Final Report:

Autonomous RC Security Car - F1Tenth Car

XX, B.S Computer Science

XX, B.A Computer Science

XX, B.S Computer Science

Angelo Ramos, B.S Computer Science

Advisor: Jeff Caley

CSCI 499, Spring 2024

Abstract

In an era of advancing technology, the integration of autonomous systems in security applications has become paramount. “The Security Car” represents an innovative approach to security and mapping, leveraging fundamental tools for autonomous navigation and data collection. This capstone project presents an autonomous RC car equipped with Lidar sensors for precise environmental mapping and obstacle detection, integrated with ROS (Robot Operating System) for robust control and sensor data management.

As the vehicle navigates its environment, it employs frontier exploration algorithms to identify and mark new frontiers for exploration. Additionally, utilizing a Raspberry Pi-based camera system in conjunction with OpenALPR, the car scans license plates along its path to enhance security monitoring capabilities. This combination of Lidar, ROS, and computer vision technologies ensures efficient and reliable autonomous operation for security and mapping tasks.

Table of Contents

List of Figures

1. Introduction

With an ever increasingly dangerous world, the thought of losing a loved one becomes a looming idea that comes across a lot of families' minds. Nobody wants to lose a loved one. This is much stronger for families of law enforcement. So, when coming up with the idea, the goal is to prevent as much loss of life as possible while still upholding the common duties of a police officer. This was achieved by utilizing technology that has a promising future in the tech sector, robots. The F1 Tenth Car serves as the base of the project, incorporating both software and hardware capabilities to it.

The project explores the use of the F1 Tenth Race Car and a camera that is attached to a Raspberry PI. The F1 Tenth Car has several pieces of technology attached to it to provide the

best possible navigation around the environment. It has a NVIDIA Jetson as the main computer of the robot and LiDAR sensor that detects the surrounding environment. The F1 Tenth robot is usually used for racing but will fit security measures onto it for the ease of use of law enforcement. This project utilizes embedded systems programming and machine learning with license plate scanning by using the OpenALPR software that is open-source.

1.1 Functional Objectives

Before embarking on the specifics of the project, there are a couple of things that the robot must do:

1. Able to view the surrounding environment via a camera
2. Using the LiDAR sensor on the car, it will be able to make a map of the room it is deployed in
3. Create a fluid and adaptive environment for the car to drive on its own

1.2 Educational Objectives

1. Gain additional experience in C++ and Python programming languages that are common in tech.
2. Become familiar with ROS and the systems that go along with making robotics applications.
3. Learn about Ubuntu/Linux and the package manager used in conjunction with ROS
4. Gain experience working with a team towards a common goal
5. Learn how to use OpenALPR to scan and store license plates for later use

2. Requirements

2.1 User Stories

As the project is quite specific, the majority of the intended audience will be law enforcement. As such,

- The car will scan information about a license plate and the relevant environment to see if a security risk is at play
- The car will autonomously drive and map a building using LiDAR so a preview of the room can be accessed before entering

2.2 System Stories

The camera that is attached to the Raspberry PI will be on the robot. It will be scanning around the environment. Once the Raspberry PI detects a license plate, it will scan the plate and determine what the license plate reads. After that, it will store the result with the confidence score into a database for later evaluation. The robot car will also be able to generate a map of the room that it is driving around in using the LiDAR sensor equipped with the car already.

2.3 Hardware Requirements

The main component of the project will be the F1 Tenth RC Car. This will be the platform that has all of the technology strapped on, both hardware and software. The components on the car will be controlled by a pit/host computer that will SSH into both the car via the

NVIDIA Jetson and the Raspberry PI. This will be important for starting the robot software and changing any parameters if necessary.

2.3.1 F1 Tenth Robot Car

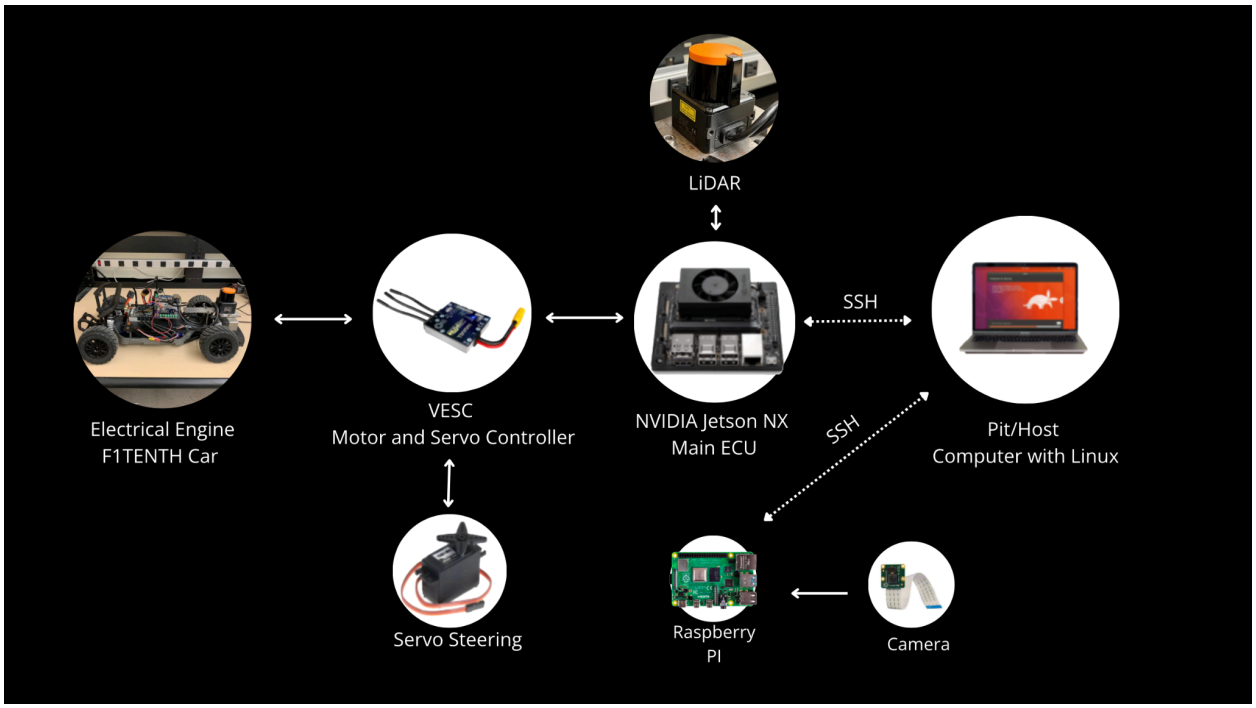

Figure 1: F1 Tenth Car

The F1 Tenth RC Car is built around an NVIDIA Jetson which is the main brain of the car. The car has a LiDAR sensor which can be seen at the top and uses laser scans to calculate the distance to walls and objects that is sent to the Jetson for further use. Then there is the Electronic Speed Controller (VESC) which takes information from the Jetson about velocity and steering angle and translates it to speed the motor functions which control the steering angle for the servo appropriately. The Raspberry Pi and a Camera (Figure 1) are then attached behind the LiDAR sensor to provide an optimal view of license plates causing the least amount of impact on

the LiDAR effectiveness. The Raspberry Pi was picked because it is a cost effective way to provide instructions to the camera while also being small enough to be strapped onto the car.

2.3.2 Raspberry PI Setup

Figure 2: Raspberry PI with camera

2.4 Software Requirements

In order to facilitate the communication between the different components F1 Tenth RC Car, use the GNU operating system, Ubuntu, which is the perfect environment for ROS that will facilitate the majority of the project. For ROS, use both C++ and Python. The majority of the project was written in Python as the majority of the group members were experienced in Python more so than C++. To use the Raspberry PI (figure 2) to scan license plates, it will also house the OpenALPR software. OpenALPR is open source software and free so it helped cut down costs of the project, making it possible to use it for other things in the future.

3. Design

3.1 Structural Design

The final decision was to have the robot detect license plates, drive around on its own and to map out the surroundings. The robot will use a camera that is attached to the car to be constantly scanning the environment for license plates. Once done so, will give a reading on what the license plate is saying and a confidence level about that prediction. Then take those two points of data and store them in a csv final that is stored locally on the car for security purposes. Next, the mapping will be done with LiDAR on the robot to scan the room out while driving. It will be similar to a recon drone, scouting out for information about the room. Finally, the car will drive on its own. This is one core feature of the project that all of the other features around it compliment it.

3.2 ROS Environment

3.2.1 Nodes and Topics

In ROS (Robot Operating System), nodes are fundamental units of computation. Each node represents a distinct process responsible for specific tasks within a robotic system, such as reading sensor data, controlling actuators, processing information, or implementing algorithms. Nodes operate independently and communicate with each other using ROS's messaging infrastructure. This modular design allows developers to create and manage different functionalities as separate nodes, promoting scalability and flexibility in ROS applications.

Nodes can run on a single machine or across multiple computers connected via a network, enabling distributed computing in robotic systems.

Topics in ROS are named communication channels through which nodes exchange messages. They provide a publish-subscribe mechanism that allows nodes to communicate asynchronously. When a node wants to share data with other nodes, it publishes messages to a specific topic. Meanwhile, other nodes interested in receiving this data subscribe to the same topic. Topics can have multiple publishers and subscribers, facilitating flexible and decentralized communication between nodes. Messages sent over topics are defined using ROS message types, which specify the format and structure of data being exchanged.

3.2.2 Publishers and Subscribers

ROS facilitates communication between its packages using publishers and subscribers, which operate through topics. This mechanism allows messages to be exchanged between nodes efficiently. When one node wants to send a message to another, it publishes that message to a designated topic. Meanwhile, the receiving node subscribes to the same topic to retrieve messages published to it.

This system of publishers and subscribers enables a decoupled and modular approach to communication within ROS. Nodes can independently publish or subscribe to topics, allowing for flexibility and scalability in robotic systems. The publisher-subscriber model forms the backbone of ROS's inter-process communication, supporting real-time data exchange and coordination among different components of a robotic system.

3.2.3 Node Design

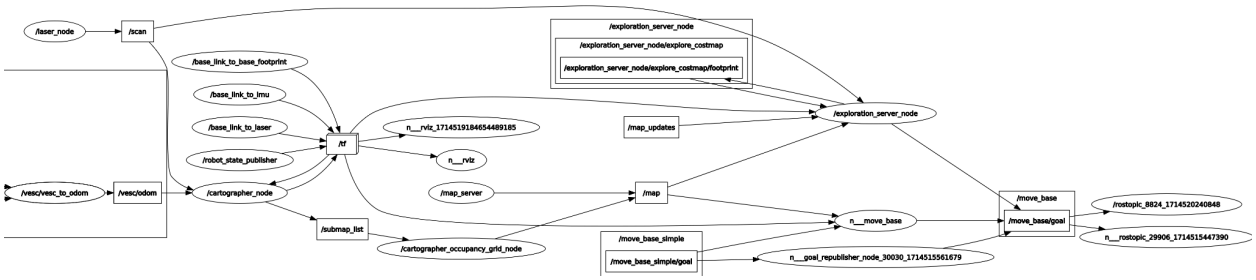

Figure 3: Node Graph

This is the node graph design for the project. To start, the arrows that go out of each box are publishing routes while each box with an arrow attached is subscribing to the box that the arrow came from. To begin with, the far left side of the graph (Figure 3) is responsible for all the things that are relevant to LiDAR, which are used to calculate distances from the robot to the objects in the environment. Moving towards the right, the scan node sends those calculations to Google Cartographer, which generates the maps of rooms based on the information received by LiDAR, which come in LaserScan messages. Following that, it sends the map and takes in the accompanying LaserScan message to frontier exploration where it can determine points in the map that the robot hasn't discovered yet. This is dealt with inside the exploration_server_node near the upper right corner. Once Frontier Exploration has determined the points that the robot hasn't discovered, it publishes that information to the move_base node, which is part of the Navigation Stack package that is mentioned later in the Implementation. This sends the points found in frontier into move goals that pick a direction to go and determines a safe velocity for the robot to follow. Thus, move_base publishes to the VESC which takes care of the speed and steering angle of the robot.

3.3 Wall Follow

The basic setup is that it starts with LiDAR to get the sensor data as a laserscan message. Based on that, calculate the current and the projected error. After that, compute the PID and send the drive command. These are captured in topics, nodes, publishers and subscribers.

3.3.1 Tracking Reference Signal

The main idea is to maintain a certain distance from the wall. The main goals are to maintain a desired speed with minimal delay for steering. With the robot car PID, which controls certain parameters of the car such as the distance from the wall, in conjunction with the VESC, which handles the steering angle and the velocity of the car. This controller converts the current distance to a steering angle. After each update, which happens after a certain amount of time, the controller outputs a new steering angle. This is due because the controller converts the rotation speed of the wheels into the angular velocity. This is what tracking a reference signal entails. With all this happening, there is an inherent error that occurs. This error should be down towards zero.

3.3.2 PID

The proportional, integral and derivative control, or PID, has several different objectives that it wishes to accomplish. The first is to keep the car driving on a centerline. It makes sure that it doesn't throw the car into the wall. It also maintains stability in the driving. The other objective is to be parallel to the wall. This represents the two objectives mathematically as y and ø. So as expected the car to divert from the centerline, reinforce the second objective by doing the following, $L\sin(ø) = 0$.

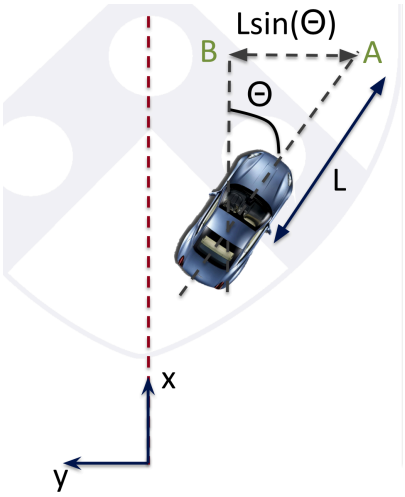

Figure 4: PID Control - Computing Input

The variable L is the distance from the wall in meters (Figure 4). Theta, ø, is the angle between the current driving angle and the centerline. The default L value is 1.5 meters. The goal is to have both y and Lsin(ø) to be zero. The error term is e(t) = -(y + Lsin(ø)). This value should be as close to zero as possible. When the y value is greater than zero, the car is to the left of the centerline. In order to correct this, the ø value has to be less than zero to turn the car to the right. If the scenario was flipped, so the y value is less than zero, then the car is to the right of the centerline and has to be turned to the left. To do this, the theta value has to be greater than zero. To get the optimal steering angle, by using the error term formula from above and add the proportional term as a coefficient. Once adding a scaling constant, C, to the formula, that would be the proportional control. The constant depends on the dimensions, the wheelbase distance of the specific and the cornering stiffness.

The PID control is based on several different parameters that determine different aspects of maintaining the center line. These are proportional, derivative and integral control. For proportional control, if the gain is too low, the action for controlling the car may be too small for correcting the car. It takes care of determining how far from the center line it takes to turn the car

back towards that line. The biggest downfall of the proportional control is it often results in too large of offsets on the centerline of the robot. The car should continuously make adjustments throughout the drive to progressively make the error smaller and smaller. Then make correct adjustments to predict this offset. The simplest way to do this is to add a derivative gain. This is important because the proportional gain follows the error value but not how quickly it changes. The main idea is if the error term is increasing too quickly, The controller should react quickly. This is applied through adding a derivative term onto the proportional gain, which is just an equation similar to the error formula. This controller term nullifies the intensity of the y term. The trajectory correction takes longer for the smoothing out. The integral control is proportional to the sum of the error. The term gets added to the gain formula where it has the integral of the error over a specific time frame. The purpose of the integral control is to address the error creep over the course of the driving from the car. This smooths out the correction that the derivative control offers to the error. The desired steering angle is the culmination of these terms that take in the error. With these terms, they each have coefficients to get a better understanding for the different potential setups. This took some time to do and learn how to do.

3.3.3 Distance Finder

To put proportional, integral and derivative gains into application, the LiDAR will be used to measure the distance from the wall to the car. The LiDAR uses light waves that bounce off objects from the surrounding environment. When they come back, LiDAR calculates the distance traveled based on the time it takes from when it leaves LiDAR to when it comes back. The light rays project out 10 meters in front of the sensor. It is used to continuously align the car to drive along a fixed distance from the wall. In order for the car to get back on the centerline, it

takes one LiDAR ray at 0 degrees and one at some angle ø. Then, with trigonometry calculations, can determine the distances as the rays to the walls.

4. Implementation

4.1 Google Cartographer

Google Cartographer is an advanced simultaneous localization and mapping (SLAM) system designed to create detailed indoor maps using sensor data, particularly from LIDAR scanners and IMU sensors. This system operates within ROS (Robot Operating System) environments commonly used in robotics. Cartographer subscribes to the scan topic, where LIDAR sensor data is published as laser scan messages. These messages contain information about distances to nearby surfaces as detected by laser beams. Cartographer processes this data to extract features like walls, doorways, and obstacles, using probabilistic mapping techniques to estimate the robot's position (localization) and create an accurate map of the environment (mapping). By analyzing changes in laser measurements over time and space, Cartographer identifies and incorporates meaningful features into the evolving map. Additionally, Cartographer employs loop closure detection algorithms to correct localization errors and ensure map consistency. Overall, Cartographer enables robots and autonomous vehicles to navigate indoor spaces autonomously by building and continuously updating precise maps based on real-time sensor input from the scan topic and then publishing that map to the map topic in the map_server.

4.2 Frontier Exploration

The frontier exploration algorithm utilizes the detailed Cartographer map to identify and prioritize unexplored areas, known as frontiers, within the vehicle's operational environment. By referencing this map, which accurately represents the surroundings with precise localization and mapped features, the algorithm can intelligently plan exploration paths. It identifies frontiers—areas adjacent to known free space but not yet explored—and strategically determines navigation goals based on these frontiers that the navigation stack uses to determine where to move to ending in the car moving autonomously.

The integration of Cartographer's map data enhances the effectiveness and efficiency of the frontier exploration process. It allows the security vehicle to navigate and explore with confidence, utilizing up-to-date spatial information to make informed decisions about path planning and exploration priorities. This synergy between the frontier exploration algorithm and the Cartographer map empowers the vehicle to adapt dynamically to its environment, ensuring comprehensive coverage and effective surveillance during security operations.

4.3 Navigation Stack

The navigation stack is how everything ends up working together in a way that the user can see. What the navigation stack does is it subscribes to the goals that are published from the frontier exploration node. It also uses the map built by Google Cartographer from the map topic in the map_server. The transform information from the tf topic allows the navigation stack to transform points between any two coordinate frames at any desired point in time. The Navigation

Stack uses the input information from frontier exploration, Google Cartographer, and transform information into the move base which uses odometry lidar scan data and goal pose to give safe velocity commands to VESC, moving the Car to unexplored areas of its surroundings.

4.4 OpenALPR

OpenALPR, the software seamlessly integrated into the Raspberry Pi system, is designed to efficiently detect and catalog license plate information. Its operation is straightforward yet sophisticated: when presented with an image containing a visible license plate, OpenALPR meticulously analyzes the visual data. It adeptly identifies objects resembling license plates and proceeds to generate ten potential outputs, each accompanied by a confidence score reflecting the software's certainty in its findings. The illustrative example below vividly demonstrates OpenALPR's capability in producing accurate and reliable results.

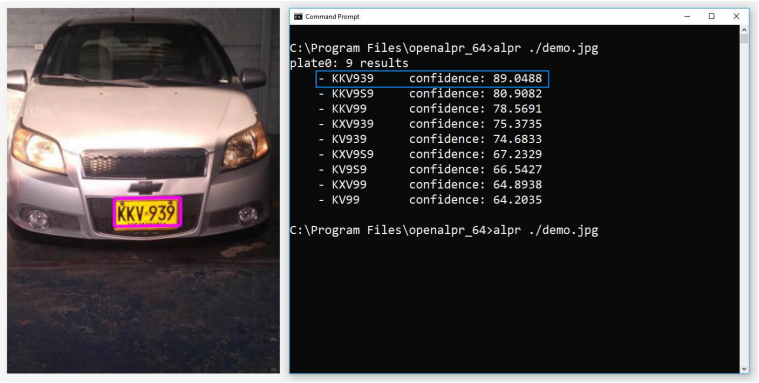

Figure 5: How OpenALPR Works and Output

Utilizing OpenALPR within a Python script, orchestrated a seamless process on the Raspberry Pi system. With the camera configured to capture an image every 5 seconds, the script orchestrates a sophisticated interplay: upon image capture, OpenALPR meticulously scrutinizes the visual data (Figure 5). If a license plate is detected, the script promptly archives the latest image,

alongside the pertinent information gleaned from OpenALPR's analysis. This data is then elegantly cataloged into a CSV file, following a structured format: Date-Time, Plate Number, Confidence, and Image Path—providing a comprehensive record of each entry, complete with its corresponding image location (Figure 6).

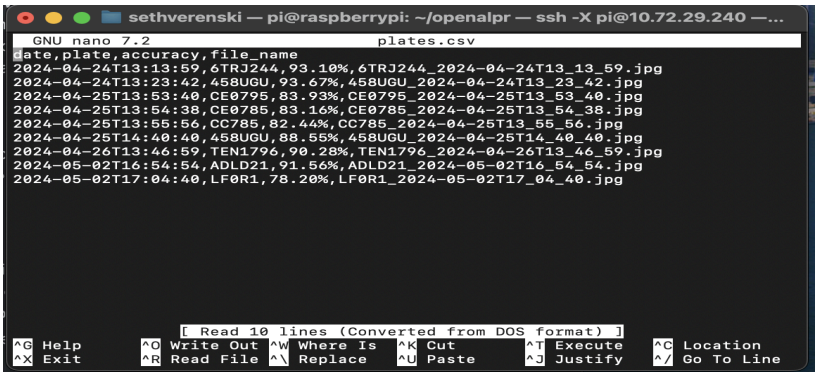

Figure 6: OpenALPR Local Database

4.5 Team Communication

How the team communicated with each other throughout the project was by two methods. One method was by discord which acted as a way to get instant feedback from each other if one member of the team was working and the others were not there. The other method was doing in-person meetings, four times a week Monday through Friday excluding Tuesday. Usually on Thursdays, meet up with Dr. Caley to give a progress report and to ask help if needed.

5. Ethical Discussion

5.1 Ethical Dilemmas

5.1.1 Privacy Concerns

One of the primary ethical dilemmas inherent in the project revolves around privacy concerns, particularly regarding the scanning and storage of license plate information. While the technology aims to enhance security measures, it also introduces the potential for intrusive surveillance. The collection of license plate data, even in public spaces, raises questions about individual privacy rights and the extent to which individuals should be subject to surveillance without their explicit consent. Moreover, the continuous monitoring and recording of license plate information could lead to concerns about unjustified tracking and profiling of individuals, potentially infringing upon their privacy rights.

5.1.2 Data Security

Another ethical dilemma arises in ensuring the security of the collected data. The Autonomous RC Security Car's operation relies on the storage and processing of sensitive information, including license plate data and environmental mapping data. Ensuring robust data security measures is paramount to prevent unauthorized access, data breaches, or misuse of the collected information. Failure to adequately safeguard the data could lead to serious consequences, including privacy violations, identity theft, or misuse of personal information for nefarious purposes. Balancing the need for data collection with the imperative to protect individuals' privacy and security poses a significant ethical challenge.

5.1.3 User Consent

The issue of user consent presents a complex ethical dilemma in the context of the project. While the product aims to enhance security and public safety, obtaining explicit consent from individuals for the collection and use of their data poses challenges, particularly in public spaces where individuals may not be aware of or able to control the surveillance activities. Furthermore, there may be legal and ethical considerations regarding the extent to which individuals should be able to opt out of surveillance activities conducted by autonomous systems like the Security Car. It is essential that the product maintains a balance between the benefits of enhanced security measures and respecting individuals' autonomy and privacy rights which may require careful consideration of the users.

5.2 Accessibility and Inclusivity

Accessibility and inclusivity aren’t major concerns for the project since it is primarily aimed at law enforcement and other government agencies. The general public shouldn't have access to this vehicle for personal use due to potential security concerns. Therefore, in theory, it would only be sold to government agencies and wouldn't require any accessibility settings. Additionally, there isn't much for the user to do, so it is already quite accessible and inclusive. All they really need to do is turn the robot on and ensure that everything is running smoothly.

5.3 Biases in Implementation

There is no bias programmed into this RC car; its sole purpose is to map out buildings and capture images of license plates. There is no bias if it is used appropriately by law enforcement for its intended purpose, such as when law enforcement obtains a warrant to enter a building or compound. Law enforcement should have the legal right to deploy the car for building searches and mapping, provided they have obtained approval from a judge to use this device in accordance with regulations.

5.4 Maintaining Ethical Standards

It's crucial to acknowledge that once the product is in the hands of government officials, particularly law enforcement agencies, maintaining ethical standards becomes partially out of the direct control of the creator of this project. However, to promote ethical use beyond the initial involvement, there must be advocating for ongoing training and education for government personnel on the responsible and ethical use of surveillance technologies. Collaborating with oversight bodies, such as regulatory agencies or independent auditors, can help establish checks and balances to ensure compliance with ethical guidelines and legal regulations.

5.5 Long-Term Effects

With a thorough understanding of ethical considerations by law enforcement agencies, there's potential to create a product that not only motivates but also ensures security for officials. By equipping law enforcement with tools like the Security Car, there's a prospect of reducing risks associated with entering unknown compounds or spaces and being surprised by potential

threats. This proactive approach to security, enabled by autonomous surveillance technologies, could lead to enhanced situational awareness and improved response capabilities, ultimately contributing to the safety and well-being of both officials and the communities they serve.

6. Future Work

In the future, The goal is to seamlessly integrate the navigation stack and connect it with inputs from Frontier Exploration and Google Cartographer to achieve full autonomous control of the car, surpassing basic wall-following scripts. Additionally, The aim is to upload the Raspberry Pi license plate database to a cloud platform to ensure redundancy and accessibility in case of car damage or loss. Furthermore, prioritize user-friendliness by consolidating multiple programs into a single launch file, simplifying the process of getting the robot operational quickly and efficiently in law enforcement scenarios.

7. References

- F1tenth: https://f1tenth-coursekit.readthedocs.io/en/latest/introduction/overview.html
- OpenALPR: https://github.com/openalpr/openalpr
- ROS: https://wiki.ros.org/ROS/Tutorials
- Ubuntu: https://ubuntu.com/
- Frontier Exploration: https://github.com/paulbovbel/frontier_exploration
- Google Cartographer: https://f1tenth-coursekit.readthedocs.io/en/stable/lectures/ModuleC/lecture12.html

8. Glossary

Term	Definition
ROS (Robot Operating System)	Open-Source framework that helps developers and researchers build and reuse code for robotics.
LIDAR (Light Detection and Ranging)	Remote sensing method that uses light in a form of laser to measure how far objects are from the car.
OpenALPR (Open Automatic License Plate Recognition)	An open-source software library designed to recognize and read license plates from images or video streams.
VESC (Vedder Electronic Speed Controller)	Electronic speed controller used for electric vehicles, including autonomous RC-cars
PID (Proportional-integral-Derivative)	Is a mathematical algorithm that is used in robotics to control motion.