle to run inspections in those locations. At the same time, this data could be used to further train and refine our AI model.

For instance, several big companies, including OpenAI, Meta (formerly Facebook), and Google, have adopted a similar strategy to train their data. They have deployed their models to the public, allowing for increased interactions with humans. This approach has proven effective in enhancing the models' abilities over time. By adopting this approach in our project, we could potentially improve the efficiency of our analysis. Additionally, the application could double as a communication platform, enabling us to receive feedback from users.

Alexandra's Idea

At the beginning of this project, I had been operating under the assumption that the team would be building on top of SenseRator 1.0's work done the previous year. My first idea was to extend the original service to the public, including a web application that gave alerts where the SenseRator made a detection. This would have been a map that marks locations where potentially dangerous detections could be, so users of the application know where to exercise caution after a natural disaster, and maintain public safety.

However, when our team was preparing to propose the initial plan we had, our sponsor redirected us towards a different area of potential, which ended up being smart cities. This sudden change in direction meant that we had to brainstorm our ideas once again. As I had a close friend studying architecture, I asked how the SenseRator could maintain its original goal while implementing the idea of smart cities. This is when she gave

suggestions on what green architecture was, which led to me looking more into quality of life infrastructure.

After settling on quality of life infrastructure, I then wanted to find a method that would use the technology available to our team, which is when I learned what Camera-LiDAR fusion was. My original idea was to use multi-modal object detection to find the available infrastructure that encourages a healthy quality of life in urban communities. However, I stepped down from that idea, when I realized how recent Camera-LiDAR fusion was, and that the team may not be equipped to complete the research needed to create this product.

Kristian's Idea

My initial ideas with this project had to do with air pollution. I had seen that the project could be a great resistor against the world's increasing amount of pollution. As the world becomes more populous and the people of the world continue to use technology such as cars and other equipment, more and more technology produces more and more pollution. As a result of this, cities have a responsibility to be sophisticated in their attempt to destroy pollution. So as a result, I suggested that the city can find the areas where this pollution is the highest. As a result, I can see that the car can be useful to cities to reduce pollution in an area. This is the reason the car can be useful for reducing the pollution in a city, and increasing the number of green areas in an area. Therefore the car could be very useful for reducing air pollution overall. This can apply to cities all over the world.

Objectives and Goals

- To integrate the use of cutting edge technology such as RGB cameras and the Jetson Nano with real time object detection on a vehicle to collect detailed data about urban environments.

- Develop a PFS that evaluates and scores different regions of a city by factors such as walkability and safety in order to provide a measurable and comparable assessment of urban quality.

- The creation of a user-friendly web interface that allows users to interact with a map to visualize the area as well as the corresponding PFS score.

- Enable more informed and effective urban planning and policy-making by providing accurate, detailed, and up-to-date data.

- Providing local residents and the community with more information about their environment to foster more participation in the urban development process.

- Promoting sustainable urban development practices by identifying areas of improvement in various urban features such as an area that commonly contains litter.

- Demonstrate the scalability of the SenseRator so it can be adapted and expanded upon in the future.

- Lead the path for more development on smart cities by showing technological capabilities and setting a benchmark for others.

Specifications and Requirements

Hardware Specifications

Vehicle

- Model: Arcimoto FUV or micro mobile vehicle
- Modifications: Mounts for RGB cameras and equipment
- Performance: Must be capable of navigating through urban environments safely and efficiently

Sensors

- RGB Cameras: High-definition cameras for capturing detailed visual data for object detection
- Stabilizers: Camera stabilizer to correct horizontal, vertical, and rotational movements to ensure the camera is taking stable footage

Software Specifications

Object Detection and Analysis

- Technology: Advanced computer vision algorithms capable of classifying and detecting urban features such as sidewalks, traffic lights, and crosswalks in real time

- Frameworks: PyTorch for developing and training machine learning models

Mapping and Visualization

- Web Development: Tools like Three.js for 3D visualizations and frameworks like React for building the web interface

- Mapbox: Displays a map and is sectioned off into regions to display corresponding PFS Scores and video footage

Data and Connectivity Requirements

Data Storage and Processing

- Capacity: High-capacity, secure data storage to handle large volumes of sensor data such as videos

- Processing: Powerful on-cloud computing capabilities to process data in real time

Connectivity

- Network: Reliable and fast connection to process data

Operational Requirements

User Interface

- Accessibility: Easy-to-use interface for all user groups including urban planners, city officials, and citizens

- Functionality: Features such as interactive maps, data overlays, viewable PFS scores, and video footage

User Story Requirements

- As a developer, I need to look into all of the hardware that is mounted onto the SenseRator such as the RGB cameras in order to understand how they work.

- As a developer, I need to learn how object detection works with algorithms such as YOLO in order to train the data collected by the SenseRator.

- As an object detection developer, I need to look for more datasets to train models on in order to increase the accuracy of the model.

- As an object detection developer, I need to learn how to optimize inference speeds to run object detection in real time.

Concept of Operations

The SenseRator 2.0 is designed to advance urban assessment by integrating cutting-edge technologies into a micro mobile vehicle, enhancing its capability to collect and process data for urban sustainability. The SenseRator 2.0 begins with the deployment phase where the vehicle is equipped with a RGB camera and is dispatched to urban areas. This deployment strategy will help gather different kinds of data for the walkability of each region.

Once deployed, the SenseRator collects data continuously using its onboard sensors. The RGB cameras capture detailed visual information to be used for training/analysis. This data is critical for identifying and classifying features related to urban walkability and safety. Camera stability is ensured to collect accurate and stable images.

Data collected by the SenseRator is transmitted to a central processing unit to use for analysis in real time. This setup allows for immediate data processing using advanced algorithms for object detection and classification. Simultaneously, data is securely stored in a database to support further processing and future access.

Following data collection and transmission, a PFS is then calculated for each mapped region, providing a quantifiable measure score of the area's walkability and safety based on the data collected. This data is then stored into the respective region within the map where the data was collected in our web application.

The final phase involves user interaction through a web interface, developed using technologies like Three.js for 3D rendering and MapBox for map visualization. This interface allows users to explore the data interactively, view different city regions, and understand the PFS scores. Users can also access detailed video footage of the regions with either a clear view or an object detection view.

Hurdles

One struggle we have had so far in this project is trying to find enough datasets for training. Trying to find datasets with similar segmentation techniques as well as the correct labels is a difficult task to achieve. Many of the datasets that we have found also do not contain that much data within them. They usually only contain a couple hundred images which is not much data to train with. We had to manually collect and label the data for three datasets to solve this issue.

Getting the hardware to boot up correctly was also a big challenge. The Jetson Nano hardware and the needed python version to support Ultralytics were contradicting and the jetpack version installed on the Jetson could not install the minimum python version required. This ended up taking us a week to solve by reflashing the Jetson Nano and creating a virtual environment to be able to install the necessary python version to run YOLO.

Optimizing the object detection inference speed was another encountered issue. Initially our detections were very accurate, but very slow at about 3-4 seconds per inference due to the large datasets and high accuracy. The

solution to this was decreasing the size of the datasets to decrease inference speed down to 500 ms as well as keeping fairly high accuracy.

Another struggle for this project was deciding on what the scope of the project would be. The first couple of weeks up to the first TA check-in were used on researching for the scope of the project. Many different ideas were thrown in, and they were pretty much all scrapped when the final scope was chosen. The weeks of research we put into those ideas didn't contribute too much and the team ended up behind schedule.

Another challenge was implementing video stabilization in the initial state however, using OpenCv's **videoStab** functionality resulted in an "overlaying bug" which kept placing the current frame it was on repeatedly which resulted in a jarring video in the playback with incomprehensible imagery.

Furthermore, In the initial attempt this was combated with changing how the overlay format was being displayed as it was a simple change from "`layer_func=vidstab.layer_overlay`" to "`layer_func=vidstab.layer_blend`" however even due that corrected the jarring repeated splatter it also resulted in a new bug which came in the form of blurring the video completely, as a final motion was removing the parameter altogether which resolved the issue.

However, another issue arrived which led to the video to jump around different placements in the playback however, the solution to this problem was shifting the parameter from "`border_size=0`" to "`border_size=100`" which improved the quality of the playback but was still jittery which finally resulted to just setting the parameter to "`border_size=auto`" which resulted in the best playback but a similar bug to the "overlaying bug" which

attempting to fix it led to either the processing time to increase dramatically or crash on frame transitions given the the nature of how Google Colab functions which ultimately led to the abandonment of this software resolution to the hardware solution discussed later involving a gimbal for stabilization due to time constraints of the project.

Another hurdle that was faced was bitrate conversion, VLC was initially chosen for its capability of real-time bitrate conversion based on the hardware capability of the computer system however the actual implementation was never achieved in applying it to the webapp as the necessary components were deprecated by 2020 as the fall of plugins arose in major browsers and attempts for a javascript(js)-base approach resulted in deprecated code which couldn't be assessed and revised in the timeframe for the project.

However, there was documentation for a WIP js by Jean-Bapiste Kempf which attempted to convert it strictly to webassembly which resulted in a proof of concept but hasn't been attainable in a usable state as of now which eventually led to this approach to be abandoned altogether. Ultimately the solution to the bitrate conversion came in the form of **MoviePy** which gave the freedom to choose your own respective bitrate for the video which allowed for seamless playback on major computer systems.

Another hurdle that was approached was in terms with the video compression algorithm as in the preliminary approaches resulted in either bloating the video size, crashing due to Google Colab's `runtime disconnected` bug or misinterpretation of the documentation as well as misuse of libraries with similar names but different setups. Which were resolved as discussed in the following text.

Furthermore, continuing on the previous hurdle that had to be resolved was frame compression which was initially attempted to be resolved by **Gifsicle** which only ran from the command line and after a while it became tedious to continue testing on this respective approach which ultimately led to the python-based one in the name of **pygifsicle** based on preliminary understanding on how gifs worked.

However, attempts with it led to the realization even though it removed the audio which resulted in a smaller file size it led to a consequence of removing the actual encoding that allows video compression to take place as it removed the H264 encoding and replaced it with large quality images in the form of gifs which essentially rendered the attempted resolution meaningless which led to this approach to be ultimately scrapped. However, **MoviePy** once again came to the rescue and resolved the frame compression problem as it had an built-in audio remover as well as a built-in codec conversion which made the process seamless.

Furthermore, another consequential hurdle that was faced during the video compression algorithm was validating and resolution checking the video playback which was initially done from the command line using **FFMPEG** but eventually similar to **Gifsicle** it eventually became too tedious to execute the command which resulted in using the python-based approach called **ffmpeg-python** however the python-based one was heavily limited in what I was attempting to execute as it lacked major components of **FFMPEG** which eventually resulted in finding the `forked` version called **python-ffmpeg** which had everything I wanted however there was a certain bug with the `.run()` which needed to always be synchronous however given Google Colab's `Runtime Disconnect` bug led to using the

asynchronous function version of `await .run()` which resolved the issue and allowed the compression algorithm to function seamlessly even given the `Runtime Disconnected` bug.

Design Documentation

Architecture

Urban Architecture

A major aspect of this project intersects with another line of research: Urban Architecture. To provide a quality product, it was important to know what is being researched, why it was being researched, and what it means. This begins with answering “what is urban design in architecture?”

One of the URBANity Labs main goals was to implement the concept of “smart cities” into urban areas, and research how technology can improve urban design. Urban design can be defined as focusing “on how the elements of the public realm can be built and arranged to create a positive experience for everyone.” (nyc.gov). In short, it focuses on making decisions that can improve the quality of life of a community, and is accessible to multiple communities.

According to MRSC.org, urban design must Address how people interact and perceive their environment, consider how “form-giving actions” affects ranges of individuals at a range of scales, aims to achieve “multiple objectives for multiple clients (including affected members of the public)”, and goes through an “ explicit decision-making process that offers the public the opportunity to participate in a meaningful way.” This means that there must be identified goals, acknowledge the existing conditions, consider all possible solutions, and develop a strategy to execute the best solution.

As Urban Design is a large, overarching branch of architecture design, there needs to be a clear goal, or smaller range of possibilities of what can be detected by SenseRator and what purpose it could serve.

Quality of Life Infrastructure

An aspect of urban design is ensuring there is infrastructure that improves the quality of life of the community in consideration. *Quality of life infrastructure* can simply be defined as architectural design that improves the mental, physical, and social well-being of its community, therefore improving the quality of life within urban areas. This includes mindful and respectful attention to how certain design elements can affect the aesthetics, technology, and materials used in construction, can affect a community, and how it can be designed to promote health, accessibility, and safety to the local community being considered.

Some examples of infrastructure that affect the quality of life in a community includes, but is not limited to: public transport, natural landscape, sidewalks, water supply, and safe road design. According to whatworkswellbeing.org, "Well-designed infrastructure can enhance the local and natural surroundings through good urban design, access to blue and green spaces, and reducing pollution and greenhouse gas emissions."

Walkability promotes physical health, mental health, and economic prosperity as safe, walkable urban design allows residents of local communities to navigate to other areas with ease, and provides the comfort that community members are safe from fast-moving vehicles. As such, road width, lower speed limit, a low amount of parking lots, and a vast amount of urban vegetation such as flowers, trees, and shrubs. These are the

beginning elements of walkable communities, and as such is a stepping stone to well-planned urban design.

Cleanliness can also highly contribute to the quality of life in urban areas. This includes accessible trash bins, reduced litter, clean pathways, and efficient storm drains that avoid stagnant water. Cleanliness is a valuable element of quality of life, as the sanitation of an area can contribute to the mental and social well-being of residents in an area. This is because streets filled with trash can lead to a sense of discomfort, insecurity, and unhappiness. A lack of cleanliness will then lead to an area being less walkable to a community, as residents will be unlikely to feel safe, energized, or be willing to be outside in an area full of litter and odoriferous trash.

Security must be discussed in regards to quality of life urban architecture, and whether local infrastructure improves quality of life. While this does not include police presence, this includes urban design that makes it safer for a community to be outside. Some of the main highlights of security are lower speed limits to encourage pedestrian safety, traffic control, labeled crosswalks to indicate pedestrian crossing and promote accessibility to those who are disabled, such as the blind, and streetlights to provide vision at night. These components of architectural design allows a community to feel safe being outside and using the architecture labeled above without feeling at risk of their wellbeing.

While there are other aspects of design that encourage the quality of life, such as access to public transport and consistent access to water, they exist outside the scope of SenseRator.

Software Components

Data Collection

One of the big challenges of using machine learning is finding enough data to train the model on. For this project, datasets on many unique features such as sidewalks, crosswalks, benches, street lights, trees, and stop signs must be found. These datasets must have a good sample size of images and the features should look similar to the same features that can be found in the location the object detection algorithm will be run on. Finding online datasets with a good sample size, as well as the features being similar looking to the objects found within a certain area is a big challenge. Finding datasets online that include the features needed are a challenge as well. There is a high chance that the datasets found online will either not include the features needed or the features are not accurate to the area that the object detection algorithm will be run on.

One solution to finding a better dataset to use for an area is to manually collect and label the features in that location. This will ensure that the object detection algorithm will remain accurate in detections within that area as well as the dataset containing a proper sample size of data. The downside to this though is that it requires a lot of time and effort to collect and label thousands of images for every feature. It also required a lot of memory space as thousands of images can take up to gigabytes of storage very quickly.

Data Interference

There are many types of interference that can get in the way of data collection. Weather conditions, physical obstructions, and electronic interference are all factors that can obstruct the data that is captured by a camera. For RGB cameras, weather conditions such as rain and fog can obscure the camera's lens which would not allow the camera to capture clear images. Wind is also a factor as strong winds can shake the camera causing blurry footage or cause a physical malfunction by making the camera fall. Extreme temperatures could also cause the camera to malfunction or shut down due to heat or cold.

There are many ways to work around these issues such as waiting for the weather conditions to be better, using a towel to wipe away moisture from the camera lens before capturing an image, and using current technology that is made to counteract these weather conditions.

Physical obstructions are another big issue with capturing data with a camera. For example, cameras would not be able to capture what is behind a tree or a wall which can be an issue. A clear view of a feature is a necessary requirement to keep an object detection algorithm accurate. If a feature is hidden behind a tree, then the algorithm would not be able to detect that feature. A potential solution to this is to position the camera in a different position to minimize the number of obstructions that would be encountered while collecting data. This would not solve all of the issues with physical obstructions though.

Collected Data

Some of the datasets we have found for RGB data have contained labels for crosswalks and red lights, as shown below in Figure 1, with semantic segmentation classifying crosswalks with red and traffic lights with purple. This dataset also contains green lights and sidewalks as labels. This is good as this dataset contains some of the labels that will be used for our Quality Index.

An issue with this dataset is that the dataset only contains those 4 labels and does not have benches, streetlights, and other such labels to classify objects that will be used in calculating the Quality Index for the region.

Figure 1: Crosswalk and Red-Light labels

Another issue with finding datasets is that certain objects, such as sidewalks, found in the dataset can be vastly different from the objects that you are trying to classify with manual data. For example, the sidewalks that can be found within the same dataset as Figure 1 are different from the sidewalks that are being classified for this project. The sidewalk found in Figure 2 labeled in teal is different from the sidewalks that we will be

capturing manually so the sidewalks in this dataset will not be helpful when training a model to detect sidewalks.

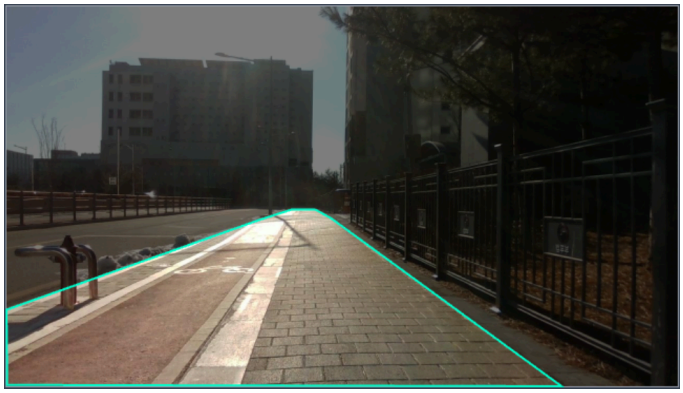

Figure 2: Sidewalk label

This issue may be fixed by including many different types of the same object within a dataset to spread out what the model can identify. For example, in Figure 3, an image from a dataset about trees, there are labels for specific tree species while also having a general label for all trees that do not have unique labels. In this dataset, Conifère trees are labeled in red, while all other trees are labeled in green.

Figure 3: Tree and Conifère labels

Classifying similar looking objects within a dataset with general labels can help the training model better identify different types of the same object as long as there is enough data within the dataset.

Potential Labels

For the SenseRator project, which aims to evaluate and score the quality of life and walkability in urban environments, particularly in the city of Orlando, developing a precise and relevant list of object detection labels is crucial. These labels will guide our analysis by focusing on specific elements that are indicative of urban quality and infrastructure. Here's a detailed breakdown of these categories:

Public Transportation - Identifies features like buses, subway entrances, taxi stands, and bike-sharing stations. Good, Amazingly High Quality, Efficient public transportation is a key indicator of a city's mobility and accessibility.

Parks and Green Spaces - Includes public parks, community gardens, playgrounds, and open recreational areas. Green Spaces are absolutely essential for any good smart home environment.

Pedestrian Crossings - Covers crosswalks, pedestrian signals, and safety signs. Areas safe from traffic are absolutely needed in a smart environment to make sure that safety is maintained for the people.

Residential Buildings - Detects various types of housing including apartments, single-family homes, and high-rises. Residential diversity can be a measure of community health and economic diversity alongside reducing bias in the dataset concerning buildings.

Commercial Buildings - Includes shops, offices, and business complexes. Super Active commercial activity is a sign of economic life and job availability.

Public Amenities - Combines a list of all the schools, hospitals, libraries, and community centers in an area. Availability and accessibility of public amenities are crucial for the well-being and prosperity of a community.

Trees and Vegetation - Identifies tree cover, landscaped areas, and undeveloped plots. Vegetation levels can affect the temperature and the climate of an area.

Pollution Sources - Pinpoints potential pollution sources such as industrial sites, waste treatment facilities, and high traffic roads.

Traffic Density - Measures the amount of vehicular traffic on the roads, which can show how congestion, and car pollution affects the quality of life in an area.

Road Hazards - Includes detections of potholes, road cracks, and other hazards that could affect driving conditions and vehicle safety.

By accurately detecting and analyzing these elements using the SenseRator's advanced object detection capabilities, we can provide a comprehensive assessment of the quality of life in different neighborhoods of Orlando. This information will be invaluable in helping city planners and policymakers make informed decisions to enhance urban living conditions.

The previous datasets found all have something to do with the above labels. Buildings are naturally a part of a car's view. More specific aspects of a car's journey such as sidewalks, trash, and crosswalks each have their own dataset to better identify key aspects of the road. Using these details, the model will be able to make an accurate estimate of an area's quality of life based on a predefined index. This predefined index will be discussed in later parts of this document. However, it is very important to consider the type of criteria needed to match the citizens' view of a particular area.

Segmentation

A big issue with finding datasets is the type of segmentation each dataset uses. The datasets are segmented by using instance, semantic, or panoptic segmentation.

Instance segmentation detects and separates objects without knowing the region surrounding those objects. This technique is mainly used when tracking individual objects. Semantic segmentation labels every pixel in an image with a class label with no other information considered. This technique is used when identifying different classes within an image is important. Panoptic segmentation is the combination of instance and semantic segmentation. It labels each pixel with a class label and identifies each object in the image. Panoptic segmentation is used for the detection and interaction of different objects in their environment.

The segmentation techniques cause an issue with compatibility between different datasets since the classification of labels for each segmentation technique is different. For example, if we classified different traffic signs to different scores such as, stop signs are 5 points and yield signs are 3 points,

then we would use panoptic segmentation datasets instead of semantic segmentation datasets. As seen in Figure 4, panoptic segmentation classifies every pixel and object within the image.

Figure 4: Panoptic Segmentation example

We will use either instance or panoptic segmentation for this project as we will need to detect objects to use for scoring. A general labeling method could cause the PFS to be inaccurate as specific traffic signs are more important than others. It could also affect the 3D map that will be generated by causing inaccuracies on specific traffic sign placement for example. The use of instance or panoptic segmentation would remove that issue and would bring a more accurate score to each region.

Data Augmentation

After collecting all of the necessary datasets containing the features that are needed, a good way to increase sample size as well as accuracy of an object detection algorithm is to apply augmentation. Augmentation changes an image in multiple ways such as rotation, flipping, resizing, brightness, color color changes, blur, and distortion. Included augmented images within a dataset grows the sample size as well as allowing the object detection

algorithm to detect features when they are rotated or blurred. Augmentation is a necessary step in making a good dataset that will help object detection algorithms increase their accuracy as well as detections in general.

One issue with augmentation is the fact that datasets may be over augmented causing the object detection algorithm to not recognize the original features as well as the augmented ones. The simple solution to this issue is to at most apply 1 augmentation to an image and keep the augmented image size to the same number as the regular images. This will ensure that the original image features will be detected with high accuracy as well as the features that are rotated or blurred.

Dataset Combination

After gathering all of the necessary datasets, they must be combined so they can be used to train an object detection model on. To combine datasets, first create a mapping for the original labels and use that to remap the labels to combine all of the images and labels in each dataset. Then split up the images into a 80% train and 20% validation set. Then move all of the images and labels within their respective set into the newly created combined dataset. Then in order to use this combined dataset for training, create a data.yaml file that instructs the training function on the labels and paths in our dataset. When the data.yaml file is created, this dataset can now be used to train an object detection model. This can be a very complicated process especially when dealing with many features with thousands of images and labels. A way to simplify this process is to combine one dataset at a time to ensure that the combining went as desired. All of the datasets we have collected and are ready for combination can be seen in figure 5 below.

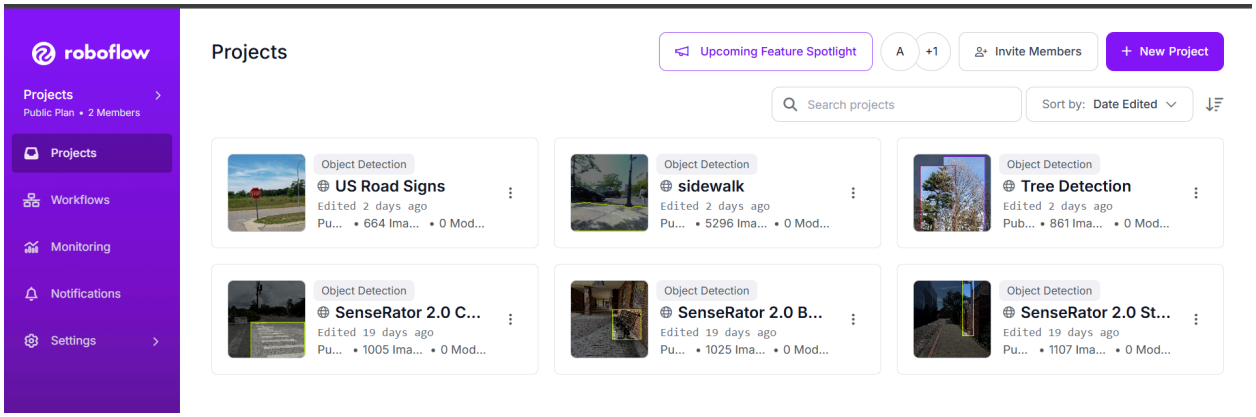

Figure 5: Collected Roboflow Datasets

Dataset Training

To train a dataset, there are 2 options. Either by training a model from scratch or by using a pretrained model. Retraining a pretrained model is the popular option as it has a baseline and it is easy to do. We retrained our model using a popular YOLO model that is pretrained on the COCO dataset which has various different sizes such as nano, small, medium, and large. The larger the model, the more accurate it gets, but it is slower and requires more processing power than the smaller models. For our purposes in real time object detection, we are using the YOLOv8n, which is the nano model which is very fast in running inference and does not require much processing power.

Object Detection

Object detection is a large component of this project used to scan objects for use in our Pedestrian Flow & Safety scoring index. There are many different algorithms that can handle object detection and the one we will use is CNN (convolutional neural networks).

There are 2 different ways to get started with object detection with machine learning. The first method is to create and train a custom object detector. The results that come from this method can be very extraordinary, but it requires a vast amount of time since you need to manually set up layers and weights in the algorithm. The second method is to use a pre-trained object detector. Many object detection algorithms use deep learning transfer learning, which allows you to start with a pretrained network that can be customized for your needs. This method is much quicker than the first, but it is also less customized for your specific needs.

There are 2 kinds of networks that will need to be chosen from to use, a single-staged network or a two-staged network. The single-staged networks produce network predictions for regions across the entire image using anchor boxes and the predictions are decoded to generate the final bounding boxes for the objects. Two-staged networks identify region proposals, or subsets of the image that might contain an object during the first stage. In the second stage, it classifies the objects within the region proposals.

Single-state networks are much faster than 2-stage networks, but the accuracy is also lower than that of two-stage networks. Two-stage networks are much better to use when dealing with smaller objects while single-stage networks are better to use with larger objects. For this project, we will be using single-stage networks since we will be dealing with large objects such as street signs, benches, sidewalks, and crosswalks.

Before the actual classification begins, the preprocessed images are analyzed to extract meaningful features. This feature extraction stage is crucial as it simplifies the subsequent classification process. Techniques such

as filtering, color normalization, and intensity adjustments are employed to highlight the key elements in each image that are pivotal for accurate object recognition.

Afterwards, the algorithm attempts to identify unique data that could make classification within an image much easier. An algorithm might attempt to look for edges or corners to discover the shape of an object. Afterwards, the model will attempt to classify an object while displaying a bounding box around it. With the right data, a machine learning model can become much better and faster at classifying images for the purpose of the programmer.

In the context of the SenseRator, this method of object detection is practical and easier to set up because there are many online models that are capable of detecting objects to the group's desire. The SenseRator's goal is to identify the walk-ability of different neighborhoods across Orlando and display them on a web application. The reason this method works so well in this context is that the SenseRator already has a high quality camera attached to it that makes it easy to take video and images of the streets of Orlando. Through this data collection method, we can use a model that's trained to identify the walk-ability of a street.

Real Time Object Detection

YOLO (You Only Look Once) is an extremely popular object detection model known for its impressive speed, making it a favorite among machine learning programmers. This speed is crucial in real-world applications where training models on large datasets can be extremely time-consuming—often taking days or even months.

While earlier versions of YOLO tended to sacrifice accuracy for speed, recent iterations have significantly bridged this gap. For instance, according to a study by Cornell professors, "YOLOv5 outperforms other algorithms in terms of accuracy with a mean Average Precision (mAP) of 0.593" (Naftali, Sulistyawan, Julian, Aug 24, 2022). Building on this foundation, YOLOv8 introduces further optimizations that enhance both efficiency and correctness, making it highly competitive with traditionally more accurate but slower algorithms.

We chose to use YOLO because it is a popular algorithm that is known for its accuracy and speed. It is also easy to work with which helps us train our model much quicker. With our trained model, we run inference on frames that are captured from our Runcam 2 which detects and displays objects on our monitor. Because we are doing real time object detection, we have focused a lot on optimizing inference speed and have gotten it down to 500 ms. As soon as our program ends, the objects counted are displayed at the end as well as our calculated PFS score for the run.

Object Detection Model Comparison

While previous phases of our project involved extensive research comparing various object detection models, we have continued to conduct manual tests to validate and verify these findings ourselves. Initial expectations were that our experimental results would closely align with the existing research data, and while this was largely the case, there were some minor discrepancies. However, these differences largely confirmed our initial hypotheses regarding the performance differences between the models.

Our tests underscored the exceptional speed of the YOLO (You Only Look Once) algorithm: it outperformed other models significantly, being approximately 5 times faster than the Faster R-CNN model and about 50 times faster than the SSD (Single Shot MultiBox Detector) model. For instance, over a test batch of 10 images, the SSD model required about 2 seconds for processing, the Faster R-CNN took around 0.125 seconds, while the YOLO model processed the images in nearly instantaneous time—demonstrating its prowess in speed.

Given the scope of our project—the SenseRator—which involves processing millions of images to derive meaningful insights and scores, efficiency in processing time is crucial. The YOLO model's rapid execution time makes it a superior choice; employing a slower but slightly more accurate model like SSD would prolong training durations excessively, undermining the efficiency of our operations.

It is important to note that for this comparison, we used the YOLOv5 model. However, the development of the YOLO algorithm has not stood still; it has seen four additional iterations since YOLOv5, with the latest being YOLOv9, released in February 2024. After careful consideration, we have decided to adopt YOLOv8 for its balance of performance, ease of setup, and current community support, despite YOLOv9's availability. YOLOv8 still offers significant improvements over YOLOv5, making it an even more formidable tool in our arsenal.

Moreover, the advanced capabilities of YOLOv8, which include enhanced accuracy and faster processing speeds due to optimized algorithms and more efficient use of computational resources, suggest that it will not only maintain but exceed the high performance standards set by its predecessors.

These enhancements are crucial as they potentially reduce the need for as many computational resources, thereby lowering operational costs and increasing the feasibility of scaling our processes.

Our choice to utilize the YOLO model for the SenseRator's camera-based object detection system is rooted in its unmatched speed and satisfactory accuracy. This combination is critical in scenarios where real-time processing and responsiveness are paramount. The YOLO algorithm's ability to detect objects in a single pass through the neural network (hence "You Only Look Once") without requiring multiple passes for accurate detection simplifies the computational process, which in turn minimizes latency and maximizes efficiency.

In summary, our continued testing and comparison of object detection models reinforce our decision to implement the YOLOv8 model in the SenseRator project. Its proven speed and upgraded features make it the most practical and powerful option for handling large-scale image processing tasks effectively and efficiently, ensuring that our project remains cutting-edge and operationally viable. Figure 6 below represents the results of the testing conducted to find the ideal machine learning model for this task.

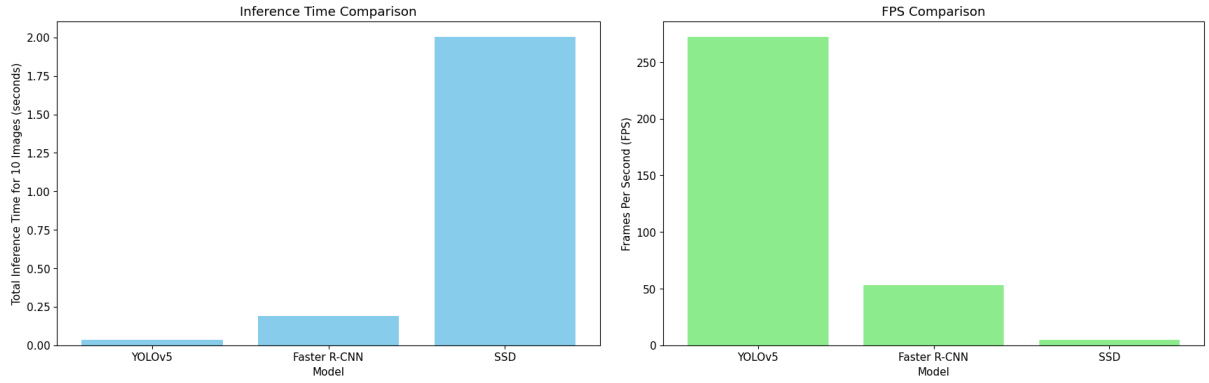

Figure 6: Comparison Between YOLO, Faster R-CNN Model, SSD Model

The green graph on the left compares the frames per second which is how many images a model can process in one second. The right side represents the inference time comparison among the different models which represents the amount of time it takes a model to process a single image.

How YOLO Works

Imagine You're in a Huge Art Gallery

Let's start with a fun analogy. Imagine you're in a huge art gallery with lots of different paintings and sculptures. Your task is to quickly walk through the gallery and make a list of what types of art pieces are there and roughly where they are located. It sounds challenging, right? This is somewhat similar to what the YOLO model does with images.

The Big Picture: Seeing Everything at Once

The "You Only Look Once" name comes from the model's ability to look at the whole image a single time and make predictions. Unlike other systems that scan an image multiple times to understand different parts, YOLO takes one quick look – it sees the entire image in one go, just like when you glance at a scene and take everything in all at once.

Dividing the Image: The Grid System

YOLO starts by dividing the image into a grid, say a 13x13 matrix of cells. Each cell in this grid has a job: it predicts what objects are in that cell and

where they exactly are. Think of each cell like a little detective tasked with exploring its own tiny area.

Predictions and Bounding Boxes

Each grid cell makes predictions. These aren't just any predictions; they are structured guesses. The cell guesses bounding boxes (rectangles that might contain objects) and the likelihood (confidence) that these boxes actually contain an object. Additionally, it predicts what type of object might be within these boxes. The bounding box is defined by its center (x, y coordinates), its width, and its height.

Confidence Scores: How Sure Are We?

Each prediction comes with a confidence score, which reflects how sure the model is that the box contains a specific type of object. The confidence score also includes how accurate the model thinks the box fits around the object (this is where Intersection over Union, or IoU, comes in – it's a measure of overlap between the predicted box and the true box).

Class Probabilities: What's in the Box?

Apart from guessing boxes and their confidence, each cell also predicts the likelihood of the object belonging to different classes (like person, bike, car, etc.). This helps the model not only tell you that there's something in a part of the image but also suggest what that might be.

Making Sense of It All: From Predictions to Decisions

After making all these predictions, YOLO then uses what it knows about objects and combines all the overlapping and redundant boxes it might have predicted to produce a final, clean output that tells you exactly where objects are and what they are. It filters out weaker predictions and clarifies conflicts in overlapping boxes using a process called non-max suppression.

In Summary: Quick and Efficient

YOLO is praised for being exceptionally fast and accurate. It's like having super quick reflexes in our art gallery scenario—glancing quickly at each room and making instant notes about where the important pieces are and what they look like. This makes YOLO incredibly useful for real-time object detection tasks, such as in video processing or autonomous driving, where understanding the surroundings accurately and quickly is crucial.

And that's essentially how YOLO works! It's a smart, efficient system that sees everything at once, makes intelligent guesses, and continuously refines its predictions to provide clear and precise information about where objects are and what they are.

Object Tracking

Object tracking is a necessity to make sure our model is not overcounting objects to create inaccurate data. This is because Detection alone doesn't link objects across frames, so it can lose track of which detected object corresponds to which real-world object. Tracking helps assign consistent IDs to each object across frames, so you know, for instance, that an object in frame 1 is the same object in frame 2. When objects move in and out of view, tracking can also predict where an object is likely to reappear based on

its previous path, improving detection reliability in challenging scenes. In cases where objects are close together or overlap, tracking helps maintain accurate associations over time, reducing the chance of "ID switches" where identities are wrongly assigned to different objects across frames. These features of object tracking are a necessity to make an object detection algorithm remain consistent and accurate.

We have implemented object tracking through the use of a tracker called bytetrack. Bytetrack is included within the Ultralytics library specifically for use on an object detection algorithm. The tracker compares detected objects to previously detected objects to determine whether it is the same object or not. Implementing object tracking has removed the majority of overcounting and has been a great asset in gathering accurate results. In figure 7 below, we can see that the object tracking algorithm assigned an id to a detected object.

Figure 7: Object Tracking with ID

Camera Distortion and Calibration

Most cameras come with a lens that contributes to some sort of distortion within an image. Camera distortion is a type of optical distortion that affects

the accuracy of how an image represents real-world geometry. It typically arises from imperfections in the camera lens or the way light passes through the lens to the sensor. Distortion can alter the shape, size, and positions of objects in the captured image, deviating from their true appearance. There are two primary types of camera distortion: radial distortion and tangential distortion. Radial distortion occurs when the light bends as it passes through the curved surfaces of the lens, causing straight lines to appear curved in the image. Radial distortion is often broken down into two categories.

Barrel Distortion: This makes lines bulge outward from the center of the image, resembling a barrel shape. It's common in wide-angle lenses, where the edges of the image curve outward, making objects appear larger near the center.

Pincushion Distortion: This causes lines to curve inward toward the center, as if being pinched, and is often seen in telephoto lenses. Objects appear smaller near the center and larger near the edges.

Tangential distortion happens when the lens and sensor are not perfectly aligned, causing objects to appear stretched or skewed. It can make objects appear as though they are leaning in a particular direction, especially noticeable near the edges of an image. These distortions cause a large issue for object detection algorithms by causing the object detection algorithm to either miss the object entirely, or lower the accuracy confidence of each object.

The Runcam 2 we are using has a curved lens that slightly bends objects in the video feed that it captures. This means that we need to offset this distortion in order to get accurate results with our object detection

algorithm. To offset this distortion, we must calibrate our camera and remove the distortion within the frame before inference is run on it. This can be done with a checkerboard pattern on a flat surface and a simple algorithm online. Around 10 to 20 images from different angles of the checkerboard should be captured and input into the algorithm. The algorithm then outputs the values needed to undistort an image and we use this to undistort each image that is sent for inference ensuring an accurate detection. As seen in figure 8 below, the camera calibration script draws rainbow colored lines across the edges of a checkerboard pattern to determine the distortion within the camera.

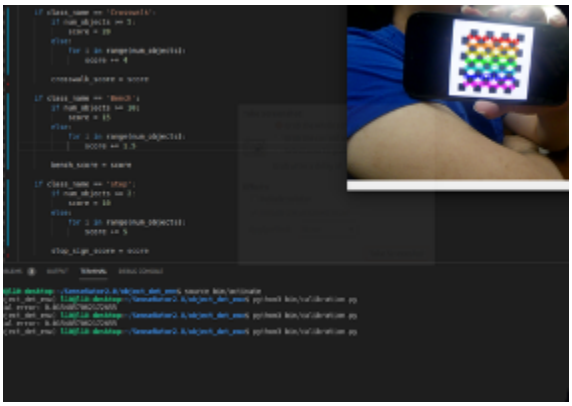

Figure 8: Camera Calibration Software

Data Transfer (From Jetson Nano to Database)

The data that is captured on the Jetson Nano needs to be transferred into a database to be able to be used for display within a web application. To transfer the data gathered from the jetson nano into our database, we store our scores and the video name into a json file that can be uploaded into our database through a simple python script. This script takes our data and inserts it within a json file that can then be referenced when it is in a database. We then check and verify that the json file is correct and get

ready to process it through the script. As soon as we finish running the script, the json file is transferred to the database and is ready to be displayed on our webapp instantly. The webapp reads in the data from the json file in the database and with the region name in each json file, displays that data to the corresponding region.

Video Compression

We used Google Colab to host the video compression code as well as to handle the pip install commands for the necessary installation of library dependencies (i.e. moviepy and ffmpeg-python). Then we hosted the large video files by mounting the Google Drive storage onto Google Colab.

The video compression code uses MoviePy to correct the bit rate of the video as well as to remove background noise from the video and produces a smaller and still smooth video. The further use of MoviePy was allocated to perform the conversion to libx264 codec which helps allow the necessary compression to take place so little storage is needed to be hosted on our Firebase Database.

Furthermore, the libx264 codec is a widely used and highly compatible encoding scheme on major computer systems. Compression of the video is necessary so when users view the video on our web app it does not require as many processing resources to load it up.

Furthermore, ffmpeg-python was used to perform a rigorous validity check to ensure the bit rate would align to the new bit rate of the video as well as its frame rate, and from there it would correct the resolution of the video with limited loss to the quality of the video.

The resulting bitrate that was utilized in the algorithm stemmed from calculating the relation between the requested resolution of the video, the video's frame rate as well as the file size constraint for the Firebase Database. After some trial and error a bitrate was calculated that would perform well in all aspects of the aforementioned constraints.

The encoding scheme was chosen to have a preset of `**fast**` to speed up the compression process of large videos (2.0 gbs+) as well as still being able to preserve the resolution and quality of the video in the webplayer's playback scheme and have a lower buffer rate for lower end computer systems.

To handle the time constraints of the compression algorithm, the removal of the logging feature was necessary as that what preliminary caused the `**Runtime Disconnect**` to appear more frequently than before however, the fix was simple as it was simply setting the parameter to `Logger=None` and that shaved off **62%** of the wait time of the algorithm.

However, as the files were starting to reach the breaking point of 2 gbs+ the algorithm started to falter and take more processing power to do the calculation as well as performing integrity checks which validated the video was still intact and optimal by affirming the frame rate and resolution to be back to the original's specifics.

The only further improvements that can be made to the video compression algorithm is finding a similar compression idea to Burrows-Wheelers Transformation (BWTs) in the attempt to lower the file size even further as well as tackling the long runtime that is approached in the bottleneck that arises in processing large hi-res video files (2K+).

Walkability

Walkability is the ability to safely walk to accessible amenities and services within a reasonable distance. Walkability is an important idea that reduces the need for vehicles while improving the quality-of-life of citizens. Its main use is for high-density urban neighborhoods where citizens can reach services and amenities without the need to ride a vehicle to access those resources in a reasonable amount of time. It also promotes the idea of making the streets complete, livable spaces that serve a variety of uses, users, and transportation modes.

The walkability of a neighborhood can affect public health, air quality, economic development, civic engagement, and cultural resources. Higher walkability scores are linked to economic well-being and are a key strategy against climate change. There are several different walkability indexes that are used to evaluate walkability. The most well-known is Walk Score, which evaluates walkability based on distance to key destinations, population density, and metrics including block length and intersection density. Walk Score does not account for street conditions or other factors that can make the streets less walkable or safe.

Pedestrian Flow & Safety (PFS)

We decided to go more into a smaller section of walkability which is safety and accessibility for pedestrians. The Pedestrian Flow and Safety (PFS) Score is a rating method that measures the efficiency and safety of pedestrian navigation in urban spaces. It considers elements such as: sidewalks, crosswalks, trees, street lights, benches, and stop signs. This score puts more emphasis on features that will benefit pedestrians safety and

accessibility rather than the overall streets with the walkability score. Scoring begins by using our object detection and tracking algorithm to capture our desired features for scoring. The total objects found are then weighted, and compiled to create our scoring metric, which is then displayed on our application.

We are currently using 6 features for this scoring metric.

- Sidewalks: Sidewalks are measured to assess the availability of safe, dedicated walking paths, impacting overall pedestrian accessibility.

- Crosswalks: Crosswalks are included to evaluate safe street-crossing options, directly impacting pedestrian safety and transportation effectiveness.

- Trees: Trees are measured to account for environmental comfort, as they provide shade, improve air quality, and aesthetics of an area.

- Street Lights: Street lights are assessed to determine nighttime visibility, impacting both pedestrian safety and overall accessibility.

- Benches: Benches are considered to evaluate resting opportunities, supporting accessibility and comfort for longer walks.

- Stop Signs: Stop signs are included to gauge intersection safety, as they help manage vehicle flow and make pedestrian crossings safer.

We chose these specific features as we believe they are the most critical features that affect pedestrian safety and accessibility. The way we are

calculating this index is by providing a score for each unique counted object that provides a score up to a specific limit based on the weight of the feature. The weights of each feature have been decided by how important we believe each contributes to the safety and accessibility of pedestrians.

- Trees: 10%
- Crosswalks: 20%
- Street Lights: 20%
- Sidewalks: 25%
- Benches: 15%
- Stop Signs: 10%

In figure 9 below, it showcases our website directory explaining the PFS score, descriptions of why we chose those features, and their corresponding weights.

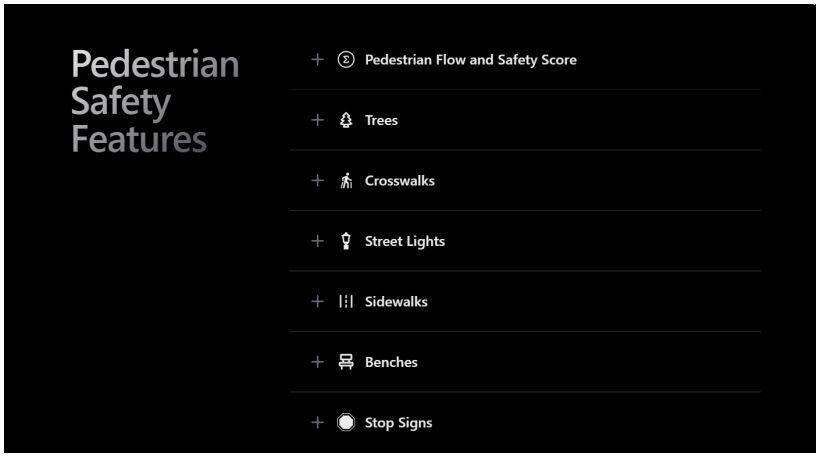

Figure 9: PFS Directory

Hardware Components

System Overview

The SenseRator 2.0 is essentially designed as a mobile data collection platform, mounted on a vehicle (in this case, a bicycle), enabling the dynamic capture of real-time, urban environmental data. In order to accomplish this, select and carefully crafted components were necessary in its design. The following core components allowed our team to collect accurate and comprehensive data in order to calculate region-based PFS scores:

- ***Jetson Nano (Seeed Studio Recomputer J101):*** Acts as the central processing unit of the project, running our software in real-time.
- ***Camera System:***
    - ***RunCam 2:*** A high definition action camera, chosen for its wide field of view, high frame rate, and high resolution, being essential for capturing highly detailed visual data.
    - ***Camera Stabilization System:***
        - ***Z-Axis Stabilizer:*** A custom, 3D-printed, spring-loaded gimbal system that mitigates vertical jolts of motion caused by uneven surfaces the bike might traverse.
        - ***DJI OM5 Gimbal & Mount:*** Custom designed RunCam2 mount to fit the DJI OM5 that counteracts lateral and rotational movements to ensure smoother video footage.
        - ***NOGA Israeli Arm:*** strong, jointed arm to clamp and angle the camera system to the bicycle.
- ***Vehicle Mount:*** A custom 3D-printed assembly of harnesses to fit the Jetson and it's monitor, crucial to securing the Jetson to the Bicycle
- ***Small Monitor:*** An LCD touch monitor to interact with our program.

Figure 10: Full SenseRator Hardware Setup

Jetson Nano

The Jetson Nano, in particular the Seeed Studio J101 model, serves as our main hub of processing and analytics for the SenseRator 2.0 system. Our team selected the Jetson due to its processing capabilities, energy efficiency, and compact size, all of which are capabilities that align with the needs of the project. The decision was largely driven due to the Jetson Nano's ability to perform advanced Machine Learning (ML) tasks, specifically computer vision tasks, such as those essential for calculating the PFS Score.

Equipped with a Quad-core ARM Cortex-A57 MPCore CPU and 128-core NVIDIA Maxwell GPU, the Jetson Nano provides the SenseRator substantial computational power within a small form factor. This hardware configuration enables the execution of complex neural networks and deep learning models while still maintaining critical energy efficiency. When using a mobile computing setup like the SenseRator, having an energy efficient unit is crucial to making sure that the unit is mobile. The GPU acceleration also

facilitates parallel processing of multiple data streams, which makes handling high-resolution video input from the camera system and performing real-time analysis without significant latency possible.

Upon the first week spent with the Jetson during the initial setup phase, our team encountered several challenges mostly related to insufficient memory availability, outdated libraries and packages, and integration with hardware components. The initial solution was related to flashing the operating system (OS) onto the Jetson nano and reconfiguring it for optimal performance and package needs. When we attempted to follow NVIDIA forum guides for how to re-mount the OS to the microSD card that we had added to the system, unfortunately, the system was corrupted and we were unable to restart the device because the OS had lost its launching point into the system.

In order to address this major setback in progress, our team was able to force the Jetson Nano into “forced recovery mode”, which essentially allowed us to access the terminal and memory of the system from another Linux-based device. This was done by using computer jumper wires to short the electrical pins on the carrier board of the Jetson Nano and then connecting the device to another Linux device via a USB-C connection. From here, we were able to reconfigure the system properly and then re-flashed the microSD card with the correct version of the OS we needed to continue working with the Jetson. In order to re-flash the Jetson, we needed to first write the JetPack SDK (which included the necessary Ubuntu OS version and all necessary libraries) and then reconfigure the direct access memory of the Jetson to mount the microSD (now with the OS) as the primary memory access point for the system. Now after rebooting the system, it was correctly configured and ready to be set up for development.

Another issue in setting up the Jetson Nano was the connectivity of peripheral devices such as the camera. Initially, our goal was to utilize the CSI ports built into the carrier board of the Jetson and connect a RaspberryPi Camera Module 2. However, upon initial setup, we realized that the way the Jetson's OS was configured, there was no power being output to the CSI ports and therefore it was not allowing the RaspberryPi camera to be read by the device. This issue was also resolved similarly to the formerly mentioned issue, in that it required a reconfiguration of the OS in order to allow the device to send power to the CSI ports and then to reboot the device.

A critical aspect of the project involved running resource-intensive object detection and tracking algorithms on the Jetson Nano, without compromising real-time performance. Our team employed the YOLOv8 model, known for its speed and accuracy, working in tandem with the ByteTrack algorithm for multi-object tracking. The custom trained YOLO model was optimized for detecting urban features relevant to the PFS score, including: sidewalks, crosswalks, benches, stop signs, trees, and street lights.

Implementing these algorithms on the Jetson presented several challenges:

1. ***Model Optimization:*** In order to ensure that the neural network models could run efficiently on the Jetson's hardware constraints, our team had to carefully select our dataset input for training, our weight of our model, and the size of it as well. By experimenting with different parameters in our model training, we were able to obtain fairly high confidence scores in our detections, while also maintaining a relatively quick inference speed in real time.
2. ***Memory Management:*** the limited memory resources of the Jetson Nano required efficient memory management within the application. A

combination of techniques was employed such as pre-allocating memory, minimizing data copies, and optimizing our code to process frames in batches then release unused memory promptly. We also used a Python Virtual Environment in order to set up our coding environment with memory efficient use of storing all required packages and libraries.

3. ***Concurrency & Multithreading:*** To maximize the utilization of the CPU and GPU of the system, we designed our application with multithreading in mind. We created separate threads for video capture, inference, and result processing. A thread-safe queue was implemented to handle frames awaiting processing, making sure that the video capture was not hindered by processing delays.
4. ***Video Pipeline Optimization:*** The team faced challenges with the GStreamer pipeline, which is crucial to handling multimedia streaming on the Jetson Nano. Initial configurations not only resulted in latency and dropped frames, but also major synchronization issues between our cameras and the processing unit. By experimenting with different cameras (eventually using the RunCam2 for its reliability) and various pipeline parameters, we were able to achieve a stable and secure video stream.
5. ***Real-Time Performance Metrics:*** To monitor and ensure real-time performance, we scripted our application to measure frame processing times and calculate the average frames per second as well as the inference time. This allowed us to identify and find bottlenecks in the processing pipeline and verify our system met the necessary requirements for real-time calculations.
6. ***Thermal and Power Management:*** Running intensive computations on the Jetson raised concerns about thermal management of the system. Prolonged operation led to increased temperatures as well as

system throttling, risking hardware damage. The first solution was the implemented heat sink that came with the Jetson from the Seeed Studio carrier board package. The team also added internal fans that kept the system a significant few degrees cooler, especially useful when using the system in Florida September heat as we worked on the project. For power management, the use of a wireless power pack, intended for charging larger devices, was used and was able to deliver the voltage we needed to ensure the Jetson performance was consistent.

Camera Selection

The selection of an appropriate camera was a critical aspect of the SenseRator 2.0, directly impacting the quality of data collected for calculating the Pedestrian Flow and Safety Score. The camera needed to provide high-resolution images with a wide field of view while being compatible with the Jetson Nano's processing capabilities.

Initially we selected the RaspberryPi Camera Module 2, widely used and compatible with the Jetson Nano through its Camera Serial Interface (CSI). We selected this camera due to its ease of integration, affordability, and availability of documentation and support. Its 8 megapixel sensor with 1080p and 30fps specs seemed sufficient enough for preliminary testing of the SenseRator. Despite its advantages, we struggled with its significant limitations that we came to realize as we continued working and developing our project. The camera had too narrow a field of view to capture all the details we needed to calculate our PFS. Its image quality, while good enough for testing, did not perform outdoors in varying light conditions and was suboptimal and caused issues like motion blur and low dynamic range.

Mounting the RaspberryPi Cam was also an issue as it was small and too lightweight to be able to secure its positioning as well as the lack of built in housing and very small connection cord that got caught up by the wind.

Next, our team attempted to switch our camera to a small generic webcam that only lasted our team about a week with testing. It caused many pipeline issues and crashed our system, forcing us to reboot often, and its dynamic range was less than optimal. Although the webcam was easier to connect through its USB-A connection and had a wider FOV, the issues with its handling of exposure settings as well as instability kept this camera from being useful to our project

Finally with more research, we were able to find a camera that suited our needs perfectly, the RunCam 2. The RunCam 2 offered us several key advantages that were able to fit our project's needs. It had a high resolution of 1080p and a framerate of up to 60 fps, it had a wide field of view capability so that we would be able to capture much more data with each frame, and it was a small and durable action camera suited for outdoor usage with a USB connection mode. One thing that made the camera so easy to work with was its compatible app that allowed us to adjust camera settings such as exposure, frame rate, and even field of view to be able to fit our project's exact needs.

The main drawback that we faced by switching to the RunCam 2 was that our detections became much lower in accuracy and much less frequent. After extensive testing, our team came to realize that this was caused by the wide field of view, which was distorting the shapes of each object in each frame. Essentially, straight lines warped outward from the center, meaning our model was not detecting shapes because the pixels were misplaced.

With our solution mentioned above, we were able to counteract the distortion of the camera and fix our detections by using a camera calibration script that calculated the distortion of the camera using a chess board and then adjusted each frame's pixels to undistort each one.

The next issue that we needed to face and one that would serve to be one of the most critical challenges of the project, was how to stabilize our camera. The camera stabilization system would prove to be an absolutely critical component to the SenseRator 2.0, directly influencing the quality of the video input and data captured, and consequently the PFS score calculations. The mobile nature of the data collection platform (mounted on a bicycle) introduced significant motion that needed to be mitigated to ensure reliable object detection.

Camera Stabilization System

Capturing stable footage while cycling presents unique challenges, in particular with vertical or Z-axis motion caused by the bike traversing uneven terrain. This challenge was not as simple as attaching the bike to a standard camera gimbal, because they typically primarily focus on X and Y-axis stabilization or horizontal and rotational movements, but not the vertical jolts that could be faced while riding a bike over a bumpy road or path. The vertical jolts often resulted in motion blur, inconsistent frames, and very difficult-to-watch video i/o, adversely affecting the performance of our object detection and tracking algorithms by reducing the accuracy and reliability. To address these challenges, we worked to design a comprehensive stabilization system, focusing on both mechanical dampening and the advanced gimbal technology available to counteract unsteady motion across all three axes.

The Z-Axis Stabilizer was the first crucial step in designing a stabilization system for the camera, which is designed to function similarly to a car shock absorber by dampening any vertical motion. The core of the stabilizer is a spring-loaded mechanism with an adjustable tensioner, allowing fine-tuning of the spring strength to accommodate different terrains and camera needs. The adjustability is important for ensuring precise stabilization without introducing excessive bounce or stiffness.

The Z-axis stabilizer went through about 4 iterations of improvements before the final design. Each version was fabricated using 3D printed technology, It was designed and configured to model classic Z-axis stabilizers often used in videography, only with customizable and adjustable hardware and knobs that allowed the device to properly fit the adjustable Noga Israeli Arm that held it onto the bicycle through a ¼-20 mm screw and socket. Pivot points were incorporated into the design to allow for necessary rotational flexibility. However, initial iterations faced issues related to friction at these pivot points as well as a wobble, caused by an unrefined and loose connection between points. Tightening the bolts to enhance the stability greatly compromised the smoothness of movement required for the gimbal, while loosening them led to unwanted side-to-side (X-axis) play, affecting the lateral stability.

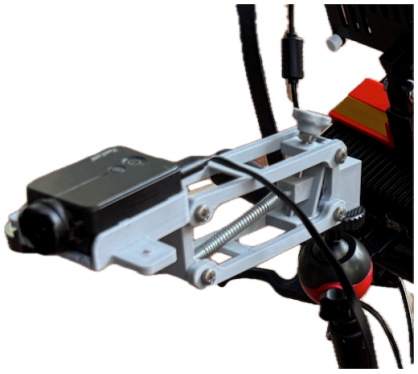

Figure 11: Version 1 of Z-Axis Stabilizer

To resolve these issues, the stabilizer underwent several design refinements:

- ***Bearing Integration:*** Bearings were added at each pivot point of the design to eliminate friction without introducing any lateral wobble. This modification allowed for smooth, unrestricted vertical movement, significantly improving the effectiveness of the Z-axis dampening.
- ***Joint Grease:*** Along with the bearings, we also added a light grease to the joints responsible for the friction and tightness of the gimbal in order to ensure that the pieces moved smoothly to enhance the vertical motion dampening.
- ***Structural Reinforcement:*** The components were redesigned with sturdier structure and thicker connecting components between them to enhance the durability and be able to withstand vibrations and outdoor environmental changes. This also gave more weight to the system which helped against outdoor factors such as wind.
- ***Fine-Tuning Mechanisms:*** The adjustable tensioner was refined to provide more precise control over the spring resistance, enabling quick adjustments based on real-time feedback during field testing. A design component of this was that it could be a swappable spring in order to account for potential future camera system changes that might include heavier or lighter components.

These iterative improvements resulted in a Z-axis stabilizer that could effectively dampen vertical motion while maintaining stability in other axes. However, there was still multi-axes stabilization to figure out and more measures were required. In **Figure 12** you can see the Z-axis stabilizer in its near final design stage. In this iteration it was fitted with a custom made RunCam2 Mount that was removable.

Figure 12: Z-Axis Stabilizer V.6

Integration with the DJI OM5 Gimbal was necessary for the success of the camera stabilization. Recognizing the need to address lateral and rotational motion (X & Y axes), the DJI OM5 gimbal was integrated into the system. The DJI OM5 is a professional-grade gimbal designed for smartphone stabilization, capable of countering roll, pitch, and yaw motions through fine motorized adjustments.

Adapting the gimbal for the RunCam 2: (Shown in Figure 13)

- ***Custom Mount Design:*** A 3D-printed mount resembling the size and shape of a smartphone was created to hold the RunCam2 inside a hole printed with grooves to secure and lock it into place. The mount was designed to fit within the DJI's clamp mechanism and included slots for counterweights in order to achieve the proper balance needed for stabilization on the gimbal
- ***Balancing and Calibration:*** Counterweights were added to offset the weight difference between the smartphone shaped mount and the RunCam2, ensuring that the gimbals motors operated without strain.

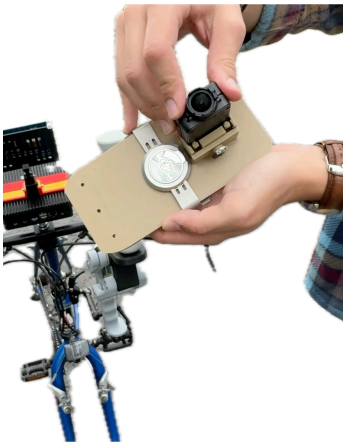

Figure 13: 3D Printed RunCam to DJI adapter mount

The combined system provides a comprehensive three-axis stabilization:

- ***Z-Axis Stabilizer:*** Managed vertical motion caused by uneven terrain that the bicycle might encounter, reducing all up and down bounces that lead to motion blur
- ***DJI OM5 Gimbal:*** Addressed lateral and rotational movements, keeping the camera oriented correctly and eliminating shaking and rotational jitters.

Figure 14: Z-axis stabilizer with DJI attachment and RunCam Adapter

This combination ensured the camera remained stable regardless of the bike's motion and greatly enhanced the quality of our captured footage.
Field Testing of the camera system demonstrated substantial improvement in the quality of our video, now with the dual stabilization components that we had added. This was a milestone in the project.

Initial Recordings without any stabilization had exhibited bouncing, shaking, and leaning, making it difficult for our model to run accurate inference on each frame that it processed. Footage often came out blurry and objects were out of focus and bounding boxes were not appearing on the footage either. Items also entered an exited frame too fast for the model to run inference on because when the bike moved or tilted in order to turn, the camera would immediately pull away from the object that it was currently focused on.

When the footage had become stabilized with our combined stabilization system, the footage was now smooth, clear, and had minimal motion blur to each frame that it captured. Objects maintained consistent positions between frames and the overall quality was enhanced greatly as well. With the DJI OM5’s settings, the camera would also stay positioned in its current direction and slowly pan to catch up with whichever way the bike now faced. This allowed our algorithm to run more inference on objects that were in view at the same time allowing us to ride the bike without worrying of riding perfectly in a straight line at an object at all times.

The rate of success in object detections improved approximately by 25 - 30% due to the reduction in motion blur, the improved field of view, and the fixed camera distortion. Our stabilization hardware solution had also reduced the number of false detections that could have been caused by motion blur.

Due to the reason objects now stayed in frame longer and moved much more predictably between frames (instead of jolting to different positions from the unsteadiness of the bicycle), the tracking accuracy of our ByteTrack implementation greatly increased. From one ride prior to including our new system to the one where we did, our detections went from some [2 trees, 1 street light, 1 sidewalk] to a sample of data that resembled something like [78 trees, 4 crosswalks, 17 streetlights, 6 sidewalks] with much greater accuracy. This essentially solidified that our detections, tracking, and scoring were now working how we wanted them to, and mostly because of fixing the lack of proper hardware required to give our YOLO model the resources to perform properly.

Throughout the development of the stabilization system, we encountered a multitude of technical challenges:

- ***Friction management:*** Initial designs faced friction and wobbling in the pivot points. The integration of bearings resolved these along with greasing a few of the joints, allowing for smooth movement, without compromising the stability and durability of the system.
- ***Integration with existing hardware:*** Ensuring that as we added pieces to our system, they would all be compatible as to not create double the amount of hardware development needed, was crucial. Creating a system that allowed for compatibility between the Z-axis stabilizer, the DJI OM5 gimbal, and the RunCam 2, all attached to an Israeli camera arm, required careful planning and the custom creation of all the mounting components.

The final stabilization system proved to be both practical and effective. Field tests conducted around campus confirmed that the dual stabilizer effectively smoothed footage, delivering significant improvements in video quality

compared to initial setups. The adjustable tension of the spring inside of the Z-axis stabilizer allowed for quick adaptation to conditions, and the integration of the DJI gimbal provided the missing piece to ensuring a fully comprehensive stabilization system. The iterative design process and innovative solutions implemented in the system proved difficult but great challenges to overcome through careful planning and adaptation.

Figure 15: Camera Stabilization System on Bicycle

Vehicle Mounting Solutions

The effective mounting of the hardware components of the SenseRator 2.0 onto the bicycle was a critical aspect of our project, ensuring stability and safety of the system during data collection. The challenge presented was now to take the entirety of the slightly heavy camera stabilization system that we developed, along with the Jetson Nano itself, and to attach it to the bike in a way that provided ease of riding for the bike rider, access to the hardware to make adjustments on the fly, to be secured as to not damage any of our hardware, as well as have ease of installation and removal.

The design of the vehicle mounting solution was completed using 3D printing as well as the combination of various tools and clamps available to our team. This approach allowed for a rapid prototyping, customization, and cost-effective production of parts tailored to our project needs.

- **Jetson Nano Mount:**
    - **Design:** The mount for the Jetson Nano consisted of a 3D-printed strap that fit snugly around the case of the device. This strap was connected and tightened by screws on both sides ensuring a secure grip around the Jetson Nano.
    - **Handlebar attachment:** The bottom of the strap featured a screwable clamp designed to attach firmly around thin walls or cylindrical objects. We took advantage of this to place the Jetson on top of the handlebars of our bicycle. This provided easy access for monitoring and maintenance while keeping the device out of the way of the rider’s movements.
    - **Stability:** The design minimized any vibrations that might have knocked the Jetson loose, ensuring it's safety in the system through the whole bike ride.
- **LCD Touchscreen Mount:**
    - **Design:** A second strap was designed using the same shape as the Jetson Mount to fit around the Jetson, this time incorporating an adjustable angled bracket for the 5” LCD touchscreen.
    - **Adjustability:** The angled mount could be adjusted to accommodate the riders height and preferences to reduce glare from the sunlight.
    - **Integration:** This allowed minimal cables and connections and minimized parts needed. The assembly of this component maintained a compact profile and kept the system simple.

The assembly and installation of the hardware components involved integrating the custom mounts with additional equipment to create a cohesive and functional system.

- **NOGA israeli arm:** The Noga arm is a flexible, articulated arm commonly used in photography and industrial applications for precise positioning of equipment. It provided our team the versatile and sturdy mount needed for mounting the camera stabilization system to our bicycle.
- **Camera System Mounting:**
    - **Attachment:** The base of the NOGA arm was securely attached to a fixed point (that did not rotate) on the bike frame's front fork.
    - **Adjustability:** The adjustable joints allowed for precise positioning of the camera and stabilizer assembly, enabling our team to align the camera how it was needed for optimal data collection.

The 3D-printed Jetson Nano mont was clamped to the bicycle handlebars using the screwable clamp mechanism. This was enough to keep the Jetson from shifting or wobbling during a ride. It also faced the ports on the Jetson away from the rider in order to keep all the cables away from their knees so that they wouldn't get in the way. In order to power the Jetson while on the go, the rider would wear a small shoulder pouch with a large portable power bank inside of it, using it to connect the power cable and power the device for a few hours at a time.

Figure 16 & 17: The Jetson Nano Bike Attachment & Touchscreen Mount

Web Application

Overall Idea

SenseRator version one processed data locally without any platform to display the results generated by object detection models. However, in version two, we agreed with the sponsors of the project to create a user-friendly web app to show the results and make it accessible to all users. The core concept behind the web application is to develop a centralized platform that combines the outputs from AI/ML object detection models. The intention is to integrate these results with a 2D map interface, which will display ratings related to various areas. This visual integration aims to provide a comprehensive analysis tool that enhances understanding of the physical environment.

Furthermore, the web app is designed to create connections between our team and various external partners, including corporations and governmental entities like neighborhood associations or real estate agents. They often aim to enhance and promote features such as walkability, accessibility, and cleanliness within their localities. By providing a detailed, interactive map that illustrates these features through direct data visualization, the web app seeks to serve as a crucial tool in their promotional and development efforts.

I was tasked with researching all possible options to create the web app and identifying the most suitable stack capable of handling heavy data processing and other requirements. To determine the optimal stack, we first need to understand the types of data the web app will handle.

Market Need

The development of the SenseRator Version 2 web application is a key step forward in urban planning by using artificial intelligence and machine learning. Modern cities face many challenges including growth, sustainability, and safety. By putting these advanced tools into an easy-to-use platform, the SenseRator Version 2 aims to solve these problems and make cities more attractive places to live. This could lead to more people choosing to live in these improved areas, helping to create more lively and sustainable communities.

User Targeting

The primary users of the web app are professionals and organizations involved in urban planning. This group includes city planners, real estate developers, environmental consultants, and governmental entities like neighborhood associations. Additionally, individuals searching for new and intelligent places to live will find the app useful. These users will benefit greatly from the app due to their need for accurate and comprehensive data for making critical decisions about infrastructure development and resource management. Furthermore, the app's ability to display precise ratings on walkability, accessibility, cleanliness, and other parameters—generated by AI and ML models—makes it a valuable tool for real estate agents looking to enhance their businesses.

Architecture

The architecture of the web app is designed to efficiently handle large-scale data and enable real-time interactions between users and the backend. To achieve this, we utilize one of the commonly used software design patterns, MVC (Model-View-Controller), which organizes software into three main components:

- Model: This component represents the data logic and the data stored within the application. It also provides a way for the user interface to access this data.

- View: This is where the user interface is located. It is what users will see and interact with, containing all HTML markup, CSS stylesheets, and any other logic.

- Controller: This component is responsible for handling user interactions and performing any necessary logic to be displayed to the user, essentially connecting the model and view components.

Using this pattern is particularly helpful for large-scale applications. It aids in development, testing, and debugging processes. Moreover, this approach allows a large team to work on different components separately and integrate their work later.

Responsive Design

One of the key design choices is implementing responsive design in the web app, which means ensuring that the web app operates well across all screen sizes, such as mobile and desktop screens. Additionally, it is important to ensure that all functionality works well regardless of screen size. To achieve

this, the web app employs a fluid design strategy, utilizing grids that divide the page width into several columns, with content placed accordingly as shown in Figure 18. This layout facilitates automatic adaptation to different screen sizes by adjusting the layout of components. CSS media queries are the most commonly used method to apply varying styles and rules depending on the device in use.

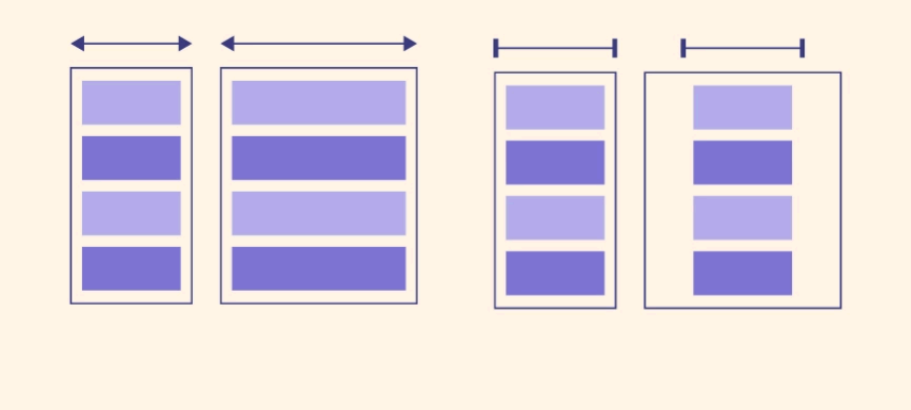

Figure 18: A visual demonstration on what responsive design is, and how it is fit to work for different screen sizes

User Interface

The user interface of the web app is designed for clarity and ease of use. To create an optimal UI, we adhere to several key principles:

1- Clarity: This principle ensures that the role of each element, especially interactive ones like buttons, is clear and contributes to intuitive navigation. It's crucial that all elements and components within the web app are easily understandable to avoid confusion.

2- Familiarity: As per Jacob's Law, which states that 'Users spend most of their time on other sites, and therefore prefer your site to work the same way as all the other sites they already know,' we must design the web app so that it feels familiar to users.

3- Hierarchy: This critical design principle is widely used in modern websites. Implementing it helps users understand the importance of each element, encouraging them to take desired actions. Hierarchy can be achieved through

- Size: Making key elements larger than those of lesser importance.
- Fonts: Using various font sizes and weights to highlight important headlines.
- Color: Employing different shades of a specific color to establish visual hierarchy.

As shown in Figure 19 an example of using this principle:

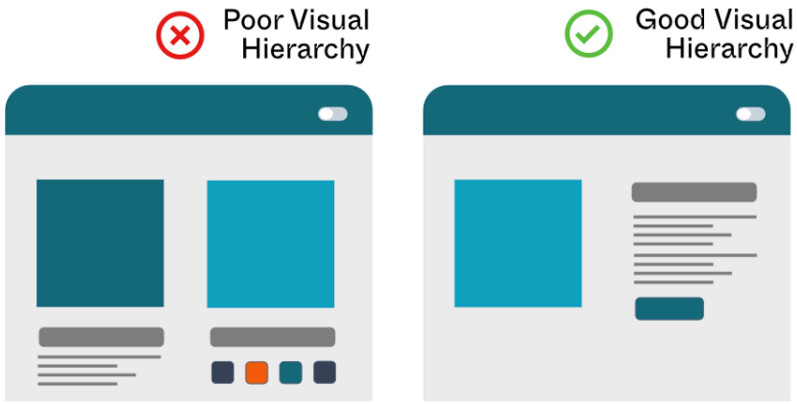

Figure 19: Visual Hierarchy Example

4- Flexibility: This is achieved by incorporating various features that enhance user experience, such as search functions and shortcuts.

5- Accessibility: Our web app aims to be accessible to a diverse range of users, including those with disabilities such as visual and hearing impairments. According to the World Health Organization, there are 285 million people with visual impairments and 360 million people with hearing loss globally. To improve accessibility, we will:

- Follow the Web Content Accessibility Guidelines (WCAG), which include using specific contrast ratios and offering guidelines to avoid common design errors as shown in Figure 20.
- Ensure all images have appropriate alt attributes.
- Make the website easily navigable via keyboard.

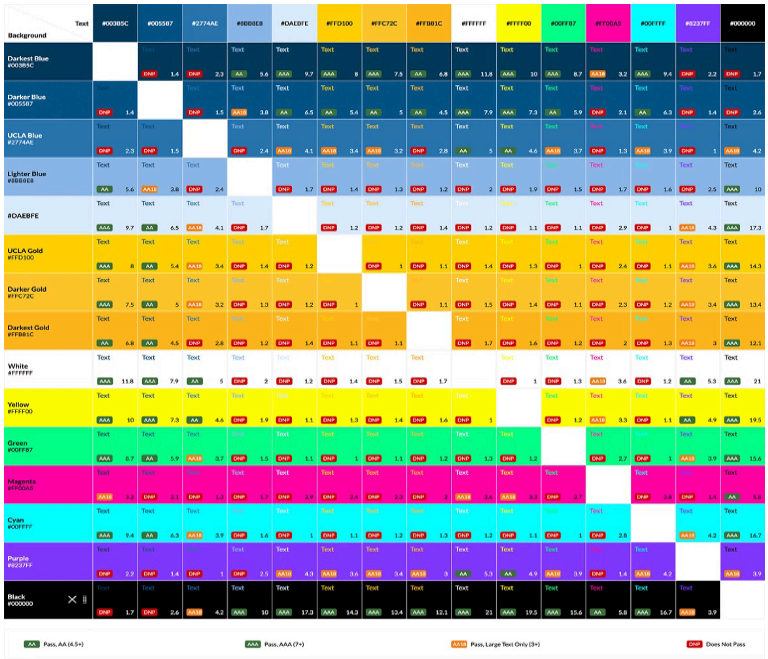

Figure 20: Web Content Accessibility Guidelines (WCAG)

Build/Prototype/Test Plan

Choosing MERN for Web Development

In our search to select the ideal development stack, we evaluated several technology stacks before making our final decision.

1. LAMP Stack:

   LAMP, consisting of Linux, Apache, MySQL, and PHP, is a traditional stack with a long history of use.

2. MEAN Stack:

   The MEAN stack, designed for building dynamic websites, includes MongoDB (database), Express.js (backend framework), Angular.js (frontend framework), and Node.js (runtime environment).

3. MERN Stack:

   Similar to MEAN, the MERN stack is used for developing dynamic

websites but incorporates React’s virtual DOM, which improves performance and page load times. It consists of MongoDB (database), Express.js (backend framework), React.js (frontend framework), and Node.js (runtime environment).

Our Initial Choice:

Initially, we opted for the MERN stack, using React for its efficient user interface capabilities and MongoDB for its flexibility in handling structured data. However, as the project's requirements evolved, we recognized that the web application would heavily rely on real-time updates and efficient handling of video data.

Switching to Firebase Services:

To meet these evolving requirements, we decided to continue using Express.js for the backend but transitioned to Firebase for data storage and real-time functionality. Firebase provides robust tools tailored for modern web applications, including:

- Firestore Database: For storing structured and semi-structured data with real-time syncing capabilities.
- Firebase Storage: Specifically designed for storing and serving large multimedia files, such as videos.
- Real-Time Database: To facilitate dynamic updates for live data visualization.

Advantages of Firebase with Express:

- Real-Time Functionality: Firebase's real-time capabilities ensure data is instantly updated across users and sessions, critical for dynamic map visualizations.
- Multimedia Storage: Firebase Storage offers a scalable and efficient solution for storing and managing video files.

- Backend Efficiency: By continuing to use Express.js, we retain the flexibility of a robust server-side framework while integrating Firebase for its specialized backend services.
- Simplified Integration: Firebase SDKs easily integrate with the Express framework, allowing seamless development.

Application Architecture:

1. Frontend:
    - React.js remains the framework for building the user interface, ensuring modularity and performance.
2. Backend:
    - Express.js continues to serve as the primary backend framework for server-side logic and API development.
3. Data Storage:
    - Firestore Database: Manages structured data, such as user information and metadata, with real-time syncing capabilities.
    - Firebase Storage: Handles video and other large multimedia files for efficient storage and retrieval.
    - Real-Time Database: Supports live updates and dynamic data visualization on the web app.

By combining Express.js with Firebase services, we ensure a powerful, scalable, and efficient system capable of meeting the application's needs while maintaining an efficient development process.

Front-End Technologies

1- Main language: JavaScript was the most popular programming language in 2023. According to a survey by Stack Overflow, 63.61% of developers used JavaScript, followed by HTML and CSS at 52.97%, as shown in Figure 21. This highlights the importance of JavaScript in making websites, as it is

the main language used for modern web applications. Our team has decided to use JavaScript as the main language for developing the SenseRator version two web application. Also, JavaScript works on all browsers, which helps our web application reach as many users as possible.

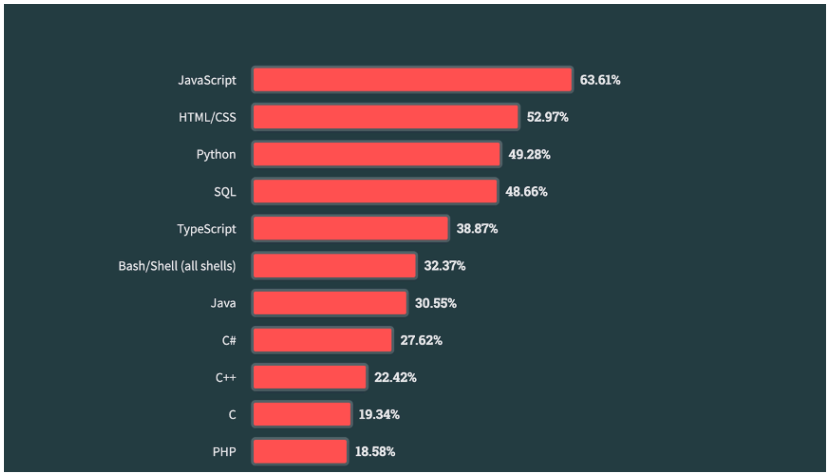

Figure 21: Results of survey by Stack Overflow

2- Main library (React): React is one of the most widely used JavaScript libraries. It was developed by engineers at Facebook (now Meta) in 2010. React is known for its capability to build user interfaces using components. It works by breaking down the main sections of a web app into smaller, reusable components. Additionally, React employs a virtual DOM (Document Object Model), which is essentially a node tree that holds the elements, as shown in Figure 22. It then renders these elements with the browser's actual DOM. Thanks to this approach, React is efficient at updating page content.

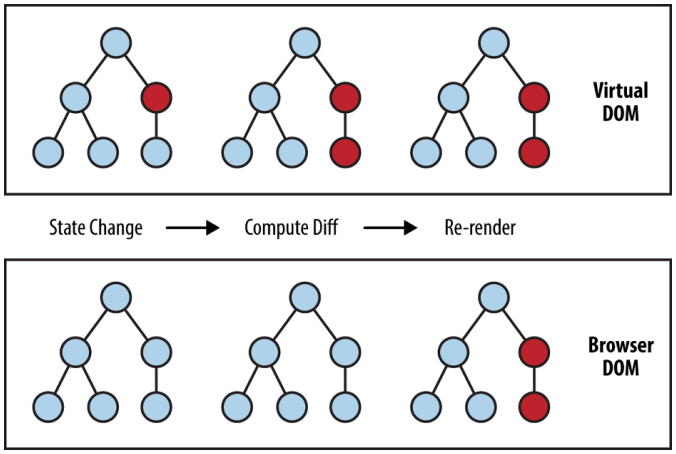

Figure 22: How React employs a virtual DOM

Since we are primarily using React for our web application development, I have prepared a brief overview on how to begin developing with React.

1- Setting Up Your Environment and Creating a React Application:
To set up the environment for developing a React application, first ensure that Node.js and npm (Node Package Manager) are installed on your computer. Then, you need to create the app. There are several methods to start a React project, but for simplicity, we will use the 'create-react-app' utility. This method involves executing the following commands in your terminal:

- npx create-react-app <name-of-project>
- cd <name-of-project>
- npm start

2- Understanding Key React Concepts

- Components: Components are crucial in a React application as they can be reused throughout the app and contain both logic and JSX code.

- JSX: JSX is an alternative syntax to write HTML within a React app, which looks similar to HTML but allows the integration of JavaScript directly within the code line. This is particularly useful for creating dynamic components.

- State and Props: State in React is commonly managed using hooks, particularly the useState hook. State is crucial as it triggers a re-render when users interact with elements in the app. Props, on the other hand, are data passed to child components from their parent component, similar to function arguments in other programming languages.

- Hooks: Hooks are an essential part of React, with various hooks available for different needs. For instance, useState is used to manage state within the app, while useEffect is employed to execute functions after every component render.

3- Building and Running

After understanding the fundamental concepts of a React application, we can begin developing web applications. I typically use Visual Studio Code as the primary IDE for React development because it offers many extensions that enhance the development process. To build the React application, you simply run the following command:

  - npm run build

After this, the application is ready to be deployed. For more details, it is always a good practice to review the official documentation, which you can find at this link: https://legacy.reactjs.org/docs/getting-started.html

2- Styling (using Tailwind CSS)What makes a good web application better than a poor one is its overall design. Most modern web applications are carefully designed to offer the best user experience. Styling web applications is mainly done using CSS (Cascading Style Sheets), which helps control things like color, font, the size of elements, and their placement on the page. There are many ways to apply CSS, but for our application, we decided to use Tailwind CSS.

Tailwind CSS is a utility-first framework that offers a lot of helpful classes such as ‘flex’, ‘text-center’, and many others. What makes Tailwind CSS different from traditional CSS is that it lets you add styling classes directly to HTML elements. This method makes it easier and faster to style components. Plus, it allows us to include logic within these classes, making it easier to customize components to our needs.

To further enhance our styling approach, we also incorporated NextUI into the application. NextUI is a modern React-based library that provides pre-designed and customizable UI components, allowing us to achieve a sleek and professional look with minimal effort. Its responsive design system and accessibility features complement Tailwind CSS, ensuring the application is both visually appealing and user-friendly.

3- State management (Redux): One of the most important aspects of developing a React application is managing the state. State is crucial in React because it enables dynamic interactions within the application. To effectively manage state, especially when targeting a broad range of users, we use a state management library called Redux. Redux is a JavaScript library that acts as a container holding all the states in a specific application, as illustrated in Figure 23.

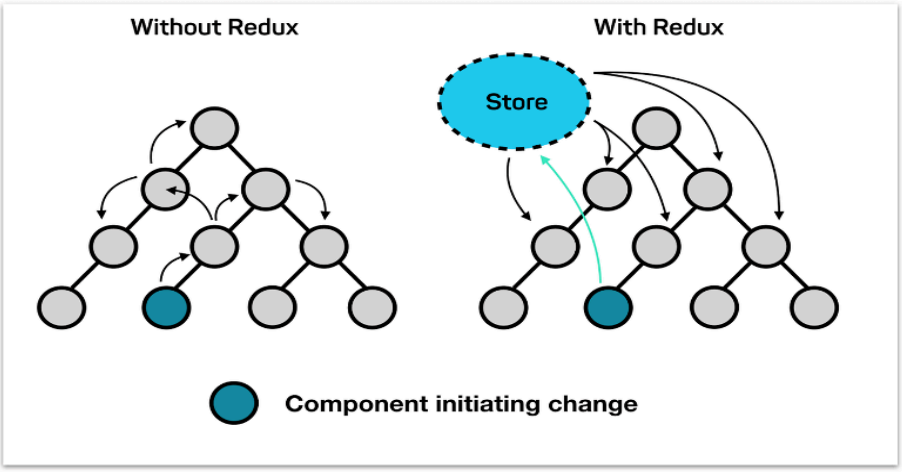

Figure 23: How Redux container works

It ensures that applications behave consistently across all components. Additionally, Redux allows for easy access to any state throughout the application without the need to pass the state via props. This also makes testing and debugging the application easier, as all states are centralized in one location.

Database Selection

After careful consideration, we have opted to utilize Firebase for this phase of the project due to its seamless integration with our web application and its robust support for handling dynamic and large-scale data. This decision

aligns with our current priorities, which focus on efficiently managing structured data and multimedia storage without the additional complexity of handling LiDAR point cloud data.

Why Firebase?

- Real-Time Database: Firebase's Real-Time Database enables live synchronization of dynamically changing data, such as area ratings, directly within the application.
- Cloud Firestore: A scalable and flexible NoSQL database, Cloud Firestore is ideal for structured and semi-structured data storage, ensuring quick and efficient querying.
- Firebase Storage: Specifically designed to store and serve large multimedia files, Firebase Storage will be instrumental in managing video data and other assets needed for the application.
- Scalability: Firebase offers built-in scalability for both its database and storage services, ensuring the system can grow alongside the application as user demands increase.
- Ease of Integration: Firebase's comprehensive SDKs allow seamless integration with the existing backend powered by Express.js, simplifying development while maintaining performance.

Benefits of Switching to Firebase

- Simplified Development Process: By utilizing Firebase, the need to maintain and manage separate systems for structured data and multimedia files is eliminated.
- Cost-Effective and Scalable: Firebase's pricing model and scalability are well-suited for the project's current scope.

- Future Readiness: While the application won't render or store LiDAR point cloud data at this stage, Firebase offers the flexibility to integrate additional features as the application evolves.

In summary, this shift to Firebase allows us to focus on delivering a robust and efficient application while leaving the complexities of rendering and managing LiDAR point cloud data for a future phase. Firebase's comprehensive suite of tools ensures we can meet current project needs while remaining adaptable for future expansions.

Back-End Technologies

Most of the core logic and operations of web applications are managed by the backend, which is responsible for data management, security, and performing computational tasks that are not visible to users. In our web application, we will use the backend to retrieve and process data from our database. The backend of the MERN stack consists of Node.js and Express.js.

Node.js is an open-source JavaScript runtime environment that allows developers to run JavaScript on the server side. It uses Google's V8 engine to compile JavaScript directly into native machine code, enhancing the performance of the web application. Additionally, Node.js has huge community support and is widely used, making web application development more accessible. There are also many libraries and frameworks available, such as Express.js.

Express.js is a well-known framework for Node.js. It offers numerous features and tools for developers, simplifying the process of calling and managing APIs. Express.js includes middleware functions that execute code

and handle requests and responses. These functions operate in the middle of these calling objects, as illustrated in Figure 24. Since we are primarily using Node.js and Express.js for our backend, I will provide a brief explanation of how to create a simple backend server using Express.js.

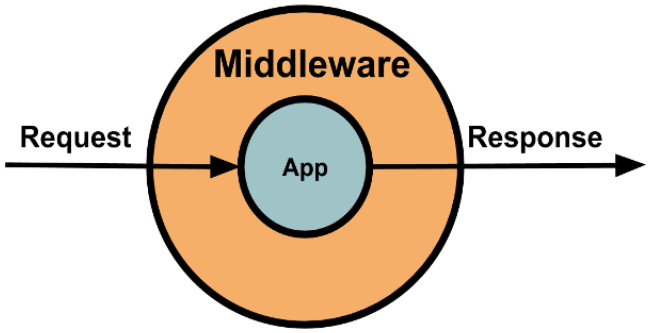

Figure 24: How the middleware works

To set up a simple backend server using Express.js, follow these steps:

- First, ensure that Node.js and npm are installed using these commands:

  § node -v
  § npm -v

  These commands will display the versions of Node.js and npm installed

- Then, enter the following commands in the terminal:

  § npm init -y
  § npm install express

- Next, after creating a file named index.js or another name, write these lines of code into the file:

1- const express = require('express');
   - This line imports the Express module.

2- const app = express();
   - Initializes a new app of an Express application

3- const port = 3000;
   - Sets the port number to 3000.

4- app.get('/', (req, res) => {
   - Defines a route to GET requests to the URL ('/').

5- res.send(' Building Server!');
   - Sends the text "Building Server!” back to the client as a response.

6- app.listen(port, () => {
   - make the app to listen on the specified port (3000

7- console.log(Server running at http://localhost:${port}/`);`
   - Print the Output of message that confirms the server is running and displays the URL.

Web Development Libraries

In the development of web applications, we often integrate various libraries, whether they are prebuilt or custom-made for specific tasks.
Here is a list of some prebuilt libraries we use:

1- Axios
   - Description: Axios is a promise-based HTTP client for Node.js, known for its ability to perform CRUD operations (Create, Read, Update, Delete).

   - Benefits: It is particularly useful for fetching data from the backend to the frontend. Additionally, it automatically serializes request bodies to JSON, which is advantageous as we frequently use JSON, especially with point cloud data.

   - Documentation link: https://axios-http.com/docs/intro

2- Cors (Cross-Origin Resource Sharing)
   - Description: Cors is a security feature that allows servers to identify origins (such as domain, scheme, port) and manage cross-origin requests.

   - Documentation link:https://www.npmjs.com/package/cors

3- Dotenv
   - Description: Dotenv is used to load environment variables, like database environment keys. This tool is valuable for managing credentials securely during development.

- Benefits: It allows environment variables to be stored in a separate file, which keeps them out of source control.

- Documentation link: https://www.dotenv.org/docs/

4- Vite

- Description: Vite is a frontend build tool that enhances the development experience by integrating the native module loader.

- Benefits: It offers hot module replacement, which speeds up rebuilding the application.

- Documentation link: https://vitejs.dev/guide/

5- Nodemon

- Description: Nodemon is typically used on the backend to automatically restart the server when code changes are detected.

- Benefits: It saves a significant amount of development time and simplifies backend development.

- Documentation link:
  https://www.npmjs.com/package/nodemon

6- React Router

- Description: React Router is a library for routing in React applications, facilitating navigation across pages in the frontend.

- Benefits: It is essential for creating single-page applications that appear as multiple pages, enhancing user experience with smooth transitions and interactions.

- Documentation link: https://reactrouter.com/en/main

Version Control Tools

In complex projects like Senserator version two, where multiple developers are involved in software or web application development, effective source control management is crucial. Source control, also known as version control, is the method of tracking and managing all changes in a software codebase. It facilitates effective collaboration among developers. Several tools can be integrated for version control, such as CVS, SVN, and the most widely used, Git. For our development, we'll be using Git as our source control tool.

- Git:
    - Description: Git is a version control system designed to handle projects of all sizes. It allows developers to work on the same project in different versions without interfering with each other's changes.

    - Benefits of using Git:

- Branching and Merging: Git enables developers to create different branches of the same project, allowing them to develop new features without immediately changing the main codebase. This helps prevent bugs and facilitates the testing process. They can then merge these new features into the main codebase, as illustrated in Figure 25.

- Recovery of Lost Work: Git simplifies the recovery of lost code and work by tracking previous states and changes, which is particularly useful when multiple developers are working on the same project.

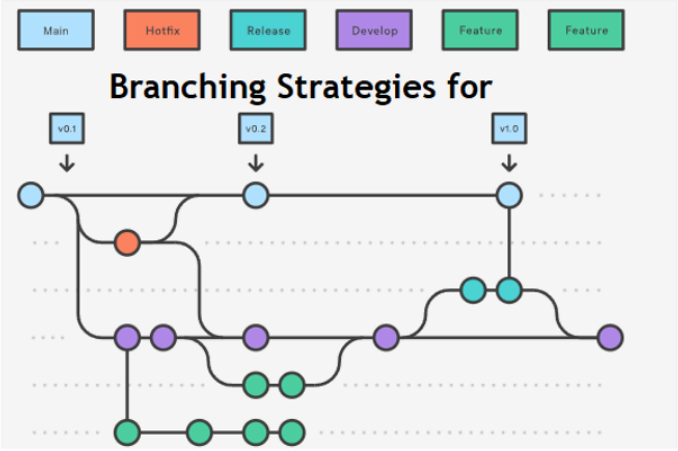

Figure 25: Visualization of Git

· GitHub:

  ○ Description: GitHub is a platform built on top of the Git tool, which helps to manage and visualize the source code online.

  ○ Benefits of using GitHub:

- Collaboration: It enables collaboration with anyone from anywhere.

- Documentation: GitHub can host documentation for various packages and libraries, serving as a repository for content.

- Deployment: GitHub offers numerous CI/CD tools that automate and smooth the testing and deployment processes.

2D Mapping

ThreeJS and Dynamic 2D Mapping

Utility and Flexibility

While ThreeJS is renowned primarily for its 3D rendering capabilities, it has lots of potential utility in creating interactive partial 2D map visualizations that are often overlooked. The framework can be incorporated to enrich our 2D mapping elements of our website with dynamic elements as well as sophisticated interaction layers that can surpass the traditional layouts of most 2D mapping capabilities. With ThreeJS, our team will be able to integrate animated graphics that can reflect our collected data for each region of the map and scoring, overlay interactive markers and objects that offer additional information on a hover or click, and be able to create custom layers to our maps that can represent different aspects of each region (ie. weather, traffic patterns, etc) in engaging ways for the user. This is what

brings so much value to using ThreeJS as a tool in enhancing our site and the way that our data will be shared.

Interactive Features of Map Visualization

Developing interactive map elements involves creating responsive, clickable markers and icons that provide additional insights about specific regions. These features include highlighting scanned regions by area or neighborhood and displaying quality index scores derived from our assessments. Users can interact with these regions in the following ways:

1. **Hovering:** As the user hovers their cursor over a region, a tooltip will appear displaying the quality index score for that area, offering insights into walkability, safety, and cleanliness.
2. **Prolonged Hover:** Holding the cursor over a region for an extended period (e.g., 2 seconds) expands the tooltip to display detailed scoring indicators used in the calculations.
3. **On Click:** Clicking on a region opens options to explore a more detailed 3D map walkthrough, enhancing engagement and data visualization.

Map Visualization Frameworks: Google Maps API vs. Mapbox

Google Maps API

The Google Maps API offers geographic coverage and highly detailed, up-to-date maps. Its features include:

- **Customization:** Developers can control visibility, styling, and tilt of map components such as roads, buildings, and natural features.

- **Overlays:** It supports the addition of real-time traffic conditions, weather data, and population density overlays, providing users with comprehensive contextual information.
- **Marker and Region Management:** Using the API, regions can be defined by coordinates, enabling us to highlight scanned areas and display index scores dynamically.
- **3D Customization:** Google Maps supports customized 3D markers and shaded regions, offering a polished visual representation for users.
-

However, one limitation is the cost structure, which can increase significantly as usage scales, especially for applications requiring high interactivity and frequent updates.

Mapbox

Mapbox emerged as a strong contender due to its flexibility and customization capabilities. It provides:

- **Customization and Style Control:** Mapbox offers unmatched styling options, allowing developers to design maps tailored to specific needs, from aesthetic choices to functional overlays.
- **Performance:** Mapbox handles dynamic and interactive map features efficiently, even for high-traffic applications.
- **Offline Capabilities:** Unlike Google Maps, Mapbox supports offline usage, which could be useful for future applications in less connected areas.
- **Integration with 3D Features:** Similar to Google Maps, Mapbox supports 3D markers, regions, and interactive transitions between 2D and 3D visualizations.

An additional advantage of Mapbox is its open-source nature and cost-effectiveness, which aligns well with the project's budget constraints while maintaining quality and scalability.

After evaluating both platforms, we chose Mapbox as the primary library for our map visualization. The decision was driven by its superior customization capabilities, performance, and cost-effectiveness. With Mapbox, we can:

- Design fully customized maps that align seamlessly with our UI.
- Integrate dynamic features like markers, region boundaries, and overlays for additional information.
- Transition smoothly between 2D and 3D visualizations while maintaining synchronization of data and ensuring accuracy.

User Interface Considerations

- **Intuitive Controls:** UI controls that allow the users to navigate the 2D to 3D maps within the environment
- **Responsive Design:** Map interface responds effectively to different devices and screen sizes.

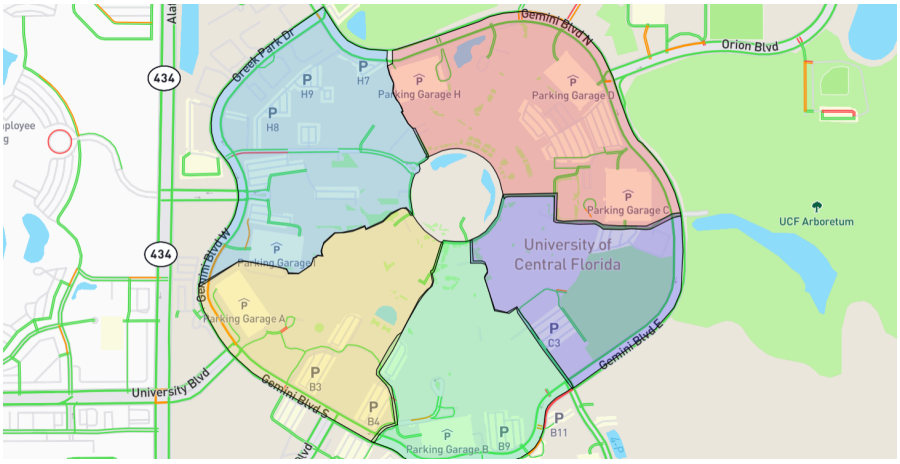

Figure 26: Design of UI for Overview Map (2D)

Block Diagram

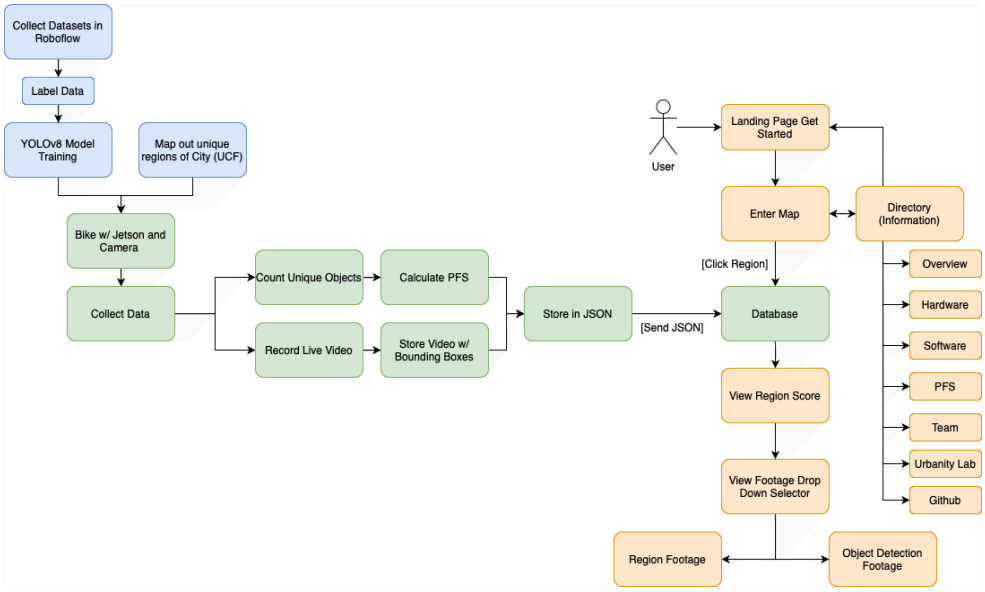

Figure 27: Block Diagram

Activity Diagram

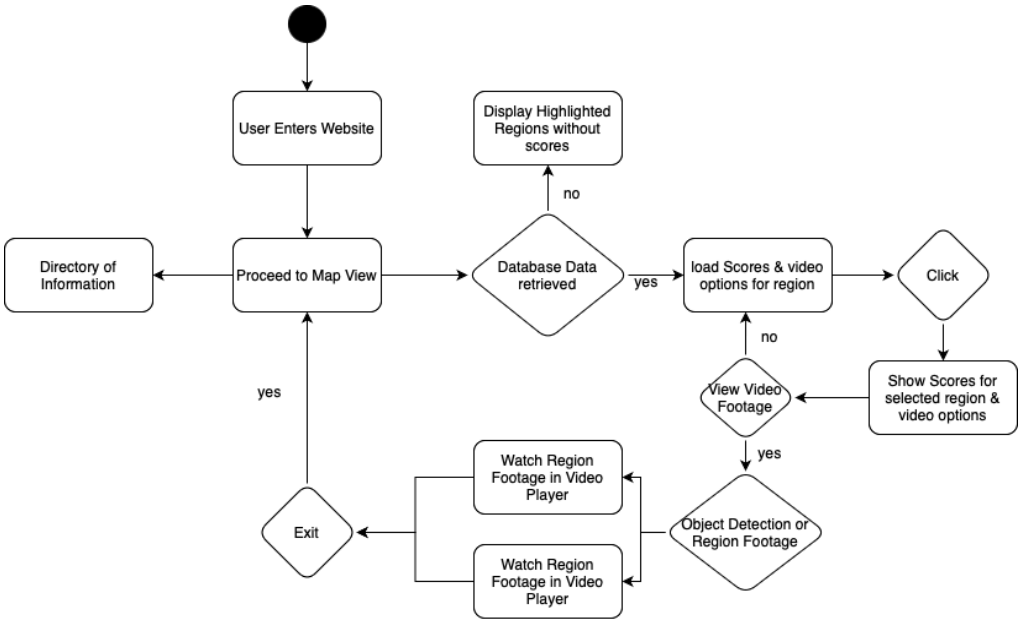

Figure 28: Activity Diagram

Use Case: Diagram

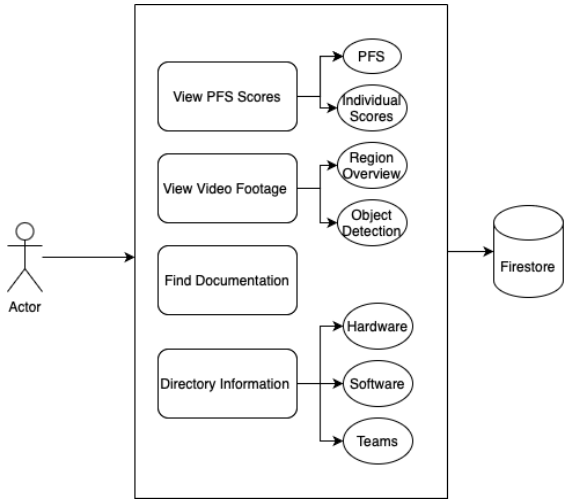

Figure 29: Use Case Diagram

Database: FireBase

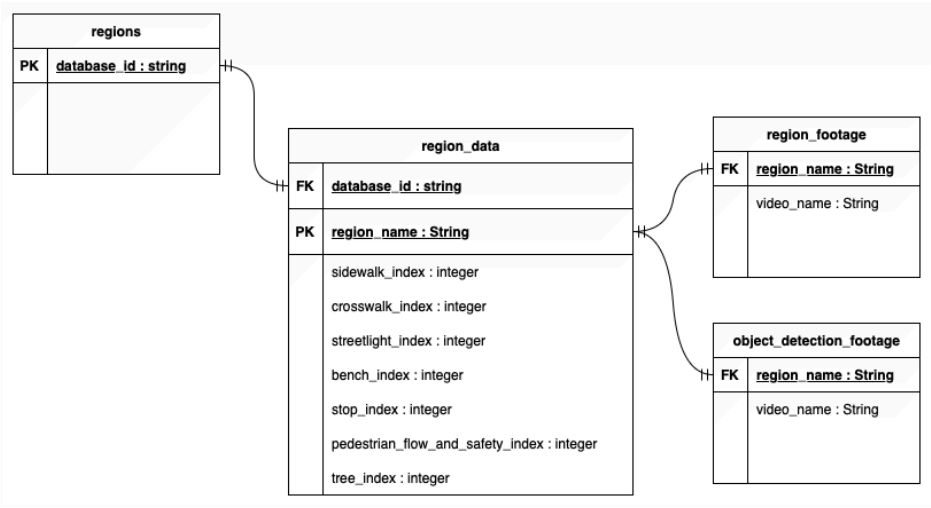

Figure 30: ERD

Conclusions

Project Summary

In conclusion, SenseRator 2.0 is an ambitious project being worked on by a team of 6, dedicated to improving the quality of life and safety in urban communities. This is done by running real time object detection on a vehicle, collecting data in a region, gathering and importing the PFS Score into our database, and displaying the data within the corresponding region on our web application. We have achieved this by finding what features are lacking regarding urban architectures, and what is already good using object detection and analysis. We have a robust web application in which PFS Scores, video footage, and object detection footage can be viewed for each region. It also contains all of the components used in this project.

Characterization of Results

Our product from this project is an interactive and accessible web application which showcases our regions within a map view. Each of the regions can be clicked on to view the PFS Scores, video footage, and an object detection view of the region. This data is available to all users and will raise awareness and open a discussion regarding infrastructure and urban planning that can affect the quality of life in a community.

Lessons Learned

- Object Detection relies heavily on collecting accurate datasets.
- There are many components that contribute and affect real time object detection.

- Hardware quality directly influences data accuracy
- Iterative design enhances hardware functionality
- Hardware integration requires extensive and careful planning along with a design-test-redesign methodology to get the best results

Insights Gained

- Manually gathering datasets will provide the best results.
- Don’t lean too heavily into inference speed or accuracy for object detection in real time.
- Custom solutions can outperform conventional “off-the-shelf” solutions. Don’t be afraid to combine ideas and take other ideas to make them fit your needs.
- User-friendly hardware enhances usability
- Adaptation is the key to hardware development
- Field testing is essential

Future Goals

- Expand features that can be detected and scored.
    - [people : vehicle] ratio
    - proximity to public spaces (i.e. parks, nature areas, etc.)
    - ongoing construction regions
    - quality or width of walking spaces
- add more features to be included in the scoring
- increase computing power of jetson nano for bigger models & faster inferencing

Administration

Team Roles

- Nicholas Karamitos: Hardware, Object Detection, Data Collection
- Allen Lin: Machine Learning, Object Detection, Data Collection
- Abdullah Al Hinaey: Full Stack Web Development, Web App Manager
- Kristian Michel: Pedestrian Flow & Safety Score, Front End Developer
- Alexandra Ramlogan Salgado: Project Manager
- Ayman Arif: Video Compression

Budget

- Hardware: $939.11
    - Jetson Nano: $239.00
    - 5 Inch Monitor: $42.99
    - Sandisk SD Card: $14.15
    - CanaKit Power Supply: $9.99
    - Runcam 2: $99.99
    - Gopro Hero 10 Black: $249.00
    - Logitech M100 Mouse: $15.00
    - DJI OM5 Gimbal: $129.00
    - Noga Israeli Arm: $140.00

- Hosting Bill: $0/Month

Milestones

- January 30th - Discord created and all team members joined
- February 8th - Microsoft Teams created to contact sponsor
- February 9th - Design Proposal and Team Contract Due

- February 15th - Scope Pitches (Sponsor Meeting)
- February 20th - First TA Check-in
- February 22nd - Increased scope pitch on one idea (Sponsor Meeting)
- February 29th - Smart City pitch (Sponsor Meeting)
- March 7th - Updates on data collection, concerns, and index score to (Sponsor Meeting)
- March 14th - Updates on using Unreal Engine, Carla, and Open 3d for mapping (Sponsor Meeting)
- March 25th - Individual Design Documents Due
- March 28th - Discussed capturing 3D space, object detection, and software flow chart (Sponsor Meeting)
- March 28th - Second TA Check-in
- April 4th - Meeting with Dr. Rawat and his suggestions for the project (Sponsor Meeting)
- April 11th - Updates on training with the YOLO model and research into architecture and A.I. object detection (Sponsor Meeting)
- April 18th - Discussed our plans for the summer and current goals for the web app (Sponsor Meeting)
- April 21st - Final Design Document Due
- April 25th - Third TA Check-in
- August 21st - Web Application Prototype With Map
- August 29th - Jetson Nano Arrives
- September 1st - Dataset Collection/Labeling Finishes
- September 3rd- First Trained Model
- September 9th - CDR
- September 11th - Region Data Collection Begins
- September 24th - Web Application Landing Page With Regions
- September 30th - Instructor Check-In
- October 18th - Web Application Scores and Videos Implementation

- October 21st - Instructor Demo
- October 28th - Final Trained Model
- November 1st - Web Application Directories Implementation
- November 6th - Finished Collecting Region Data
- November 10th - Finalized Web Application
- November 16th - Showcase Video
- November 18th - Finished Design Document
- November 19th - Final Committee

Acknowledgements

Descriptions of assistance provided by sponsors and others

Muhammad Shahbaz: Object Detection Guidance
Dr. Shaurya Agarwal: Sponsor/Networking
Dr. Samiul Hasan: Civil Engineering, 'Smart Cities'
Dr. Rick Leinecker: Sponsor of Jetson Nano and other Hardware

Descriptions of unique facilities and equipment

Vehicle: Bicycle
Hardware: Jetson Nano Mount, Camera Stabilizers
Senior Design lab computers: For object detection model training

Bibliography

[1] [2] "Crosswalk-Traffic-Light 2 Computer Vision Dataset by Traffic Light." *Roboflow*, universe.roboflow.com/traffic-light-rpqwy/crosswalk-traffic-light-2/dataset/6/images?split=train.

[3] "Tree Detection by Season / Type Dataset and Pre-Trained Model by Yolo for Tree Detection." *Roboflow*, universe.roboflow.com/yolo-for-tree-detection/tree-detection-by-season-type/images/BkM1vQQHk1UDn0lSX4FW.

[4] Cornille, Tobias. “Panoptic Segmentation: Introduction and Datasets.” *Segments.Ai*, 8 Feb. 2024, segments.ai/blog/panoptic-segmentation-datasets/.

[5] Self-Made

[6] Self-Made

[7] Self-Made

[8] Self-Made

[9] Self-Made

[10] Self-Made

[11] Self-Made

[12] Self-Made

[13] Self-Made

[14] Self-Made

[15] Self-Made

[16] [17] Self-Made

[18] "Grid System in Bootstrap." Scaler, https://www.scaler.com/topics/images/grid-system-in-bootstrap-thumbnail.webp.

[19] "Hierarchy Examples." Website-Files, https://assets-global.website-files.com/646c8b5c2c2d24d335034283/646e1101854ea082bf132333_63bc2cae0087fa02c54927af_Hierarchy%2520examples.png.

[20] "Color Contrast." UCLA Brand Guidelines, https://brand.ucla.edu/fundamentals/accessibility/color-type#:~:text=Color%20Contrast,-Color%20contrast%20is&text=Web%20Content%20Accessibility%20Guidelines%20(WCAG,4.5%3A1%20for%20large%20text.

[21] "Stack Overflow Developer Survey 2023." *Stack Overflow*, survey.stackoverflow.co/2023/#technology. Accessed 18 Nov. 2024.

[22]"Https://Miro.Medium.Com/v2/Resize:Fit:1100/Format:Webp/0*Y4G0mSaguuNEWFI2.Png Do...: Hacker News." *Https://Miro.Medium.Com/v2/Resize:Fit:1100/Format:Webp/0*Y4G0mSaguuNEWFI2.Png Do...* | *Hacker News*, news.ycombinator.com/item?id=36059314. Accessed 18 Nov. 2024.

[23]"Https://Miro.Medium.Com/v2/Resize:Fit:1100/Format:Webp/0*Y4G0mSaguuNEWFI2.Png Do...: Hacker News." *Https://Miro.Medium.Com/v2/Resize:Fit:1100/Format:Webp/0*Y4G0mSagu*

uNEWFI2.Png Do... | *Hacker News*,
news.ycombinator.com/item?id=36059314. Accessed 18 Nov. 2024.

[24] "Platform for Learning Coding & Software Development."
AfterAcademy, afteracademy.com/. Accessed 18 Nov. 2024.

[25]"Https://Miro.Medium.Com/v2/Resize:Fit:1100/Format:Webp/0*Y4G0m
SaguuNEWFI2.Png Do...: Hacker News."
*Https://Miro.Medium.Com/v2/Resize:Fit:1100/Format:Webp/0*Y4G0mSagu
uNEWFI2.Png Do...* | *Hacker News*,
news.ycombinator.com/item?id=36059314. Accessed 18 Nov. 2024.

[26] Self-Made

[27] Self-Made

[28] Self-Made

[29] Self-Made

[30] Self-Made

Appendix/Appendices

Copyright permissions