Scientific Research Publishing

Advances in Internet of Things, 2023, 13, 144-169
https://www.scirp.org/journal/ait
ISSN Online: 2161-6825
ISSN Print: 2161-6817

Machine-to-Machine Collaboration Utilizing Internet of Things and Machine Learning

Mohammed Misbahuddin, Abul Kashem Mohammed Azad, Veysel Demir

College of Engineering and Engineering Technology, Northern Illinois University, DeKalb, USA
Email: aazad@niu.edu

How to cite this paper: Misbahuddin, M., Azad, A.K.M. and Demir, V. (2023) Machine-to-Machine Collaboration Utilizing Internet of Things and Machine Learning. *Advances in Internet of Things*, **13**, 144-169.
https://doi.org/10.4236/ait.2023.134008

Received: September 13, 2023
Accepted: October 27, 2023
Published: October 30, 2023

Abstract

Machine-to-Machine (M2M) collaboration opens new opportunities where systems can collaborate without any human intervention and solve engineering problems efficiently and effectively. M2M is widely used for various application areas. Through this reported project authors developed a M2M system where a drone and two ground vehicles collaborate through a base station to implement a system that can be utilized for an indoor search and rescue operation. The model training for drone flight paths achieves almost 100% accuracy. It was also observed that the accuracy of the model increased with more training samples. Both the drone flight path and ground vehicle navigation are controlled from the base station. Machine learning is utilized for modelling of drone's flight path as well as for ground vehicle navigation through obstacles. The developed system was implemented on a field trial within a corridor of a building, and it was demonstrated successfully.

Keywords

Search and Rescue, Image Processing, Navigation Systems, Autonomous Systems, and Object Detection

1. Introduction

With the advent of Cyber-Physical System (CPS) and Internet of Things (IoT) as well as new developments with networking, cloud technologies, and machine learning, machine-to-machine (M2M) collaboration turned into a reality. The concept of M2M collaboration is utilized for search and rescue operation, surveillance, system automation, road transportation, and many more scenarios. IoT provides a network of objects that are embedded with sensors, software, and other technologies with a purpose of connecting and exchanging data with other devices and systems [1] [2] [3].

One of the main ingredients of M2M collaboration is machine learning. There are several initiatives where researchers utilized machine learning for various applications. In one initiative Chirra and his team implemented a behavioral method to detect driver drowsiness through observing facial state [4]. It utilized Viola-Jones detection algorithm to detect the face and extract the eye region from the facial image to determine whether the driver is drowsy or not. A Soft-Max layer in Convolutional Neural Network (CNN) classifier is used to classify the driver as asleep or not. This work reported better accuracy compared with traditional CNN. Munawar and his co-workers presented a classification framework for flood management to group the various technologies reviewed [5]. They found that the lack of hybrid models, which combine image processing and machine learning, for flood management was observed. In addition, the application of machine learning-based methods in the post-disaster scenario was found to be limited. Thus, future efforts need to focus on combining disaster management knowledge, image processing techniques and machine learning tools to ensure effective and holistic disaster management across all phases.

A literature review of the use of machine-learning techniques in the context of human robot collaboration was presented by Semeraro and his team [6]. They have reviewed over 45 papers to perform clustering of works based on the type of collaborative tasks, evaluation metrics and cognitive variables modelled are proposed. Then a deep analysis was carried out on different families of machine learning algorithms and their properties, along with the sensing modalities. Among the observations, it outlined the importance of machine learning algorithms to incorporate time dependencies.

There are number of research initiatives for M2M collaboration includes search and rescue, interconnected machines, energy, and time efficiency, etc. In an initiative Zhou and his co-workers proposed a method to maximize the energy efficiency of M2M transmitters and receivers. Then the joint power control and time allocation problem is solved by combining alternating optimization, nonlinear fractional programming, and linear programming to construct the preference lists [7]. The demonstrated simulation result confirms that the proposed algorithm can approach the optimal performance with low complexity. Another work proposes an energy-efficient resource allocation scheme with hybrid time division multiple access (TDMA)-non-orthogonal multiple access (NOMA) for cellular-enabled M2M networks [8]. The simulation result demonstrates the scheme can dramatically shorten the total transmission time at the cost of a little more energy when compared to the existing works. As reported by Amodu and Othman, M2M provides autonomous collaboration of machines to perform sensing, processing, and actuation activities without human intervention. Some of the applications can be smart grid, electronics systems within a laboratory or industry, servers, etc [9]. They classified M2M system in terms of their application. This includes medium access, home, mobile, standards and service platforms, LTE/LET-A, and energy reliability and security. This paper

reviewed a range of reported systems and tried to provide a mapping of the developments in terms of standard, architecture, security, reliability, and capabilities. In another work Meng and his team provide a reference architecture for collaborative M2M to enhance interoperability between interconnected machines [10]. They also highlighted major technical gaps between M2M communications in IIoT (Industrial IoT).

One of the widely used M2M collaborations is the Amazon warehouses where they process around 1.6 million packages a day with very little human involvement [11]. Most of the Amazon warehouses utilize mobile robots for order processing with very little human intervention [12]. The system utilizes M2M communication extensively and makes it possible for them to be cost effective and responsive to their clients [13] [14]. This makes them secure around 38% or e-commerce market with over 310 million active users with over 600 million listed products. **Figure 1** shows a real-life application of M2M communication in the Amazon warehouses. It shows a cluster of robots that are used by the Amazon warehouse to perform material distribution [15]. These can carry different shelves in the warehouse and move them according to their placement protocol, can fetch parts from their inventory and make them available for shipment. The whole process is automated and mobile robots communicate with each other and map their routes and path accordingly.

There are several reported works where Unmanned Aerial Vehicles (UAV) are used for outdoor search and rescue operation, most of the cases they have used UAV initiated automated for search operation when rescue was performed by human [16]. However, in some cases it is difficult for humans to perform rescue operations due to various reasons. This can be difficult to access to the area, hazardous situation, or lack of trained manpower. Within an indoor facility, search and rescue operation can be more complicated.

To address this issue this article reports on a pilot project for an indoor search and rescue system in the absence of a GPS signal. New developments on Internet of Things (IoT) and potential of machine learning made it possible to implement

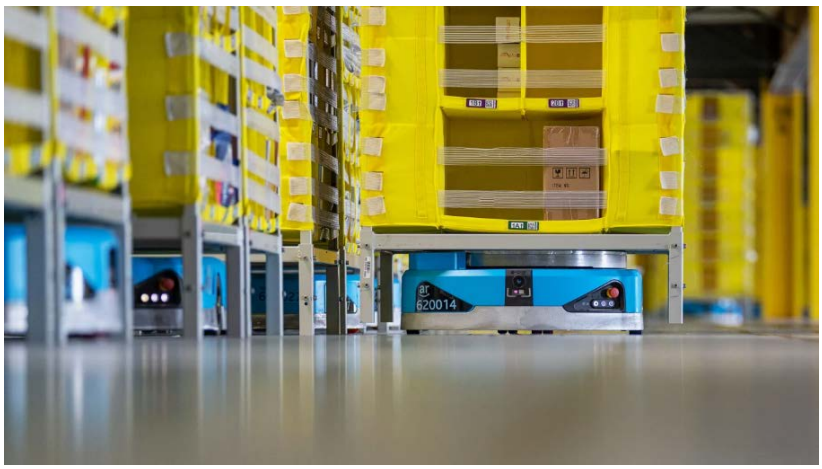

Figure 1. Robots used in an Amazon warehouse.

the M2M collaboration. Within the system a drone and two ground vehicles work in collaboration to identify a specific object and drive the ground vehicles autonomously to the identified object. This paper will explain the methods of coordination between the two vehicles, the sensors and algorithms used, the methods of mapping the indoor environment, navigation in the GPS-denied environment, and obstacle avoidance mechanism. It demonstrates how an M2M system can be designed and controlled within an IoT architecture utilizing machine learning.

The next section describes the system design where it explains the composition of the base station, machine learning model for drone flight and its object identification technique, and finally the ground vehicles and their navigation strategy. The third section describes a graphical user interface that a client can utilize to operate the system. The fourth section presents the data analysis and system evaluation. The next section presents ideas for future growth followed by closing statements.

2. System Design

The paper reports on the design and development of a prototype M2M collaboration system to perform search and rescue operations utilizing a drone and two ground vehicles (Figure 2). The developed system has three major components-base station, drone, and ground vehicles. The operation is managed by the base station that is responsible for most of the computations, communication, and directing drone and ground vehicles. The base station houses different wireless networks to communicate with the drone and ground vehicles and performs machine learning tasks.

The system operation involves the identification of a given object within an indoor environment (by a drone) and dispatching two ground vehicles so that they can initiate a recovery action. To implement this, a drone is tasked to fly over a given area (where the object is supposed to be) and search for the object as specified by the system user. The gathered location information subsequently passed to the base station. The base station calculates the optimum path and relays the navigation route to the ground vehicles that are located at two different locations. The vehicles then travel towards the identified object. In this project a drone is used for search operation and two identical wireless linked cars are used

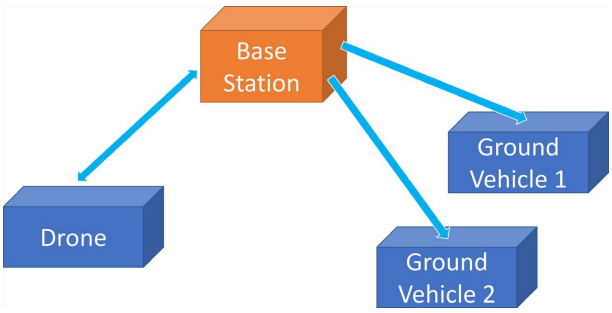

Figure 2. System components.

as ground vehicles.

2.1. Base Station

The base station consists of a NVIDIA Jetson TX2 board as the processor, which is used to run the machine learning for drone and ground vehicles [17]. Jetson TX2 is one of the fastest, powerful, and efficient embedded Artificial Intelligence (AI) computing boards. It's built around an NVIDIA Pascal™-family GPU, with 8 GB of memory and 59.7 GB/s of memory bandwidth. It features a variety of standard hardware interfaces that make it easy to integrate into a wide range of products and form factors. The board includes a 256-core NVIDIA Pascal GPU and ARM 128-bit CPUs, 32 GB eMMC storage as well as Wi-Fi and Bluetooth technology.

The network connectivity for the base station was designed in two different wireless networks, one for the drone and the other for the ground vehicles (**Figure 3**). The first wireless network was used between the base station and drone. The drone comes with a proprietary Wi-Fi access point, which was used to communicate between the base station (NVIDIA) and the drone. This communication channel is used by the drone to send camera images to the base station and receive flying commands from the base station. The second wireless network was used for communication between the base station and the ground vehicles and was implemented with a Wi-Fi router. The router was directly connected to the NVIDIA TX2 board with an ethernet cable. The router created a Wi-Fi access point which was utilized by ground vehicles. The IP addresses for NVIDIA TX2 board and ground vehicles were fixed in the router settings and mapped to their respective MAC addresses. Thereby allowing both the ground vehicles (via Raspberry Pi) to connect with the base station automatically.

The base station houses machine learning algorithms to control the drone as well as directing the ground vehicles. It accepts user inputs for object description and commands the drone to collect images (object) and generate navigation information for ground vehicles. The algorithm is composed of machine learning tools to generate a model for flight path as well as flight direction, object color detection along with optimum path calculation for the ground vehicles.

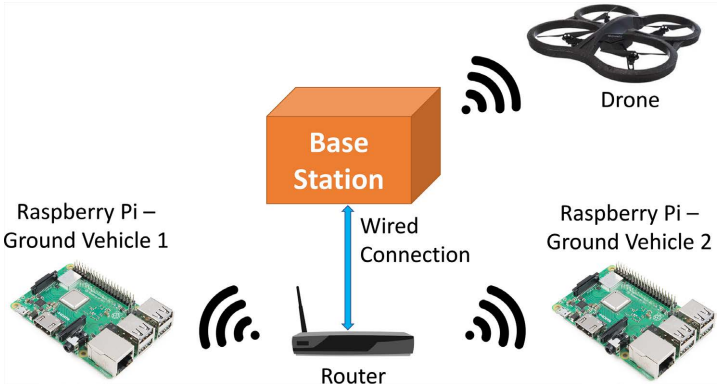

Figure 3. Wireless network connectivity between the base station and machines.

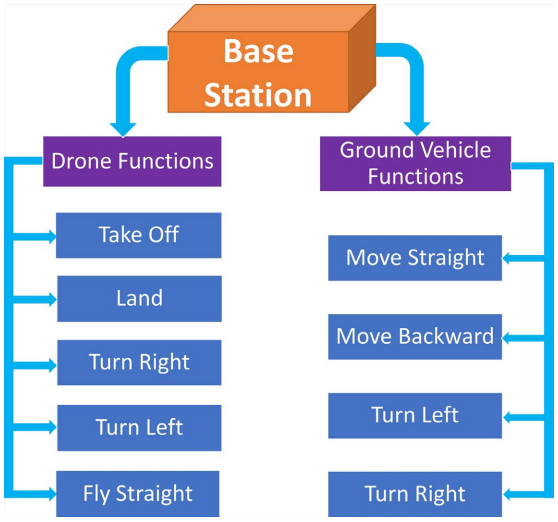

Figure 4. Functions called from the base station.

The base station also hosts the main server, which communicates with each of the machines (drone and ground vehicles) using a Python based Flask server. The Flask server was able to send and receive commands between the base station and machines, which allowed the functions that controlled the machines to be triggered from the base station. The activity plan of the base station is shown in **Figure 4**.

2.2. Autonomous Drone Flight and Object Identification

The drone's task is to fly autonomously within a given space and search for an object as specified for a mission. The drone used for this project is an open system and allows a Python library to program the drone's navigation services. The first part of the activity involves training the drone with flight path so that it can learn about surroundings for a safe flight. With this capability it can then gather data to search for a desired object so that ground vehicles can launch a rescue operation.

2.2.1. Drone Details

The requirements of a drone include open access for programming, dual camera for downward and forward view, light weight, reasonable battery life, and affordability. The open access feature is essential so that it can be integrated with the project without any modification. Most of the commercial drones are not open for user programming access and are expensive. After a rigorous search a Parrot AR Drone was selected for the project (**Table 1**).

The drone can be connected with the base station via a Wi-Fi link and can be controlled up to 50-meter range, which is good enough for the project. The beauty of the system is that it can accept user generated commands and allows access to the camera outputs for further processing. An image of the camera is provided in **Figure 5**.

The drone is made of carbon and nylon material with a weight of 380 gram.

Table 1. Drone details.

Feature	Description
Processor	ARM Cortex A8 1 GHz 32 bits
Video	800 MHz video DSP TMS320DMC64x
RAM	1 GB of DDR1 at 200 MHz
Operating system	Linux 2.6.32
Camera	Two 720p HD with 93-degree field of view and 30 frames per second. H264 encoding and has low latency streaming

Figure 5. Drone used for the project.

The frame is from expanded polypropylene hulls and coated with liquid repellent nano-coating on ultrasound sensors. The system can be operated with smart devices; however, in this project custom applications are developed for its operation. It has a 1000 mAh lithium polymer battery which provides up to 12 minutes of flight time. The battery life is relatively good and suitable for this project. There are four brushless motors at 14.5 W and 28,500 rpm when hovering. It has self-lubricating bronze micro ball bearing and uses 8 MIPs AVR CPU per motor controller. It uses low noise nylatron gears for an 8.6 propeller reducer. The brushless motors are assuredly more reliable and durable than brushed motors. The frames are easily slipped on around the frame of the drone for protection. All these features make this drone suitable for this project which can deliver low noise with required duration of flight with a single charge. More importantly it allows access to camera videos and motor controller for its flight control.

2.2.2. Autonomous Flight Using Machine Learning

During a search mission the drone needs to fly over a designated area within a building. Initially an open corridor of an academic building was selected as a test area where the drone will be flying autonomously. The initial task is to visualize/map the surrounding and obstacles within the drone's operating space and develop a map of the space. The open corridor is 1.5 meters in width and 6.7 meters in height and a length of 50 meters was used for this project. While maintaining a reasonable height, approximately 2 meters, the drone encounters three things: wall to its left, wall to its right, and path which is in front of the drone. Machine learning was used to train the drone to classify these three parameters and maneuver accordingly.

The drone is equipped with two inbuilt cameras, one directed towards the front and the second one directed to the ground. Both the cameras are utilized for this project and provide images to the base station in real time. Initially for training purposes about 8000 images were collected for: the left wall, the right wall, and the front path. These were fed to a machine learning system which was based on a convolutional neural network (CNN) tasked to classify images. The model was trained to a 99% efficiency, and half of the data is used for training and the other half for validation.

TensorFlow was used to develop the image recognition system. This is an open-source machine-learning library from Google [18]. A few steps were utilized to materialize the image recognition process: database development, noise removal, CNN model development, model feature identification, model training, and model testing (**Figure 6**).

Database Development: A key to a quality machine learning system is to have a reliable data set. A model trained on a reliable dataset can surpass any of the sophisticated machine learning algorithms. For this project three types of datasets were collected. These are images of the front path, images of the obstacles on the left side of the path, and images of the obstacles on the right side of the path (**Figure 7**). The idea behind this approach was to develop a model that could identify any of these three types of images and issue an appropriate command to the drone for its future flight path, while the drone is in object search mode.

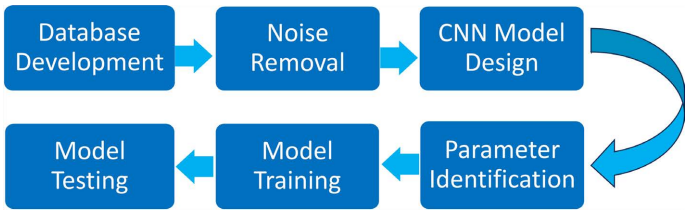

Figure 6. Steps of image recognition system.

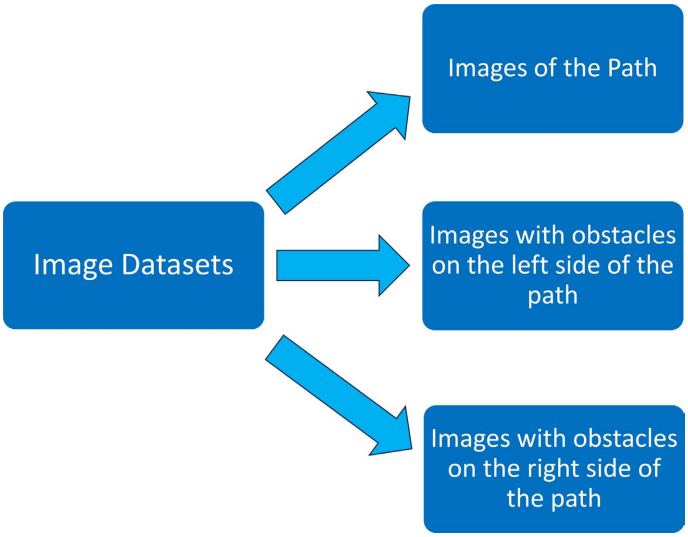

Figure 7. Dataset of images for training.

These were collected through a Python program and OpenCV and saved into the three datasets. The code used is perform this task is shown below:

```
Path_(Image number).jpg for images that showed the path
cv2.imwrite("path/path-" + str(i) + ".jpg", img)
Right_(Image number).jpg for images that showed the obstacles on the right side of the path
cv2.imwrite("right/right-" + str(i) + ".jpg", img)
Left_(Image number).jpg for images that showed the obstacles on the left side of the path
cv2.imwrite("left/left-" + str(i) + ".jpg", img)
```

It was found that when the images were saved using a progressive numbering system, the model trained would give about 67% accuracy on the test dataset. The code for the progressive numbering system looked like below:

```
i = i +1
cv2.imwrite("path/path-" + str(i) + ".jpg", img)
```

Here the letter "i" is holding the number of the image. By doing "i = i + 1" it is progressing the images as they come. However, when nomenclature changed from progressive to randomized numbering system, the accuracy of the system soared to 99%. Here instead of doing "i = i + 1" we would use a rand() function from Python and generate randomized numbers while saving the images.

Modeling Errors: There were two modeling errors that were encountered, improper images and incomplete images. This issue came forward during the initial phase of training. The training data (images) collection was performed manually while orienting the drone camera in different directions. It required facing the camera towards the flight path and collecting these images. To get a correct image it was required to have the same orientation (each time) of the camera in three different directions. However, there were times when the camera was not properly oriented and generated an incorrect image. This introduced a huge shift in the model accuracy. Incomplete images on the other hand are caused by the software. When the amount of time required to save a new image exceeds the capturing rate, it ends up with incomplete images. Both the incorrect images as well as incomplete images should have to be removed for a quality model.

CNN Model Design: Convolution Neural Network (CNN) is a type of deep learning network architecture that uses a layered neural network to understand the data. Model development process view images as pixels as array of matrices. These matrices are fed to the CNN network whose main task is to identify features from the images.

There were 4 stages of convolution layers, each of which were responsible for generating the feature matrix. A learning rate of about 1.5 with 50 epochs was utilized to make sure there was the best accuracy. The ReLU, Softmax, tanH, and

Sigmoid functions are some of the most often utilized activation functions. All of these functions have distinct uses [19]. For a 2-class CNN model, sigmoid and softmax functions are favored, whereas softmax is typically employed for multi-class classification. In this project the activation function deployed was the sigmoid function while using Adam optimizer (**Figure 8**).

The Sigmoid function is a very good tool to derive the outcome of a training model [20]. It can provide a Yes or No value based on the features calculated.

Model Training and Model Validation: Once the model was developed, the system trained for image recognition utilizing about 8000 images. These images were obtained by hovering the drone in different directions and orientations that it was expected to fly over the testing zone. During the process two major issues were encountered, changes in surrounding variables and errors in initial image selection.

Surrounding variables include the lighting conditions and stationary objects in the background, whose position can change with time. **Figure 9** shows one of the walls of the corridor which was used for collecting training images. One side of this corridor has a glass wall that allows sunlight to enter the building. The lighting level in this area is very much influenced by the sunlight, especially during the morning and afternoon times.

As shown in **Figure 10**, the stationary objects are tables and chairs stationed at various locations in the corridor. While these objects were stationary (not

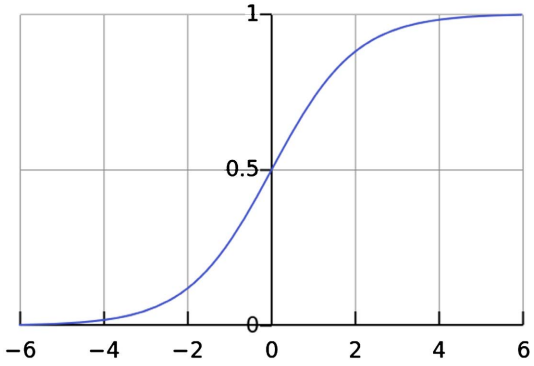

Figure 8. Sigmoid function graph.

Figure 9. Corridor with glass wall on one side.

permanent), they were movable, and their location can change with time. During the model training phase there was no stationary object present in the background. The image recognition process was hampered during the deployment phase, when using the model in the presence of stationary object in the area. Any changes in the orientation and position of these objects between the training phase and deployment phase would cause issues in the image recognition process.

The other major issue for the training process is the improper image processing and image selection. As mentioned earlier, the training images were collected manually. It is possible to miss key features from the images during processing and filtering that can produce an improperly trained model. The image in Figure 11 shows an image of the corridor which is fed to the training model to classify it as a path. Which means, the drone is expected to fly in straight when it encounters an image of this kind. However, Figure 12, which is also oriented in the direction of the path but is facing upward. This causes the drone to fly upwards when it is facing upwards. This will be an incorrect instruction for the drone.

To mitigate the surrounding variables issue (lighting level and movable stationary objects) the testing location is moved to a different location. In addition, a better image screening process was utilized in selecting the images that were

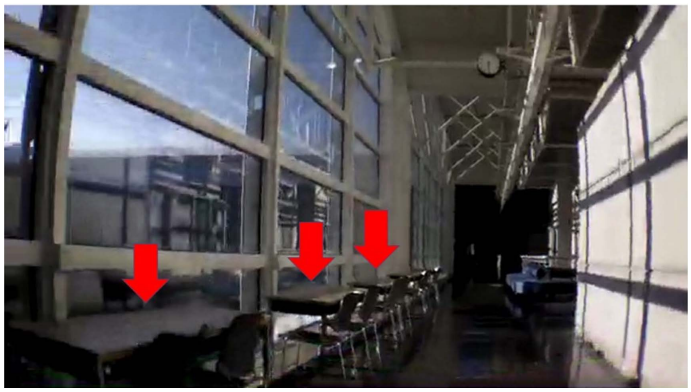

Figure 10. Stationary objects in the training images.

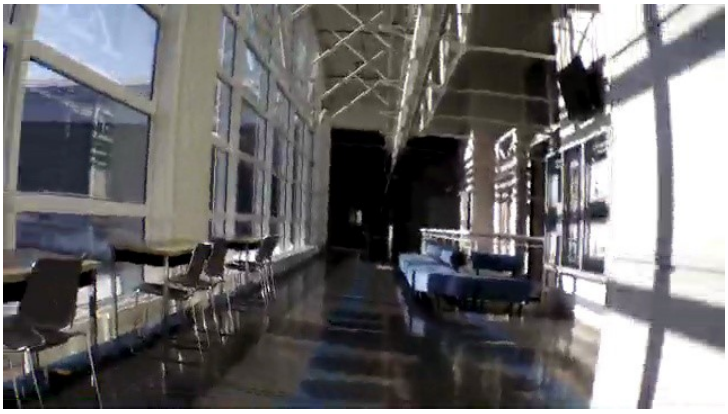

Figure 11. Image of a path for the drone to fly in the corridor.

fed to the training system. Both these steps cleared the issues and provided 99% accuracy in the testing dataset.

Figures 13-15 present a number of images of the flying zone showing the

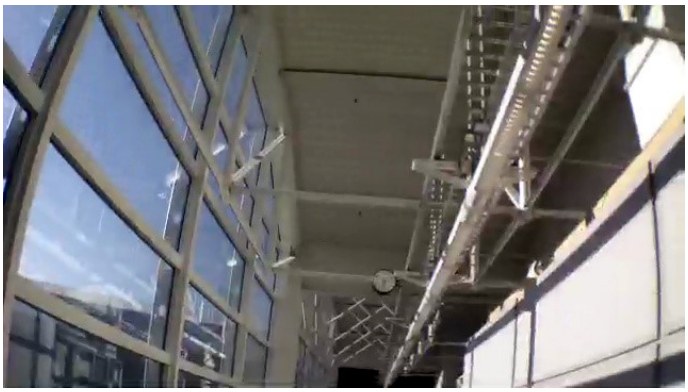

Figure 12. Incorrect image fed as a path image for training.

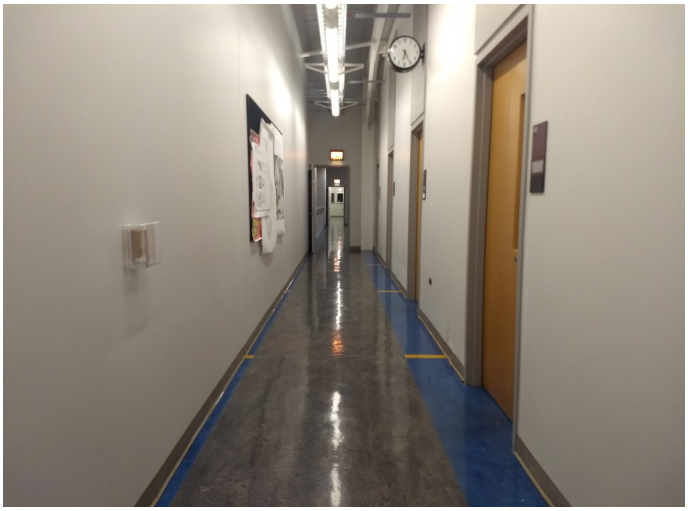

Figure 13. Image of the path for the drone.

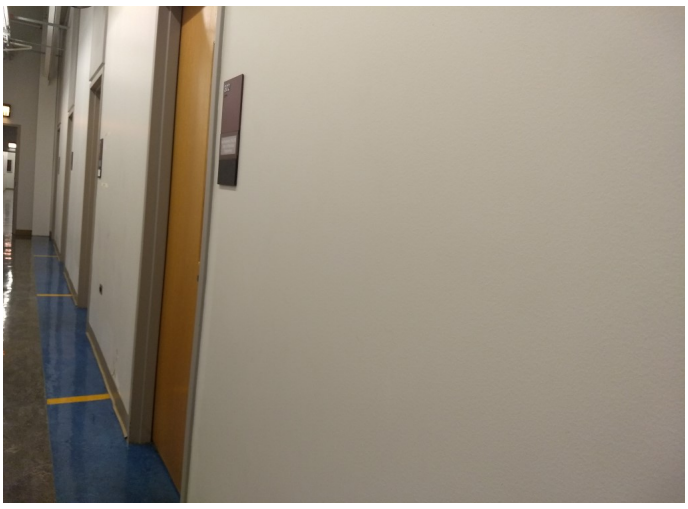

Figure 14. Image of the right side of the corridor.

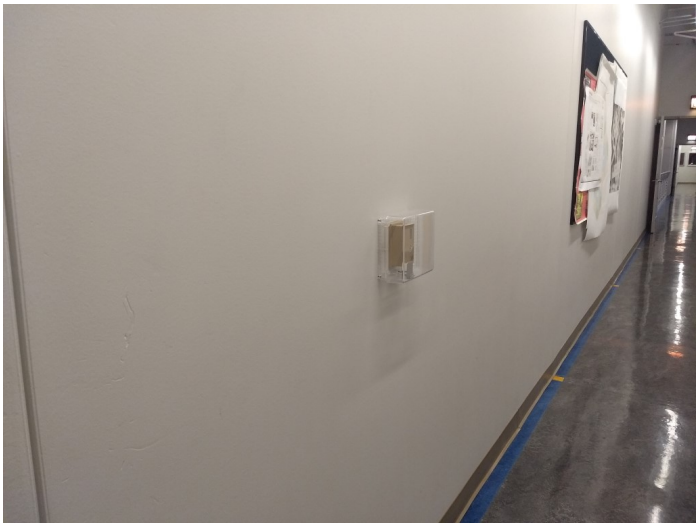

Figure 15. Image of the left side of the corridor.

path, obstacles encountered on the right and obstacles encountered on the left side of the path.

Once the developed CNN model is trained with the environment the drone is now ready for autonomous flight. When the drone is activated and initiates a flight, it starts sending images to the base station where the machine learning algorithm was running. The machine learning program then computes the images received and classifies it into one of three categories: left wall, right wall, and path. When it encounters a left wall, it sends a command to turn right. Similarly, when it encounters a right wall it sends a command to turn left. Lastly, when it encounters a front path, it commands it to move forward.

2.2.3. Object Search with Machine Learning and Image Processing

A trained model based on the ImageNet database was utilized for this purpose [21]. Object detection mainly comprises of two tasks, image classification and object localization. The first task involves categorizing the pixels in an image into one or more classes, followed by object localization to obtain a bounding box that describes the location of the object in a given image. Here, R-CNN was used to perform object detection. A R-CNN is an extension of CNN with a focus on object detection. A R-CNN is a Region-based CNN. It is a visual object detection system that combines bottom-up region proposals with rich features computed by a convolutional neural network. R-CNN proposes a bunch of boxes in the image and sees if any of them correspond to an object. It computes these proposal regions with a selective search algorithm. R-CNN achieves this task using a three-step process [22]. In the first step it creates regions in each image using selective searching [23]. Selective searching uses features such as color, location, scale etc. to selectively create regions that may correspond to an object in the image. These regions are then sent to a feature extraction stage (second step of R-CNN) that identifies what the region corresponds to and labels it. The authors of R-CNN used AlexNet that uses deep convolutional neural network to extract

relevant features in the regions [24]. These features are then sent to a linear Support Vector Mechanism (SVM) that is trained for each SVM class. Unfortunately, this is a very time-consuming process as each of the regions are sent to the linear SVM, which then goes through each of the SVM classes to perform object detection.

However, flying a drone and performing object recognition requires running two separate machine learning programs at the same time. It overloaded the processor at the base station and introduced latency. There was an additional delay in image transmission to the base station and issuing flying command to the drone. The extended latency fails the system to achieve real time performance.

To address this issue, in this project color detection method was utilized to identify the object, where color of the object was the primary goal for the system to detect. This approach requires less processing time than object detection. With this arrangement images from the drone undergo histogram splitting, thresholding and morphological operations to detect a colored object in its space. Initially the image is converted from a RGB space to Hue Saturation Variance (HSV), which allows the system to easily detect colors easily [25]. The image is then sent through a filter that allows only color of certain Hue value to go through, for example blue color has a Hue of 240 (hence allowing only blue to pass while rejecting every other color). The image is then sent through a thresholding filter to convert into binary, which allows the images to undergo erosion and dilation on the images to remove unwanted noise from the image [26]. The images are then sent to morphological operations to detect bounding box of the colored object, which gives the location of the object in the image.

All this processing was done at the base station. Once the color is detected, the drone is commanded to move towards the object and provide location information to the base station. Considering the object location information, base station then calculates the navigation plan for the ground vehicles.

2.3. Ground Vehicles and Navigation

Two ground vehicles were used for this experiment, which are running with Raspberry Pi modules for communication and control. The ground vehicles task is to move towards the target while avoiding obstacles on the ground. These obstacles information is programmed into the base station and is used for navigation algorithm to calculate an optimum path for the ground vehicles. Commands from the base station direct the ground vehicles towards the target while avoiding obstacles on the ground. There were various path planning (navigation) algorithms that were tested and investigated to assist the ground vehicles while maneuvering towards the target [27]. The algorithms investigated for this project were Breadth first search, Depth first search, Dijkstra Search and A-Star algorithm.

2.3.1. Vehicle Design

Both ground vehicles are identical and are developed from RC car chassis kit

which includes speed encoder, two drive wheel and one caster wheel, a motor driver, battery box along with a Raspberry Pi board (Figure 16). The car has a dimension of 21 cm × 10 cm (L × W) with a wheel size of 6 cm × 2.7 cm (D × H). The motor requires between 3 V and 6 V and driven by an L298 motor drive controller board with a dual H-bridge. The driver board has a strong driving ability with low power consumption and anti-interference ability. The board is lightweight, approximately 3.5 ounces. A Raspberry Pi 3 module is mounted on each ground vehicle to provide a wireless connection between the base station and the vehicle as well as generate motion command for the motors via the motor driver.

2.3.2. Path Planning and Navigation

Once the target has been identified, the next task is to guide the ground vehicles to the target through obstacles. Several small blocks are placed on the path between the ground vehicles and the target (Figure 17). To navigate through the obstacles without a collision path planning is an important factor.

Ma presented a graph-based path planning strategy to design collision-free paths for multiple robots for a real-world multi-robot system and has studied an optimization problem on graphs, called multi-agent path finding (MAPF). This review surveys different categories of classic and state-of-the-art MAPF algorithms and different research attempts to tackle the challenges of generalizing MAPF techniques to real-world scenarios [28]. Algorithmic techniques for

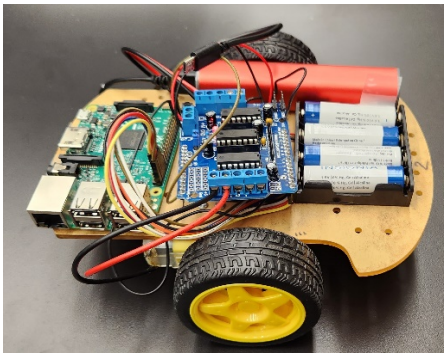

Figure 16. A picture of the ground vehicle.

Figure 17. Experimental set (red arrow pointing the target).

MAPF problems have addressed important aspects of several multi-robot applications, including automated warehouse fulfillment and sortation, automated train scheduling, and navigation of non-holonomic robots and quadcopters. This showcases their potential for real-world applications of large-scale multi-robot systems.

A novel path planning methodology based on the directional graph search and the geometry curve for the Automated Valet Parking (AVP) system was presented by Qin and his team members [29]. The whole path planning methodology is divided into three parts including the global path planning, the path coordination strategy, and the parking path planning. Simulation results show that it takes less time for the proposed path planning algorithm to generate a feasible path for the AVP system compared with the general planning algorithm.

In this project we have utilized the graph-based path planning approach that is widely used for robotic applications [30]. There are several approaches for graph-based path planning. Significant ones are Breadth first search, Depth first search, Dijkstra Search and A-Start algorithm. This section will highlight their relative performance as well as describe the final choice of algorithm. Breadth First Search and Depth First Search can do a good job in terms of searching the node. However, one should look out for all the nodes in the tree before and there is unnecessary expense of energy in finding the target object. In the case of Dijkstra's algorithm, it is good at identifying the optimum path for finding the target object. However, it was found out that it wasn't the efficient way to find a path.

The A-Star path finding algorithm is an advanced version of the Dijkstra's algorithm and is the most efficient compared to other methods [31]. This uses another parameter along with the parameters and techniques used by Dijkstra's algorithm. A term called a Heuristic variable is used in reference to decide the node to use.

Unlike Dijkstra, A-Star is not only considering the distance of the nodes from the goal node (**Figure 18**). This also accounts the actual distance from the node to goal, considering there was a direct path from the node to the goal. During the path planning the locations of the obstacles are known to the base station. However, the object's location is unknown to the base station until it was detected by the drone. Based on the location of the object and the obstacles, the base station computes the path from the location of each of the ground vehicles towards the object. The base station runs the A-Star path finding algorithm from the location of each ground vehicle towards the object's location.

3. Graphical User Interface

The developed Graphical User Interface (GUI) systems run over an online Flask Server. Flask is a Python based micro web framework which allows running python scripts on a server side. It allows calling of Python based functions directly from web browser. For example, if it is needed to call a particular function,

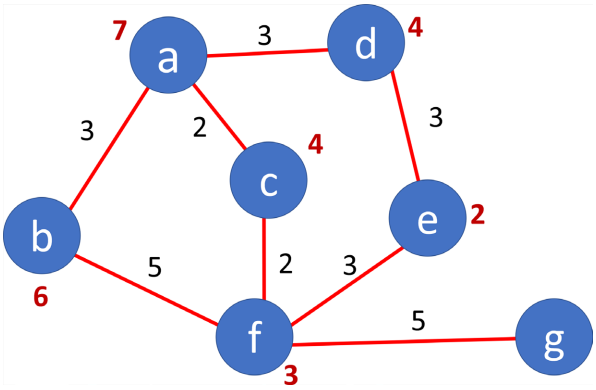

Figure 18. A-Star algorithm structure.

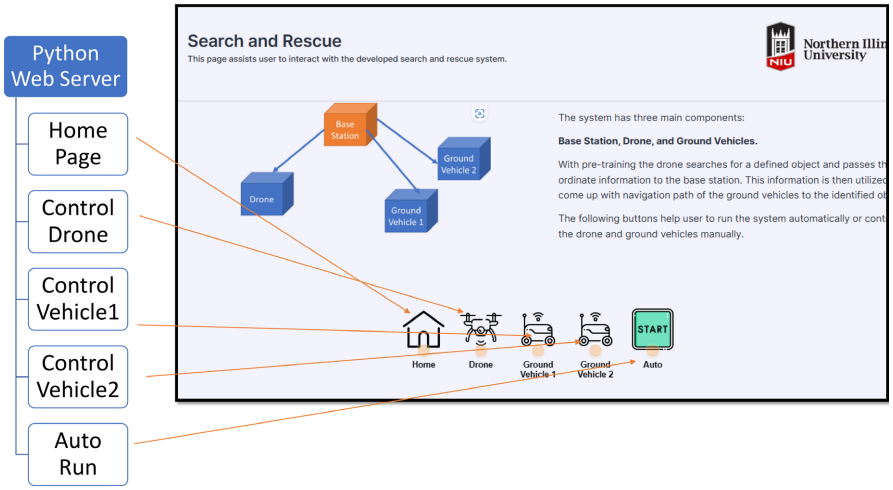

Figure 19. Python program and flask server relation.

which was responsible to move the ground vehicle forward. By invoking a particular web address, one can specifically call that function. Figure 19 shows how a Flask Server can have different compartments for accessing the Python program in case of ground vehicle control. It shows how the buttons on the page are placed that can access each of the machines in the network and perform operations.

A simple code with a Flask framework looks like below:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello World!"
```

Where @app.route() allows a Flask server to access the Python program from a web browser. The argument of this function takes the address of the program. One can have an HTML page with various buttons placed on it. These buttons can then direct to the individual codes on the Python Script. For example, a function that is responsible for moving the vehicle forward looks like:

```
from AMSpi import AMSpi
From flask import Flask
app = Flask(__name__)
# For BOARD pin numbering use AMSpi(True) otherwise BCM is used
@app.route('/forward'):
def forward():
with AMSpi() as amspi:
    # Set PINs for controlling shift register (GPIO numbering)
amspi.set_74HC595_pins(21, 20, 16)
    # Set PINs for controlling all 4 motors (GPIO numbering)
    amspi.set_L293D_pins(5, 6, 13, 19)
    # Run motors amspi.run_dc_motors([amspi.DC_Motor_1, amspi.DC_Motor_2,
    amspi.DC_Motor_3, amspi.DC_Motor_4])
```

A similar strategy was used to design all the GUIs developed for this project. Figure 20 shows a breakdown of the homepage GUI. The home page is where all the controls are housed. This can access individual machines separately, like the drone and ground vehicles. Within each of these machines, they have their own micro servers that access their individual controls.

Figure 21 and Figure 22 show the GUIs for drone and ground vehicle control.

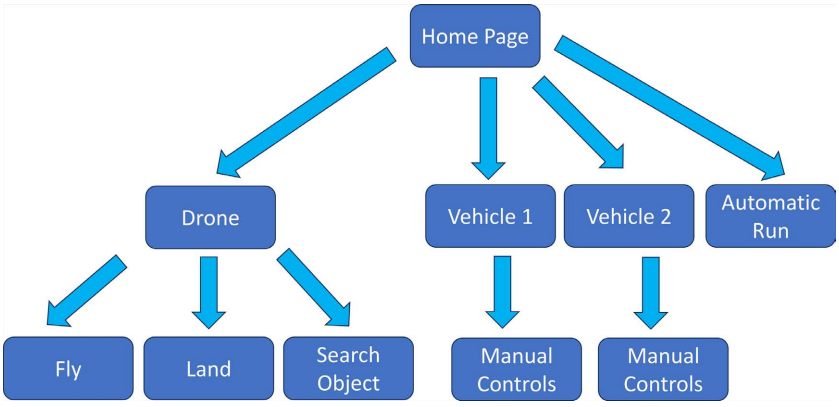

Figure 20. Web page flow layout.

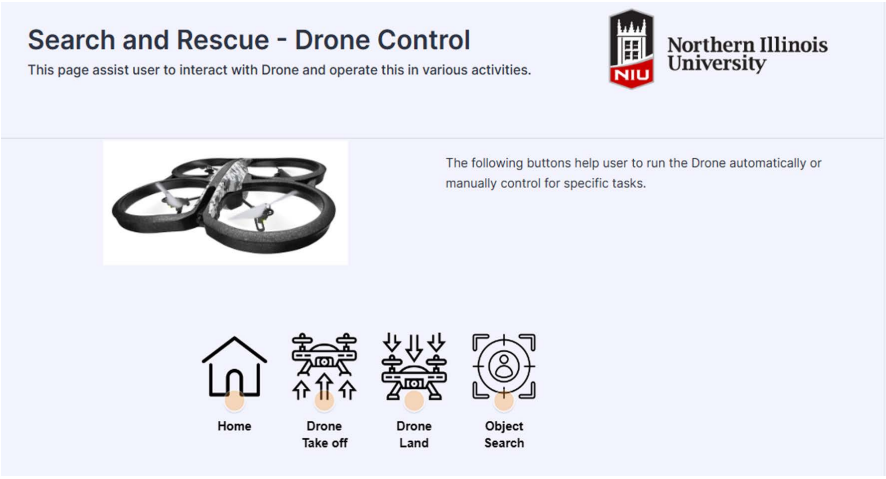

Figure 21. GUI to control the drone.

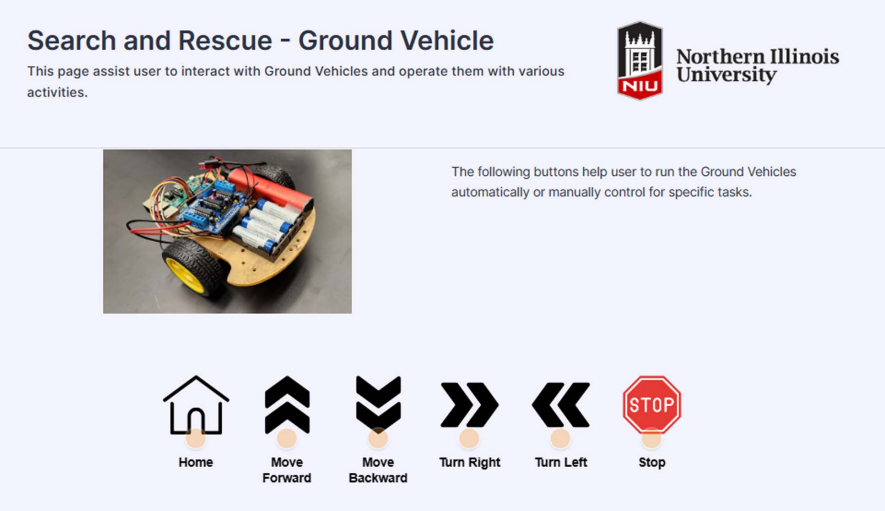

Figure 22. GUI to control the ground vehicles.

4. Data Analysis and Evaluation

Based on the system components the data analysis was performed into three segments. These are the drone, the base station, and the ground vehicles. To evaluate the system and understand its efficiency, data measured was mainly based on the accuracy and speed of data transfer between the different machines in the system. The major variables in the system are the machine learning algorithms and the IoT based servers. It was evident that the accuracy of the machine learning systems explains the system's validity, also speed of the IoT servers are sufficient to facilitate the required data transfer.

The drone utilized machine learning for image recognition and was based on CNN. The major challenges faced in this model design were the effect of sunlight in the images. The image recognition system had to be trained on a cluster of images which were collected at a different point of time varying to its use. Hence, there were times the images collected in the morning hours, when there is good sunlight, were used for training while the system itself was trained in the evening hours when the sun was already set. This caused some issues in the accuracy of the system. The system was trained at two different times during the day. For the morning hours the training was done at around 1 pm when the sunlight was at its brightest, while the evening hours training were done around 7 pm, when the sun leans towards west (during June/July). In all the scenarios, the sky was clear.

The model training performance was tested in two scenarios. The first scenario involves training and testing at the same time of the day. Where if the system was trained in the morning it would be tested in the morning and when it was trained in the evening, it was tested in the evening as well. Each time it was trained on a set of 8000 images in total with 5 trials of each case when testing. As seen from Figure 23 the systems showed 100% accuracy, where each of the 5 trials accurately achieved their goals.

The second scenario involved testing the system at a different time after training. In this arrangement the training was done in the morning, but testing was done in the evening or training was done in the evening and the testing in the morning. The training accuracy was different for these two arrangements. Accuracy attained was 40% when the images for training took place in the morning hours, while it was tested in the evening hours. With this arrangement 3 trials in every 5 trials were incorrectly detected. 20% accuracy was obtained when training images are taken in the evening hours and tested in the morning hours. With this setting 4 out of every 5 trials were incorrectly detected as shown in Figure 24.

The image recognition was also tested to see how the accuracy of the CNN changed with respect to the number of images that were available for training. The system had been tested on 500 images initially to begin with but found out that the accuracy was less than 70%, this would cause a problem for the navigation system of the drone. The accuracy of the model increased with more training samples. The training was then incremented for every 500 images, from 500 to 10,000 images, and the accuracy of the CNN was tested at each interval. The finding is plotted in Figure 25, where it shows that around 4000 images, the system achieved ~99% accuracy after which any increase of images didn't improve much.

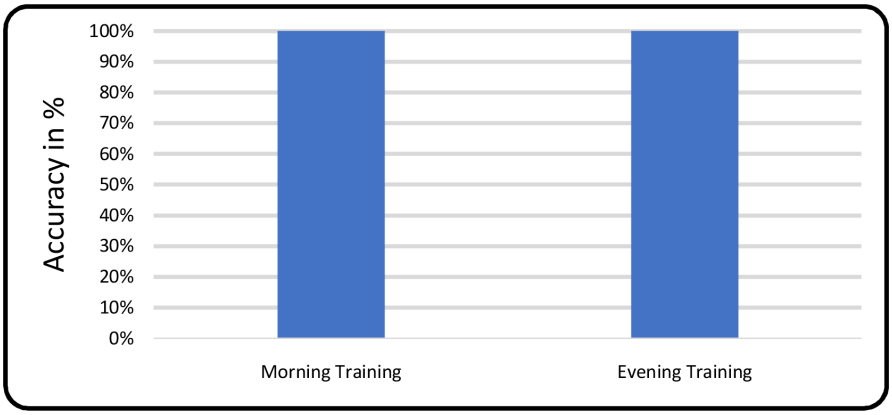

Figure 23. Image recognition accuracy with same time for training and testing.

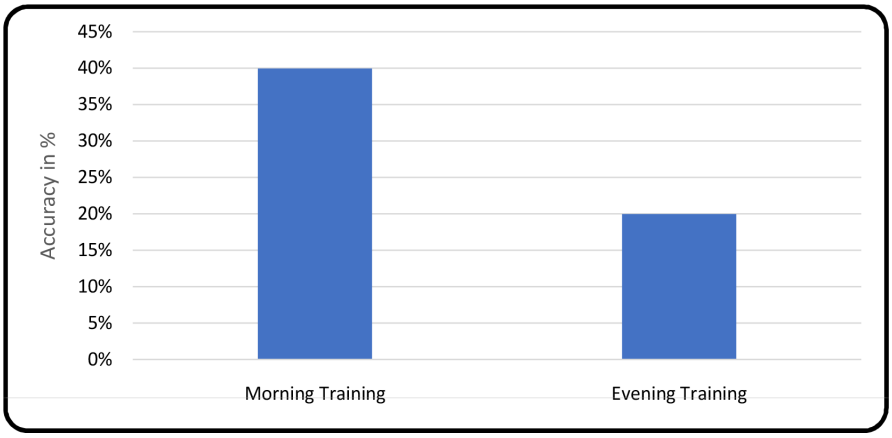

Figure 24. Image recognition accuracy with different timings for training and testing.

The area chosen for the experiment was 2.2 meters wide and 20 meters in length (Figure 26). The arrow points at the red colored object whose location have to be detected by the drone and where ground vehicles to be directed through the base station. The ground vehicles encounter blue colored obstacles (Figure 26), each of which were 0.24 meter in length 0.15 meter width and 0.06 meter in height.

Figure 27 shows the starting location of the drone and the ground vehicles. Where the drone takes off from one side of the experiment set and flies towards the red colored object. Once the location of the object is identified, the information is relayed to the base station. The base station in turns mobilizes the ground vehicles and instructs them to move towards the object using the A-Star path finding algorithm.

Figure 28 describes the experiment process in further detail. Where the image on the left shows the drone flying and trying to search for the red colored object. It flies towards the object and sends the location to the base station. The base station then mobilizes both the ground vehicles and instructs them to move

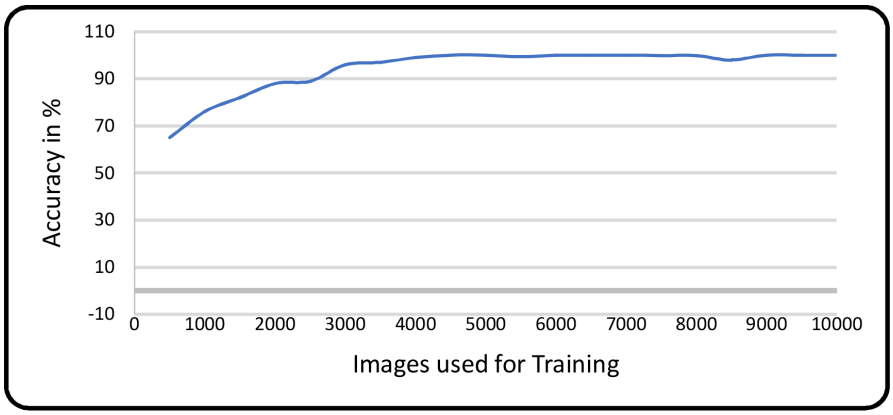

Figure 25. Image recognition accuracy as a function of number of images used for training.

Figure 26. Experiment set (red arrow pointing at the target).

Figure 27. Starting location of the drone and ground vehicle.

Figure 28. Drone flying (on the left) and vehicle moving to the object (on the right).

towards the object, which is shown as the image on the right of **Figure 28**.

The next step in the evaluation of the system was to look at the variables in the base station. The job of the base station is mainly to serve as a hub to transfer information between the machines. For this purpose, the system utilized IP connections over a local network for data transfer. As the system was based on Python, the three main sources of local network were found as Socket, Asyncore, and Twisted [32] [33] [34]. While Socket and Asyncore are part of the Python library, Twisted is an open source project that needs to be installed separately. It was found that Socket only allows one-way communication, sending or receiving in one transaction. However, Asyncore and Twisted on the other hand provided simultaneous sending and receiving of data packets. Twisted is an advanced level system, which requires seven times less amount of code to function than Asyncore. In terms of time required, in the case of this application both the techniques took the same amount of time for data transfer, and it was less than a second. Naturally Twisted was a better match. However, as the system employed Raspberry Pis, Twisted was found to be a bit cumbersome for installation. Also, Asyncore is developed on top of the Socket library. It only provided for multiplexing of data, which made simultaneous sending and receiving possible. In the case of this project, the base station had only one mode of communication. It receives information from the drone and sends corresponding information to

the ground vehicles. Hence, there was no need for a simultaneous communication system. As a result, Socket based networking worked well in this case.

5. Future Growth

There are three areas of growth that can be addressed during the next phase. The first area is to introduce a rescue operation using a robot or some other rescue vehicles. This involves planning rescue goals and facilitating appropriate mechanisms to implement the rescue operation. The second area is to address the limitation of the system in terms of processing power that was described earlier. As it is now the base station is unable to run two machine learning processes at the same time. Running two processes introduces extended latency and becomes an obstacle to achieving real time performance of the system. This issue can be addressed by having a drone with onboard processing capability where it can accommodate one of the machine learning processes. The last area of improvement is communication protocol and system security. The M2M communication within the system uses socket communication. This can be replaced with a more state-of-the-art IP communication system. This system is also subject to security concerns, which can be investigated, and a better secure system can be devised.

6. Conclusions

The paper describes the design and development of an indoor search and rescue system to demonstrate M2M communication utilizing machine learning and IoT system. The developed system works effectively in terms of autonomous drone flight, object detection and navigation of ground vehicles utilizing M2M communication and machine learning within an IoT infrastructure. The base station is hosted on a NVIDIA TX2 board which has provided processing power for machine learning for drone flight model development as well as for ground vehicle navigation. A CNN model of the test environment was developed with machine learning that allows the drone to fly autonomously with a command from the base station. While flying, the drone was also looking for the target object. Once identified it sends the object's coordinate information to the base station. With this information the base station utilizes machine learning to determine navigation path for the ground vehicles and provide driving directions. The A-Star search algorithm was used to develop ground vehicles' navigation routes. A suitable GUI was designed and developed so that the system can be operated by a user with minimum knowledge of the system.

The developed system was implemented on a field trial within a corridor of a building, and it was demonstrated successfully. The data analysis and evaluation section of the paper describes the implementation details as well as assessing the performance of various system components. This also highlights the difficulties and how they are addressed during the development and implementation process.

Acknowledgements

The authors would like to thank the NVIDIA Higher Education and Research program for supporting the project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NVIDIA.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Centenaro, M., Costa, C.E., Granelli, F., Sacchi, C. and Vangelista, L. (2021) A Survey on Technologies, Standards and Open Challenges in Satellite IoT. *IEEE Communications Surveys & Tutorials*, **23**, 1693-1720. https://doi.org/10.1109/COMST.2021.3078433

[2] Motlagh, N.H., Mohammadrezaei, M., Hunt, J. and Zakeri, B. (2020) Internet of Things (IoT) and the Energy Sector. *Energies*, **13**, Article No. 494. https://doi.org/10.3390/en13020494

[3] Ho, C.M. (2023) Research on Interaction of Innovation Spillovers in the AI, FinTech, and IoT Industries: Considering Structural Changes Accelerated by COVID-19. *Financial Innovation*, **9**, Article No. 7. https://doi.org/10.1186/s40854-022-00403-z

[4] Chirra, V.R.R., Uyyala, S.R. and Kolli, V.K.K. (2019) Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State. *Revue d'Intelligence Artificielle*, **33**, 461-466. https://doi.org/10.18280/ria.330609

[5] Munawar, H.S., Ahmed, W.A., Hammad, S. and Waller, T. (2021) A Review on Flood Management Technologies Related to Image Processing and Machine Learning, *Automation in Construction*, **132**, Article ID: 103916. https://doi.org/10.1016/j.autcon.2021.103916

[6] Semeraro, F., Griffiths, A. and Cangelosi, A. (2023) Human-Robot Collaboration and Machine Learning: A Systematic Review of Recent Research. *Robotics and Computer-Integrated Manufacturing*, **79**, Article ID: 102432. https://doi.org/10.1016/j.rcim.2022.102432

[7] Zhou, Z., Zhand, C., Wang, J., Gu, B., Mumtaz, S., Rodriguez, J. and Zhao, X. (2019) Energy-Efficient Resource Allocation for Energy Harvesting-Based Cognitive Machine-to-Machine Communications. *IEEE Transactions on Cognitive Communications and Networking*, **5**, 595-607. https://doi.org/10.1109/TCCN.2019.2925025

[8] Li, Z. and Gui, J. (2019) Energy-Efficient Resource Allocation with Hybrid TDMA-NOMA for Cellular-Enabled Machine-to-Machine Communications. *IEEE Access*, **7**, 105800-105815. https://doi.org/10.1109/ACCESS.2019.2931657

[9] Amodu, O.A. and Othman, M. (2018) Machine-to-Machine Communication: An Overview of Opportunities. *Computer Networks*, **145**, 255-276. https://doi.org/10.1016/j.comnet.2018.09.001

[10] Meng, Z., Wu, Z. and Gray, J. (2017) A Collaboration-Oriented M2M Messaging Mechanism for the Collaborative Automation between Machines in Future Industrial Networks. *Sensors*, **17**, Article No. 2694. https://doi.org/10.3390/s17112694

[11] Knight, W. (2023) Wired, Amazon's New Robots Are Rolling out an Automation

Revolution.
https://www.wired.com/story/amazons-new-robots-automation-revolution/

[12] Wessling, B. (2022) A Decade after Acquiring Kiva, Amazon Unveils Its First AMR. *The Robot Report*, June 22. https://youtu.be/AmmEbYkYfHY

[13] Bradley, T. (2023) Amazon Statistics: Key Numbers and Fun Facts. https://amzscout.net/blog/amazon-statistics/

[14] Placek, M. (2021) Number of Packages Delivered by Amazon Logistics in the United States from 2018 to 2021. Statista. https://www.statista.com/statistics/1178979/amazon-logistics-package-volume-united-states/

[15] Amazon (2022) 10 Years of Amazon Robotics: How Robots Help Sort Packages, Move Product, and Improve Safety, June 2022. https://www.aboutamazon.com/news/operations/10-years-of-amazon-robotics-how-robots-help-sort-packages-move-product-and-improve-safety

[16] Martinez-Alpiste, I., Golcarenarenji, G., Wang, Q. and Alcaraz-Calero, J.M. (2021) Search and Rescue Operation Using UAVs: A Case Study. *Expert Systems with Applications*, **178**, Article ID: 114937. https://doi.org/10.1016/j.eswa.2021.114937

[17] NVIDIA Jetson TX2. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/

[18] Apache Software Foundation (2023) Create Production-Grade Machine Learning Models with TensorFlow. https://www.tensorflow.org/

[19] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, Santamaría, O.J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021) Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. *Journal of Big Data*, **8**, 53. https://doi.org/10.1186/s40537-021-00444-8

[20] Chakraborty, K., Bhattacharyya, S., Bag, R. and Alla Hassanien, A. (2029) 7. Sentiment Analysis on a Set of Movie Reviews Using Deep Learning Techniques. In: Dey, N., Borah, S., Babo, R. and Ashour, A.S., Eds., *Social Network Analytics*, Academic Press, Cambridge, 127-147. https://doi.org/10.1016/B978-0-12-815458-8.00007-4

[21] Yang, K., Yau, J.H., Fei-Fei, L., Deng, J. and Russakovsky, O. (2022) A Study of Face Obfuscation in ImageNet. *Proceedings of Machine Learning Research*, **162**, 25313-25330. https://proceedings.mlr.press/v162/yang22q.html

[22] Chen, X., Li, H., Wu, Q., Ngan, K.N. and Xu, L. (2021) High-Quality R-CNN Object Detection Using Multi-Path Detection Calibration Network. *IEEE Transactions on Circuits and Systems for Video Technology*, **31**, 715-727. https://doi.org/10.1109/TCSVT.2020.2987465

[23] Ji, X., Yan, Q., Huang, D., Wu, B., Xu, X., Zhang, A., Liao, G., Zhou, J. and Wu, M. (2021) Filtered Selective Search and Evenly Distributed Convolutional Neural Networks for Casting Defects Recognition. *Journal of Materials Processing Technology*, **292**, Article ID: 117064. https://doi.org/10.1016/j.jmatprotec.2021.117064

[24] Arora, D., Garg, M. and Gupta, M. (2020) Diving Deep in Deep Convolutional Neural Network. *Proceedings of the* 2*nd International Conference on Advances in Computing, Communication Control and Networking* (*ICACCCN*), Greater Noida, 18-19 December 2020, 749-751. https://doi.org/10.1109/ICACCCN51052.2020.9362907

[25] Sayeed, A. and Ayesha, N. (2020) Detecting Crows on Sowed Crop Fields Using Simplistic Image Processing Techniques by OpenCV in Comparison with TensorFlow Image Detection API. *International Journal for Research in Applied Science and Engineering Technology*, **8**, 61-73. https://doi.org/10.22214/ijraset.2020.3014

[26] Uddin, M.M., Azad, A. and Demir, V. (2017) Remote Monitoring and Detection of Rail Track Obstructions. *Proceedings of the* 14*th International Conference of Remote Engineering and Virtual Instrumentation REV*, New York, 15-17 March 2017, 517-531.

[27] Goldberg, A.V. (2007) Point-to-Point Shortest Path Algorithms with Preprocessing. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H. and Plášil, F., Eds., *SOFSEM* 2007: *Theory and Practice of Computer Science*, *SOFSEM* 2007, Lecture Notes in Computer Science, Vol. 4362, Springer, Berlin, 88-102.

[28] Ma, H. (2022) Graph-Based Multi-Robot Path Finding and Planning. *Current Robotics Reports*, **3**, 77-84. https://doi.org/10.1007/s43154-022-00083-8

[29] Qin, Z., Chen, X., Hu, M., Chen, L. and Fan, J. (2020) A Novel Path Planning Methodology for Automated Valet Parking Based on Directional Graph Search and Geometry Curve. *Robotics and Autonomous Systems*, **132**, Article ID: 103606. https://doi.org/10.1016/j.robot.2020.103606

[30] Dang, T., Mascarich, F., Khattak, S., Papachristos, C. and Alexis, K. (2019) Graph-Based Path Planning for Autonomous Robotic Exploration in Subterranean Environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*), Macau, 3-8 November 2019, 3105-3112. https://doi.org/10.1109/IROS40897.2019.8968151

[31] Foead, D., Alifio Ghifari, A., Kusuma, M.B., Hanafiah, N. and Gunawan, E. (2021) A Systematic Literature Review of A* Pathfinding. *Procedia Computer Science*, **179**, 507-514. https://doi.org/10.1016/j.procs.2021.01.034

[32] Rodrigues, S.H., Anderson, T.E. and Culler, D.E. (1997) High-Performance Local Area Communication with Fast Sockets. *Proceedings of the USENIX Technical Conference*, Anaheim, 6-10 January 1997, 257-274.

[33] Hetland, M.L. (2008) Project 5: A Virtual Tea Party. In: Hetland, M.L., Ed., *Beginning Python*, Springer, Berlin, 469-487. https://doi.org/10.1007/978-1-4302-0634-7_24

[34] Fettig, A. and Lefkowitz, G. (2005) Twisted Network Programming Essentials. O'Reilly Media, Inc., Sebastopol.