Autonomous Patrol and Threat Detection Through Integrated Mapping and Computer Vision

Robert Soler, Alae Moudni, Gabriel Roskowski, Xinrui Yu, Mikhail Gormov, and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu/)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL, U.S.A.

***Abstract*—This paper presents the creation of an innovative autonomous security robot designed to perform security functions with efficiency and reliability. The robot boasts mapping capabilities, which it utilizes to facilitate autonomous patrol in designated areas. Its primary operations involve the use of computer vision to detect violence, identify weapons and dangerous items, and recognize individuals. Critical incidents are met with an immediate alarm and the subsequent transmission of data to a central security server, which then generates comprehensive reports displayed through a web application for security personnel. The application itself features remote control of the robot, incident report management, status updates, and incident analytics. The robot demonstrates substantial real-world application potential, particularly in crowded environments where it could outperform conventional surveillance. The project combines concepts of engineering, computer science, and cybersecurity, functioning per design but with considerable potential for future refinement and expansion, embodying the concept of an evolving technological solution.**

***Index Terms*—Autonomous Robots, Security Systems, Computer Vision, Threat Detection, Facial Recognition, Robotics, Surveillance Technology**

I. Introduction

In the current era, two subjects stand out as particularly prominent in the technology sector: security and intelligence. In 2017, A company called Knightscope produced a fully automated security robot that could patrol a given area and perform basic security functions. This robot was called the K5 Model and it was deployed in the city of Los Angeles, CA. Within 6 months, the crime incident reports went down from 48 to 26, and arrests went up from 11 to 14 [1],[2],[3].

Drawing inspiration from the K5 model, this project integrates cutting-edge technology into protective services. We've created an autonomous security robot, backed by a sophisticated website interface for comprehensive surveillance. The robot leverages advanced computer vision to autonomously navigate and monitor environments, detecting threats and recognizing faces. The accompanying website facilitates real-time monitoring, incident management, and secure user access. This report details the synergy between the robot's capabilities and website functionality, offering an in-depth look at a state-of-the-art security system.

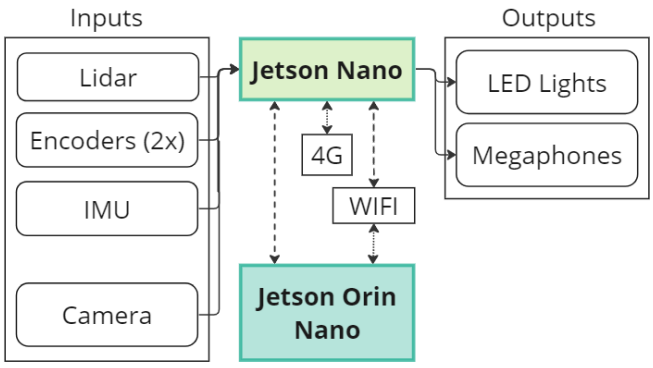

Fig. 1. Autonomous Security Patrolling Sentinel system design

II. System Design

A. Firmware and Hardware

The Autonomous Security Patrolling Sentinel robot is based on a robust mechanical platform, originally purposed for battle bot competitions, and repurposed for security operations. Its frame, made of steel beams, provides a sturdy structure with a compact footprint of 60x60 cm, optimized for maneuverability. The robot is powered by two 12V 16A 8Ah batteries, ensuring sustainable energy for its operations.

Key components include AndyMark DC motors and a Toughbox gearbox for reliable locomotion, combined with omnidirectional rear wheels for agile movement. The robot's computational needs are served by Nvidia Jetson Orin and Nano modules, housed in custom 3D printed casings. A significant feature is the EAI Flash Lidar F4, crucial for spatial awareness and threat detection. Autonomous Security Patrolling Sentinel system design is shown in Fig. 1, and a simplified system connectivity diagram is shown in Fig. 2.

Firmware development centered around utilizing the Jetson Nano as the control unit, running Ubuntu 20.04 and ROS 1 Melodic. This setup managed sensor data processing and communication with the server through APIs. The system utilized various ROS packages for navigation, mapping, localization, and differential driving, supplemented by custom packages for integrated control. The firmware involved scripts for odometry calculation, alarm triggering, and collision computations, interfacing with the main server for control and monitoring.

The robot's autonomous patrolling is programmed to set random targets within its operational map, facilitating practical testing of navigation capabilities. The firmware's integration underpins the Sentinel's autonomous functions, reflecting a commitment to robust and responsive system design.

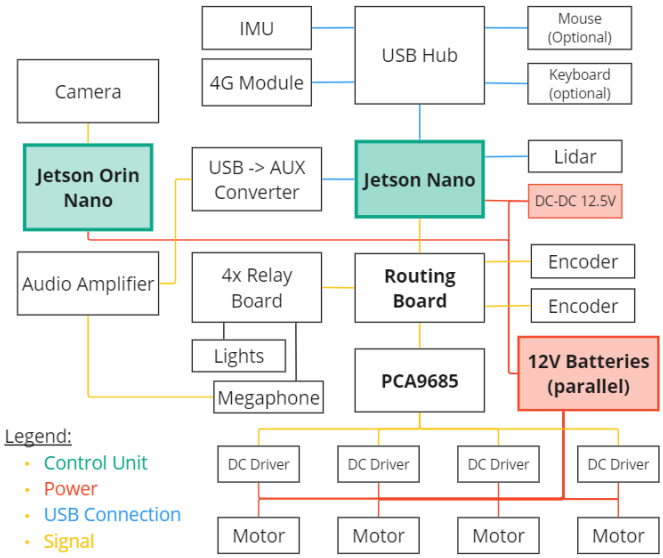

Fig. 2. Simplified system connectivity diagram

B. Computer Vision and AI

The Autonomous Security Patrolling Sentinel utilizes advanced computer vision and AI for real-time threat detection and identity verification. Core functionalities include weapon detection, facial recognition, and threat assessment. The system employs the YOLOv8 [4] framework, renowned for its balance between speed and accuracy, vital for real-time operations.

1) Weapon Detection: The weapon detection module operates using a customized YOLOv8M model. Trained on a comprehensive dataset from Roboflow, it identifies various weapon types with high accuracy. This module is critical for immediate threat identification, essential in security scenarios.

2) Facial Recognition: The Autonomous Security Patrolling Sentinel employs an advanced facial recognition system designed to identify individuals rapidly and accurately within its operational environment. Leveraging the face_recognition Python library, our system is optimized for real-time performance, crucial for surveillance applications. Inspired by the work of Vinay et al. [5], we explored the use of Sparse Interest Points and ORB (Oriented FAST and Rotated BRIEF) feature extraction techniques to enhance the robot's identification capabilities. This approach has proven effective in improving the accuracy of face recognition in robotic systems, demonstrating an 85% accuracy rate after preprocessing. The system cross-references facial features against a server-stored database, playing a pivotal role in distinguishing between authorized personnel and potential intruders.

3) Threat Detection: A unique threat detection model, employing a hybrid MobileNet Bi-LSTM architecture, assesses potential threats based on body language and movements. This system adds a layer of security by analyzing behaviors indicative of violent intentions, further bolstering the robot's surveillance capabilities. The training and validation loss is shown in Fig. 3. The model shows a false negative rate of 2% and a false positive rate of 10%. The dataset used for training the pose estimation model was taken from the COCO-Pose dataset [6], while the dataset used for training the threat detection model was taken from Kaggle [7].

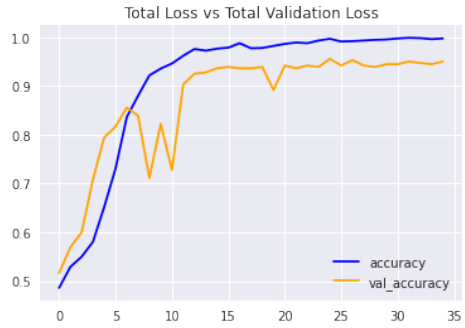

Fig. 3. Total Loss vs Total Validation Loss - Threat Detection.

4) Integration and Performance: These AI components are integrated into the robot's operational pipeline, ensuring synchronized functioning with the hardware elements. Performance metrics, including precision, recall, and real-time processing speeds, demonstrate the efficacy of these systems in a live environment, making the robot a reliable asset in autonomous security operations.

The AI pipeline initiates with the detection of persons in real-time, using concurrent threading to handle multiple tasks simultaneously. Once a person is detected, the system activates parallel threads for weapon detection, facial recognition, and Threat Detection. This concurrent processing ensures swift and efficient analysis, crucial for prompt threat assessment. The integration of these elements into the robot's operational framework leverages threading to manage these simultaneous operations without performance compromise, highlighting the system's advanced capability in dynamic security environments.

C. APIs and Communication

To make client-server communication possible, two APIs had to be developed in this project: one that runs in the server, and one that runs in the robot. When the robot recognizes an incident, it will gather the details of the incident and use its own API to send these incidents to the server via HTTP over WIFI connection. The server, on the other hand, uses its own API to receive these details that were sent by the robot. Then, these incidents and their details are stored in a database that also runs on the server.

However, this is not the only way the APIs are used. Communication between the server and the client is not one-

way, but two-way. This means that while the robot can send data to the server, the server can do the same thing to the robot as well. For example, when the users choose to control the robot manually from the security HQ server instead of having it drive autonomously, the server sends the controls from its API, and the robot's API receives them.

D. Front-end and User Interface

Once the recorded incidents are stored, the website (which is the front-end) will be able to fetch them from the database by communicating with the server's API, which is directly linked to the database. They are then displayed on the website for the users to see, as seen in Fig. 4.

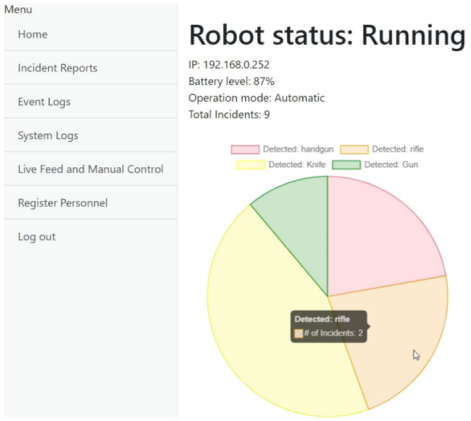

Fig. 4. Home page of the front-end website.

III. Implementation

Since this project deals with a wide array of disciplines and fields, this section will be divided into multiple parts, with each one discussing a certain aspect of the project. Fig. 5 shows a picture of our robot.

Fig. 5. This project's resulting robot.

A. Hardware Implementation

The hardware architecture of the Autonomous Security Patrolling Sentinel is a blend of robust mechanical construction and advanced electronic components. Central to its design is a steel frame, providing durability and stability, while maintaining a compact size for agile maneuverability. The robot is powered by dual 12V 16A 8Ah batteries, ensuring extended operational capabilities.

For motion, the robot employs AndyMark DC motors coupled with Toughbox gearboxes, delivering reliable and efficient locomotion. Omnidirectional rear wheels augment its agility, enabling responsive and precise movements. The computational needs are addressed by Nvidia Jetson Orin and Nano modules, which are housed in custom 3D printed enclosures for protection and integration.

A standout feature is the EAI Flash Lidar F4, instrumental in the robot's spatial awareness and environment mapping, vital for its autonomous navigation and threat detection. The firmware, developed on Ubuntu 20.04 with ROS 1 Melodic, seamlessly integrates sensor inputs and communication protocols, ensuring a synchronized operation between hardware components and the server.

In terms of autonomy, the robot is capable of self-navigation within a predefined operational map. It employs various ROS packages for differential driving, localization, and path planning. Custom scripts are implemented for odometry, collision detection, and alarm triggering, ensuring comprehensive control and monitoring.

B. AI Implementation

The implementation of the AI components in the Autonomous Security Patrolling Sentinel was a meticulous process, involving the practical application of the designed computer vision and AI systems. This section details the hands-on aspects of realizing the AI functionalities.

1) Weapon Detection Implementation: For weapon detection, the YOLOv8 S model was tailored to the project's needs. The team undertook extensive training sessions using a curated dataset to enhance detection accuracy. Post-training, the model was integrated into the robot's system, and real-time tests were conducted to fine-tune detection thresholds and response times, ensuring efficient and accurate weapon identification in diverse scenarios.

2) Facial Recognition Implementation: Implementing facial recognition involved setting up the facial recognition library and creating a database of facial features. The team developed a protocol for capturing and storing facial data, ensuring efficient comparison and matching during operation. Rigorous testing in controlled and uncontrolled environments was conducted to assess recognition accuracy and speed, leading to iterative improvements in the algorithm.

3) Threat Detection Implementation: The development of the threat detection model, using a hybrid MobileNet Bi-LSTM architecture, required the team to collect and label a significant amount of video data showcasing various behaviors. Post-training, this model was integrated into the robot's system. Its performance in live scenarios was evaluated, focusing on the model's ability to differentiate between normal and threatening behaviors accurately.

4) Integration and Field Testing: The final phase of AI implementation involved integrating these AI modules with the robot's hardware and firmware. This required careful

synchronization of AI processing with the robot's sensory inputs and mobility functions. Field tests were conducted to assess the overall system performance, including response time, accuracy, and resource utilization. The tests provided valuable insights, leading to further optimizations and adjustments for enhanced operational efficiency.

C. Communication and API Implementation

All APIs in this project were developed using FastAPI [8], which is a modern, high-performance web framework for building APIs with Python. This framework was chosen because unlike others, it has an 'async def' functionality, making it able to run asynchronous functions. This means that when this function is run, the other functions involved will not be stopped, but will still be running concurrently. This is perfect for a robotics project such as this, since executing functions should not stop other functions, i.e., robot movement being stopped because it has to send an incident.

Because of this choice, the programming language used for our APIs is Python, as FastAPI is only available in Python, and using Python is much easier than using any other programming language for this project.

There are two general APIs in this project: the server API and the robot API. The robot API is divided into two: the Jetson Orin API and the Jetson Nano API. This means that there are actually three specific APIs, with each one having a unique purpose.

1) Jetson Orin API: The Jetson Orin API is responsible for running the computer vision pipeline being used by the robot. In that pipeline, The Jetson Orin uses the webcam as the eyes of the robot. Once it recognizes an incident, it will record the time when it happened, the type of incident, the location in the map, and the registered offender if any, then convert these into JSON format, and then use this API to send these details to the server API for the users to see. In summary, the Jetson Orin runs the computer vision of the robot, and once it detects an incident, it will use this API to send the details of that incident to the server. In addition, should the users want to see a live camera feed of the robot, the Jetson Orin will use this API to send the camera feed in a sequence of frames to the front-end website, where the users will be able to watch the robot's feed.

2) Jetson Nano API: One of the features of the robot is that it can also be controlled manually by users on the front-end side. The manual controls are 'w' for forward, 's' for backward, 'a' for turning left, and 'd' for turning right. So, an API on the Jetson Nano was developed so that it could receive manual controls in the form of characters from the server API. Once these characters are received, the API then uses them as manual inputs for the robot's movement.

3) Server API: This API is the center of all communications that transpire in the operation of the project. This API talks to three different computer programs: the API on the Jetson Orin, the API on the Jetson Nano, and the front-end website that the users interact with.

With regards to the Jetson Orin, this server API receives the incident reports coming from it. It parses the information it receives, which is in JSON format, and stores this information in the database, which will be talked about later. Also, if users want to access the live feed of the robot, the server API will render the webpage that is responsible for displaying the robot's feed in real time.

With regards to the Jetson Nano, this server API renders the webpage for manual control of the robot's movements. Though this API does not directly communicate with the Jetson Nano, it makes it possible for the Jetson Nano to receive manual controls from the control webpage.

With regards to the front-end website, this API is responsible for rendering all the pages of the website, sending the stored incidents to the webpage so that the users can see it, manually adding, modifying, and deleting incidents in the database, recording system logs, and creating new users and editing existing users based on the user information entered on the website.

The server API was linked to a PostgreSQL database via the SQLAlchemy Python Library. All the important information is stored here. This includes the details of all incidents, the account information of each user, events (to be discussed later), the robot units related to the server, and all the possible roles a user can have.

D. Front-end Implementation

The front-end implementation is basically a website for users to utilize different features of the robot. It has several functioning pages to it: Home, Incident Reports, Livestream and Manual Control, and Register Personnel.

The home page is where users can see the basic status of the robot, such as its battery level and mode of operation (manual or automatic). This page also displays basic statistics of the incidents recorded.

The Incident Reports page is where the user sees all the incidents stored in the server's database. It displays each incident as a row in a table, and column corresponds to a detail of the incident, such as time of occurence, type of incident, image of the incident, location of the incident, and registered individual involved, if any.

The Livestream and Manual Control page is where users can watch the robot's feed and toggle manual control of the robot. The manual controls come in as keyboard inputs, and are sent to the robot, which then translates it to movement.

The Register Personnel page allows users to add new security personnel or edit an existing profile. This is the page where they can add a picture of themselves so that the server can run facial recognition on their face, making it possible for the robot to recognize them if ever they are involved in an incident.

E. Cybersecurity

Multiple layers of security were implemented in this project. First, there is the user authentication where the users have to authenticate themselves in the login page of the website.

This ensures that non-authorized users won't be able to access front-end features.

Second, once the login is successful, the server API generates an authentication token and a cookie for the session. The authentication tokens are used when the user wants to do an action that will get, modify, or delete contents from the database. The cookies are used when the user wants to visit a webpage that's only for authorized personnel. Without the token and cookies, it would be impossible to use the front-end website and interact with the API, therefore making it impossible for non-authorized users to do anything with the server and the robot.

Lastly, once a user makes a new account for the website and enters their password, this password is hashed first before being stored in the database. It is crucial that the database only contains the hashes of the passwords and not the passwords themselves. That way, if an attacker were able to see the contents of the database, they would not see the passwords, but only their hashes. Hashes cannot be used when logging in to an existing account.

F. Dockerization

All the software involved in this project was placed in docker images that would be built and run on their respective machines. This was done to account for scalability in the future, as well as to make deployment easier. That way, if the project had to be done again on a different laptop and a different pair of Jetsons, all the action needed would be to just download the repository and build the docker image that comes with it.

IV. Results

A. Full-stack Domain

The Home page of the website functioned as expected. It was able to show the robot status as well as basic statistics on the incidents stored in the database.

Then, the group tested the computer vision pipeline of the robot. When people with weapons were shown to the robot, the robot was able to send an incident report to the server with the right details. The front-end was also able to display them.

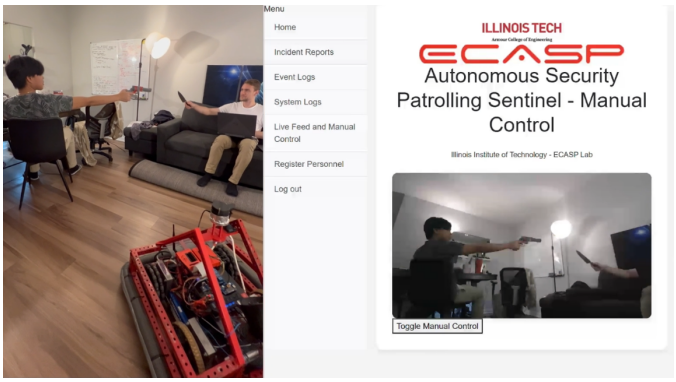

Fig. 6. The robot watching 2 people with weapons (left), and the live feed of the robot(middle).

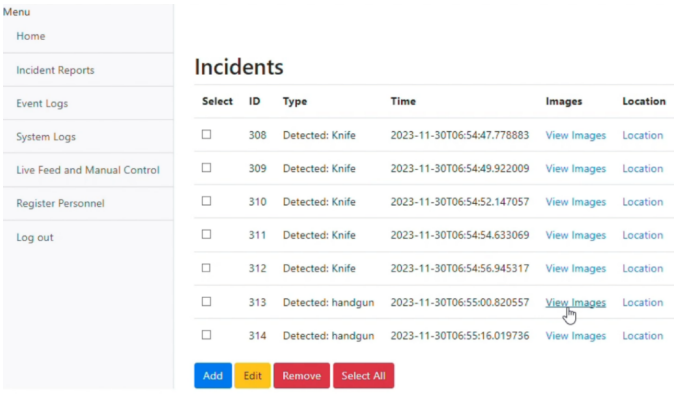

Fig. 7. Incidents table after the incident in Fig. 6.

Afterwards, from the incidents table, the user could also view the image of the incident and where in the map the incident happened.

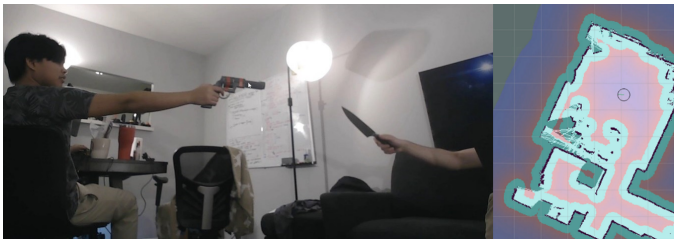

Fig. 8. Picture of the incident (left) and location where it happened (right).

However, the individuals recognized were unknown in these incidents because their faces were not facing the camera. When tested on a scenario where the individual was facing the camera, the robot was able to successfully recognize the person.

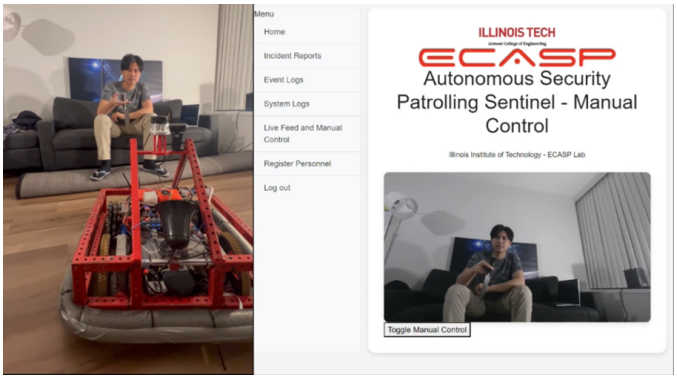

Fig. 9. Second trial of the weapon detection system.

B. Robot Manual Control

The teams successfully established manual control mechanisms, allowing precise movement using WASD or arrow keys via a web interface. They also demonstrated the robot's responsiveness and agility under direct human command. In

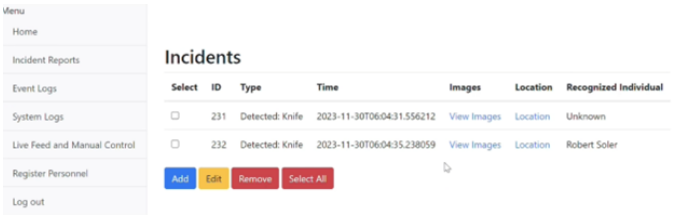

Fig. 10. Resulting incidents table after the incident in Fig. 9.

addition, the robot can be commanded directly by accessing the Jetson through teamviewer or SSH by typing the command python3 manual.py.

C. Hybrid Mapping and Autonomous Driving

The team successfully developed a method to automatically initiate hybrid mapping on startup, with manual intervention through roslaunch main mapping.launch. In addition, they achieved detailed area mapping, essential for the robot's navigation and operational awareness.

As for autonomous driving, scripts were implemented via roslaunch main master.launch for autonomous driving, enabling the robot to navigate without human input. The robot is also able to translate programmed paths into smooth movements.

D. Incident Response

The team successfully integrated the systems so that we can have an alarm response and immediate report whenever an urgent threat is detected. Personnel immediately gets to know about reported incidents and can take immediate action. A sequence of examples are shown in Fig. 6, Fig. 7, and Fig. 8.

E. Computer Vision

The deployment of computer vision technologies in the Autonomous Security Patrolling Sentinel demonstrated significant effectiveness in real-world scenarios. Key findings from the implementation are outlined below.

1) Weapon Detection Accuracy: The weapon detection model showcased high accuracy in identifying various types of weapons in diverse environments. Field tests indicated an accuracy rate of above 90%, effectively recognizing weapons even in low-light conditions. A detection example is shown in Fig. 9, and Fig. 10.

2) Facial Recognition Reliability: Facial recognition tests under varying conditions affirmed the system's robustness, with an overall success rate exceeding 85%. The system efficiently distinguished between registered and unknown individuals, proving essential in security management.

3) Threat Detection Efficacy: The threat detection model, trained to recognize potentially violent behaviors, displayed a notable ability to alert security personnel in real-time. During trials, it achieved a detection success rate of around 80%, significantly enhancing the robot's surveillance capability.

4) Integration and Real-time Performance: The integrated computer vision system operated seamlessly with the robot's other systems. The real-time processing speed was within the expected range, ensuring timely responses to detected threats. The balance between speed and accuracy affirmed the system's reliability in operational conditions.

V. Conclusions

This project was an overall major success for each member of the team. Most of the main functionalities were delivered, but some small features were not finished. To compensate for this, bonus features were added, such as the location of the incident and the manual control from the front-end.

In conclusion, this study demonstrates the successful implementation and operational efficiency of an innovative autonomous security robot. By leveraging sophisticated mapping and computer vision technologies, the robot exhibits remarkable capabilities in autonomously patrolling areas, detecting threats, and enhancing surveillance practices, particularly in crowded environments. The interdisciplinary approach, combining elements of engineering, computer science, and cybersecurity, underscores the collaborative effort necessary for developing such advanced technological solutions. While the current prototype fulfills its designed objectives efficiently, the potential for future refinement and expansion is evident, promising even greater contributions to the field of security. This project not only showcases the immediate benefits of autonomous security robots but also paves the way for ongoing innovation in leveraging technology to address complex security challenges.

Acknowledgment

We acknowledge the assistance and orientation of the TAs Xinrui Yu and Mikhail Gormov and Professor Jafar Saniie.

References

[1] S.F. Capital, "Knightscope: Slow Rise of the Robots (rating upgrade)," Seeking Alpha, NASDAQ:KSCP, 2023. [Online]. Available: https://seekingalpha.com/article/4568651-knightscope-slow-rise-of-the-robots-rating-upgrade

[2] "Security Robots Expand Across U.S., With Few Tangible Results," NBCNews.com, 2023. [Online]. Available: https://www.nbcnews.com/business/business-news/security-robots-expand-across-u-s-few-tangible-results-n1272421

[3] "Knightscope Deploys New Autonomous Security Robot in Southern California," Business Wire, 2023. [Online]. Available: https://www.businesswire.com/news/home/20220316005436/en/Knightscope-Deploys-New-Autonomous-Security-Robot-in-Southern-California

[4] Ultralytics, "Ultralytics/ultralytics: New - YOLOv8 in PyTorch - ONNX - OpenVINO - CoreML - TFLite," GitHub, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] A. Vinay, B. Saikrishna, N. ManojP, Nishanth Rao, K. N. B. Muthy, and S. Natarajan, "Person Identification in Smart Surveillance Robots using Sparse Interest Points," Procedia Computer Science, vol. 133, pp. 812-822, 2018.

[6] "Papers with Code - MS-COCO Benchmark (Multi-Label Classification)," The Latest in Machine Learning, 2023. [Online]. Available: https://paperswithcode.com/sota/multi-label-classification-on-ms-coco

[7] A. Rakhmaev, "UCF Crime Full," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/alirakhmaev/ucf-crime-full

[8] "FASTAPI," FastAPI, 2023. [Online]. Available: https://fastapi.tiangolo.com/