Optimizing farming practices with Autonomous Vehicle using Consens method and Cognitive Operator

Dinh Quan Nguyen[1], Nhat Quang Nguyen[1], Nhat Lam Nguyen[1], Ilker Kurtulan[1], and Igor Risteski[1]

Abstract: In this paper, we present an autonomous agricultural vehicle (AV) system design, while striving to improve efficiency and sustainability in modern farming. The system consists of three primary components (units): a Plant Observer Unit used for real-time monitoring of crop health, weed detection and coordinates recording, a Weeder Unit for precise weed elimination, and a Crop Protection Unit for targeted application of pesticides on unhealthy plants. Utilizing the principles from the CONSENS project, the AV system integrates advanced sensors, machine learning algorithms, and has consistent and uninterrupted connection between components, which leads to seamless cooperation between them. We outline the mechanical requirements and operational scenarios for the AV system, demonstrating its potential to enhance weed management, disease control, and overall crop health.

1 Introduction

In the current technological era, with the emergence of smart features and devices, they are becoming more and more common in almost every single area of life. One such area, can be precision farming, aimed at optimizing the farming practices using an smart farming vehicle. In this paper, we will precisely talk about the innovative ideas behind precision farming and our idea of how to implement it into an autonomous vehicle. Our design of the AV in this project is focused on addressing the issues of traditional farming. Therefore, the primary objectives are setting up a system with autonomous navigation and image processing capabilities to ensure plant identification, while simultaneously affecting farming efficiency, plant health and food production.

2 Motivation

In traditional agriculture, farmers are faced with numerous challenges. Such challenges include big amounts of intensive labour, occasional losses of plants due to plant diseases or bacteria, insects damaging the plants, abundant fast-growing weeds, etc. Traditional methods also frequently struggle to meet the demands due to inefficiency and limitations of manual labour. In the modern era, there are a number of solutions to all of these problems. One such

1 Fachhochschule Dortmund, Embedded System Engineering, Dortmund, Germany, quan.nguyen003@stud.fh-dortmund.de; nhat.nguyen001@stud.fh-dortmund.de; nhat.nguyen003@stud.fh-dortmund.de; ilker.kurtulan002@stud.fh-dortmund.de; igor.risteski001@stud.fh-dortmund.de

solution that can address these issues, is designing an autonomous vehicle (AV) capable of detecting unhealthy plants and weeds, and dealing with them using certain tools and chemicals. By leveraging the principles of the CONSENS method and MechatronicUML, we created a design of such a vehicle, which is to be discussed further on in this paper.

3 CONSENS

The CONSENS project is part of the research at the University of Paderborn in Germany, which focuses on "COoperating eNgineering through Seamless model iNtegrationS", hence, the name "CONSENS". To implement this system, we have to set up the Application scenario, create requirements for the system, decide on the functionalities for each component, set up the systems of objectives, then create the active structure, look at how it interacts with the environment, connect all of the previously stated concepts, and go through them in a circle while constantly improving them.

3.1 Application Scenario

In order to implement this system, first of all, we had to set up application scenarios. These are specific scenarios that determine how the system should behave and what it is supposed to do in those specific cases. Setting up good application scenarios gives us a clear vision of all the requirements the system needs to have in order to work properly.

3.2 Requirements

After setting up the Application Scenarios and defining what is expected of the system, the next step is determining necessary requirements. The following figure shows the initially set requirements for our design. Here, we have defined General and Behavioural Requirements for our Autonomous Vehicle. The former ones are also known as Non-functional requirements, and they represent how the AV should look, what are the dimensions of the AV, where should the sensors be mounted, what materials should be used, what safety factors need to be satisfied, etc. On the other hand, the Behavioural requirements, also known as Functional requirements, are defining how the sensor data should be processed, how each of the three main components is supposed to act, what are the margins that are acceptable for such actions, how is the state-management done for each working and idling states of all three components, how the interact with each other, etc. Setting up the correct requirements for any system is vital, as it highly reflects on manufacturing time and general workload, system and user safety, resources spent and most importantly, the system's functionality.

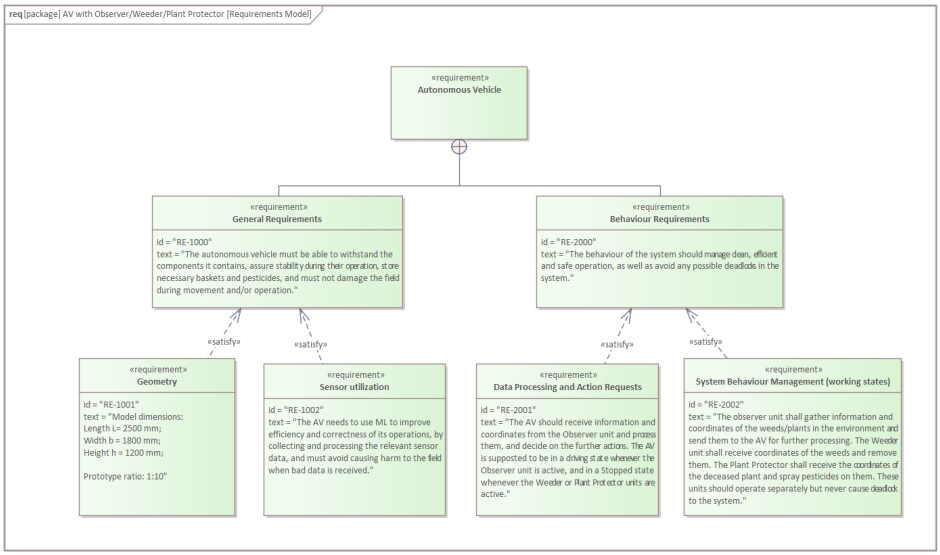

Fig. 1: Requirements Diagram

3.3 Functionality

The next stage of the system's design after setting the necessary Requirements, is to set up the systems functionality. The functionality for our system is defined in the following diagram. On the figure, it is clearly shown that the functionality is divided into three main roles. Those are: Observer, Weeder and Protector. Each role was given a different functionality based on different situations and the active behaviour it's supposed to do. These roles and functions are represented as follows:

1. Observer Functions:
Main functionalities of observer are inspecting the environment and communicating it's findings with other roles.
2. Protector Functions:
Protector role's functionalities are getting the position of unhealthy (plants with disease/insect), spraying pesticide and communicating it's state and findings with observer.
3. Weeder Functions:
Weeder functions include Spotting weed (unwanted plant), eliminating the weed and communicating it's state and findings with observer.

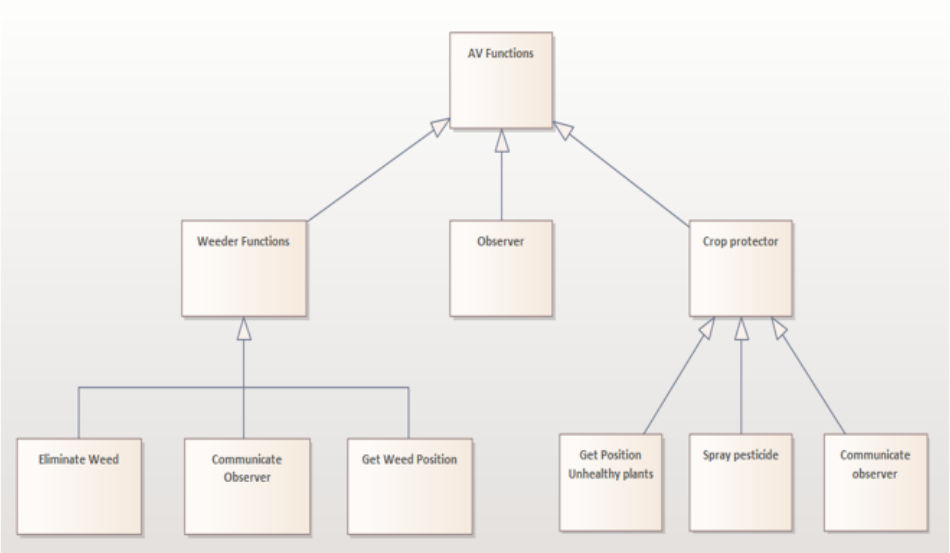

Fig. 2: Functions Diagram

3.4 System of Objectives

Objectives based on role's:
There are objectives for roles, opitimization practices, communication and reliability. Basically role objectives are the critical goals that specific for every role. For example weeder role has goals of locating weed with autonomous vehicle's resources, eliminating weed and communicating the findings (observation data).
Similarly, crop protector role also has to spot unhealth crops, keep the crops health and apply the techniques to extend crop's lifespan.
Plant observation and Monitoring can be related with observer's objectives. General objectives would be inspecting the plant's health, checking the area if there are any obstacles that can block the Autonomous Vehicle.
Optimization of Farming Practices:
Resource management and crop improvement are critical part of our objectives. Efficent resource use makes sure that we are doing sustainable farming.
Coordination and Communication:
Real-time data sharing and dynamic role assigning are essential objectives for responsive and health system. This ensures that all of our components work together perfectly and adjusting to changing conditions.
System Reliability and Maintenance:
Goals for this section increase the longevity and reliability of the system.

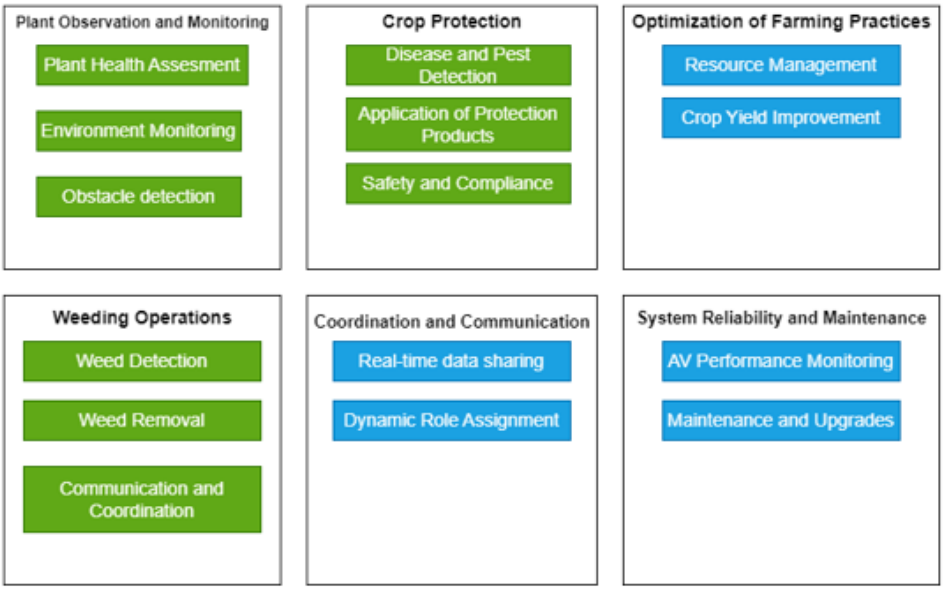

Fig. 3: System of Objectives

- **Plant Observation and Monitoring**
• Plant Health Assessment
• Environment Monitoring
• Obstacle Detection
- **Crop Protection**
• Disease and Pest Detection
• Application of Protection Products
• Safety and Compliance
- **Weeding Operations**
• Weed Detection
• Weed Removal
• Communication and Coordination
- **Optimization of Farming Practices**
• Resource Management
• Crop Yield Improvement
- **Coordination and Communication**
• Real-time data sharing
• Dynamic Role Assignment

- **System Reliability and Maintenance**
• AV Performance Monitoring
• Maintenance and Upgrading of System

3.5 Active Structure

It consist of Observer, Crop protector, weed controller and communication interface to switch between the roles.

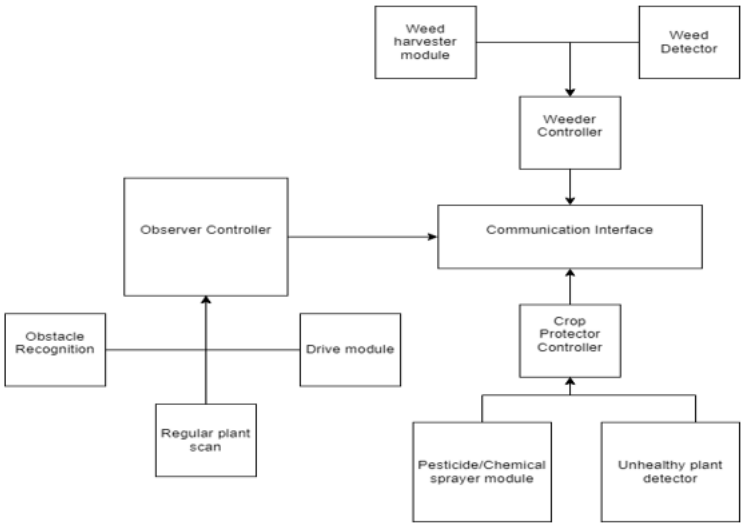

Fig. 4: Active Structure

Observer Controller:
Includes obstacle recognition, driving module and regular plant scanner. These submodules guides the AV around the farm and making sure AV avoids the obstancels and monitors plant continiously.,

Crop Protector Controller:
Crop protector contoller will be responsible for spraying chemical/pesticide and uhealth plant detection. Lastly weeder controller will be doing the harvesting and detecting the weed on the farm.
Weeder Controller:
Performs weed detection and elimination. The Weeder Controller makes sure that weeds are identified and removed properly. It prevents them to grow within our farm.

Communication Interface:
Acts as a central unit for coordination between controllers. It ensures the data from the Observer, Weeder and Protector are shared real-time and efficently.

3.6 Environment

After completing an initial design of all of the structural parts and their functions, the next stage is to determine what the possible Environment for this Autonomous Vehicle might end up being. This environment will include all of the outside factors that will in one way or another, influence the system and therefore need to be accounted for in the design.

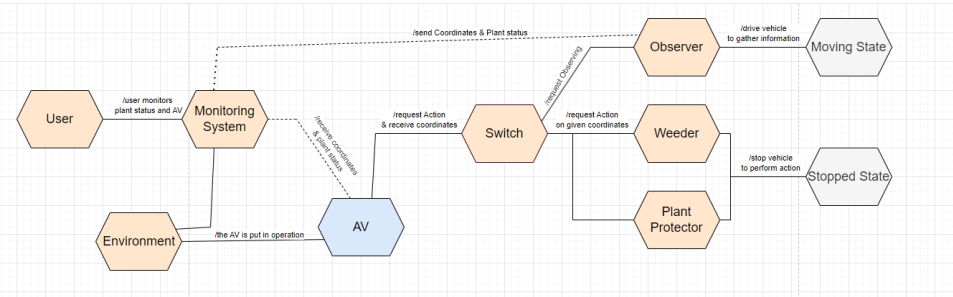

Fig. 5: Environment

Such factors for our system could be lighting conditions, which would impact how bright the environment is and how many objects are in the shade, and therefore indirectly influence the image classification model when detecting unhealthy plants or weeds. This could lead to errors and make the model inefficient, non-functional, or at worst, damaging or removing healthy plants. Another weather condition that could impact the system is the rain. Drops of water on the camera lens can impact the image processing negatively, or contribute to blur pictures. The rain also would cause the ground to be muddy, which could make the impact on the image processing even worse. In order to reduce this impact that the environment could have on our system, we need to take certain steps. One way is to feed blurry images, as well as increase the dataset for the image processing and retraining the model, so that it reacts better to such conditions. We must make sure that the annotations are done correctly, so that the model recognizes the objects easier and ends up being trained well. Future structural work can be done as well, to reduce the surface area on the sensors that could be affected. This is done by updating the structure to be able to shield such components.

4 MUML

To model and validate the intricate interactions and behaviors of the agricultural robot, the UPPAAL design of the system makes use of formal techniques. With the help of UPPAAL, it is possible to simulate and analyze how a robot would move, carry out a task, and react to various environmental factors such weed removal, obstacle avoidance, and plant recognition.

4.1 Role management

To managing the system effectively, it requires efficient coordination of the robot's roles to encompass all necessary tasks. The system facilitates sustainable farming practices and increases agricultural efficiency by guaranteeing that all roles are integrated seamlessly and maximized. The simulation of role management can be seen in figure 7.

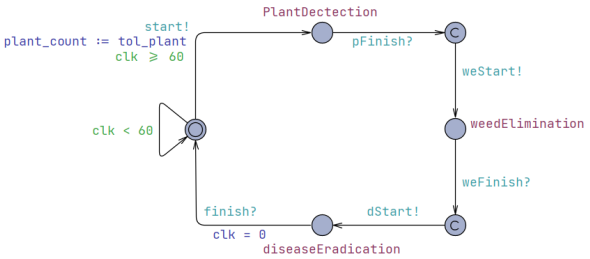

Fig. 6: Role Management

After initializing the system, the robot begins by operating as an observer to assess and identify the status of plants in the field. Once plant identification is complete, it updates the system with information on diseased plants and weeds. This data enables subsequent roles to effectively handle their respective tasks, ensuring comprehensive management of crop health and field conditions.

The robot updates the system with information on plant status after completing its observer role, then the weed eliminator and disease eradication roles sequentially activate to address weeds and treat unhealthy plants based on the updated data.. They will use the updated data to navigate to locations, which is given by observer role, in the field where weeds need removal and unhealthy plants require treatment.

4.2 Driving behavior

For simulation purposes, the driving behavior serves as the foundational operation that all roles of the robot share. This behavior includes how the robot moves across the field of

crops. The system is able to show how the robot navigates, interacts with, and performs numerous agricultural activities during the course of its operating cycle by centralizing driving as the initial activity.

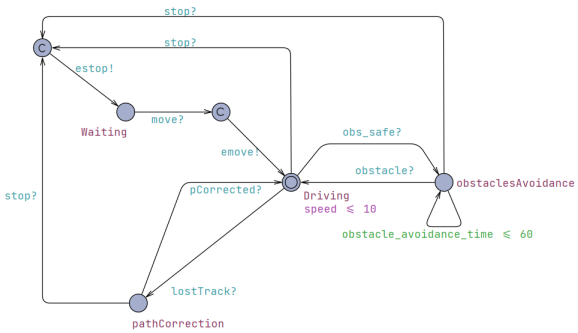

Fig. 7: Driving

As can be seen in the figure 8, there are 4 main locations of Uppaal model for driving behavior: Driving, Waiting, Path correction and Obstacle avoidance.

The system initiates in the Driving state, where it illustrates the robot's moving behavior across the field. Transition to the Waiting state occurs upon encountering a stop event triggered by the specific behaviors of each role. For instance, in plant identification, the robot will stop moving to evaluate and classify a plant it recognizes.

The system activates the Obstacle Avoidance and Path Correction states when the robot detects an obstacle within a predefined distance or deviates from its designated track. These stages are intended to guarantee that the robot can operate in the field environment both safely and effectively.

4.3 Role behaviors

Because the agricultural robot is designed to precisely detect plants, weeds, and illnesses, it is necessary to define three responsibilities in order to carry out these jobs efficiently.

The farming robot provides a holistic answer for contemporary agriculture management by combining these specialized functions, improving production, sustainability, and efficiency.

- **Observer**

As a plant identifier, the robot records and tracks the development and health of crops using advanced imagery and AI algorithms, creating crucial information for yield optimization. In

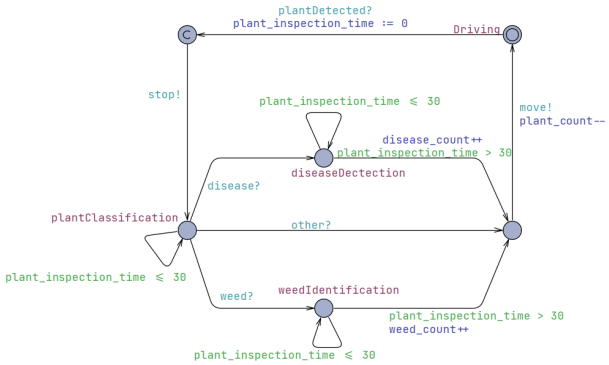

Fig. 8: Observer role

this role, the locations and condition of each plant should be provided and updated, because these details are required by the execution of other roles. Its primary duties involve traveling quickly over the field and using sensors and imaging technology to detect and take pictures in order to gather data on the number, condition, and presence of weeds as well as plant development. The role behavior can be seen in figure 8. After a plant is detected, the robot will trigger stop event. The plant is then classified and its information is updated for use by other role.

- **Weed eliminator**

The robot reaches to unwanted plant, which is detected by observer, with precision, applying herbicides selectively or using mechanical means to remove weeds without affecting the crops.

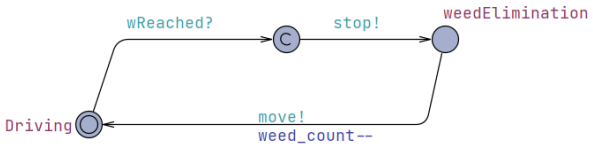

Fig. 9: Weed eliminator role

After removing the weeds, the robot updates the system with this information. Once all detected weeds are eliminated, the robot completes the tasks and prepares for the next role. This ensures systematic weed management and progression to subsequent agricultural activities.

- **Disease eradicator**

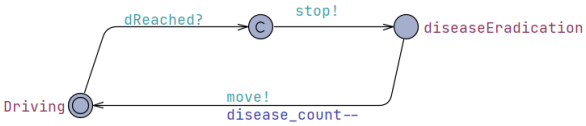

Fig. 10: Disease eradicator

The robot administers precise treatments to infected areas, preventing the spread of pathogens and ensuring the overall health of the farm.

4.4 Obstacle Avoidance

Obstacle detection is critical for the robot to operate safely and efficiently, preventing disruptions to the system's workflow and reducing maintenance costs. It ensures the robot can navigate around obstacles in the field, maintaining continuous operation and safeguarding both itself and the agricultural environment.

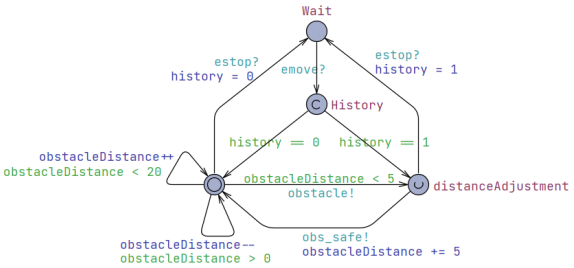

Fig. 11: Obstacle Avoidance

The simulation for Obstacle Avoidance behavior can be seen in figure 11. When the robot detects an obstacle within a critical distance, it immediately moves to the Distance Adjustment state to avoid collision. However, the priority of this behavior is lower than the robot's role behaviors, so robot will stop its current obstacle avoidance activity and transit to the Waiting state until the robot finishes its role's tasks.

4.5 Result

After simulating the Uppaal model, the system satisfy no deadlock property. The result can be seen in figure 12

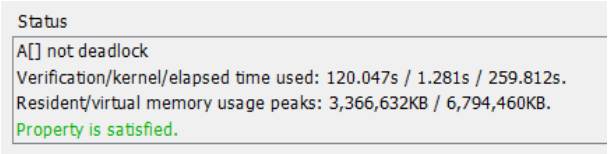

Fig. 12: UPPAAL's simulation result

5 Cognitive Operator

Cognitive part of the project consist of data preparation, model preparing and optimization. Data preparation part took two days. Collecting total amount of 1700 pictures initially. The data collection process also divided into two scenarios: the best case and the worst case. For example, best case pictures consist of AV going straight on a parkour and easy turns. Worst case pictures consist of back to back turns and intersection point.

Fig. 13: Dataset Preperation

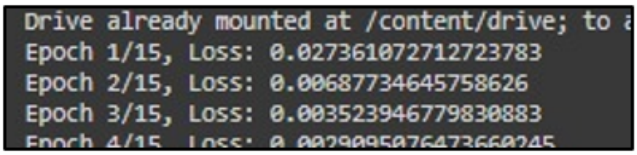

Fig. 14: Model Training

Model training has completed with resnet-18 with 15 epochs initially.

For model optimisation more pictures prepared. Total amount of pictures increased up to 3800 pictures. Later the dataset pictures are re-annotated with a custom python script. Beside these steps, worst case scenario optimized. In the worst case scenario, picture optimization is done by adding blur to copied images and using mirrored images for improved recognition.

6 Images Classification

The system will aim to identify and differentiate between healthy plants, weeds, and diseased plants. For this small-scale assessment, an experimental environment will be utilized to facilitate the image classification process.The primary objective of this experiment is to train

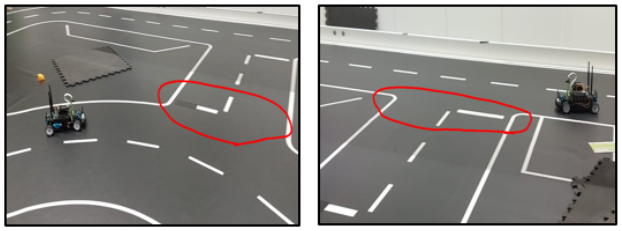

Fig. 15: Worst-case Scenario

a pre-existing model to accurately classify images of priority and stop signs. The dataset for training, testing and validating will consist of a pre-prepared collection of images depicting both priority and stop signs. Upon completion of the training process, the effectiveness of the trained model will be tested and evaluated using a confusion matrix.

6.1 Data Loading and Preprocessing

The process commenced with the creation of directories necessary for organizing the datasets. Three primary directories: Training, Validation, and Testing, were set up to house the respective data subsets required for training, validating, and testing the model.

To prepare the dataset, the total number of images available was determined by calculating the lengths of the Priority and Stop directories, thereby obtaining the total image count. Subsequently, the data was split into training, validation, and testing sets. For both the priority and stop signs, images and labels were extracted from their respective DataFrames. The 'train_test_split' function was utilized to divide the data, reserving 30% for testing. The remaining data was further split, allocating 20% of it for validation purposes.

Following the data split, image paths and labels for both categories (priority and stop signs) were combined to create comprehensive training, validation, and testing datasets. These combined datasets were then organized into DataFrames, each containing columns for image paths and labels.

To ensure an organized structure, additional subdirectories within the Training, Validation, and Testing directories were created, separating the priority and stop sign images. A custom function, copy_data, was defined to facilitate the copying of images from the source directories to their respective destination directories based on the provided path lists. This function ensured that each image was placed in the correct directory, maintaining the dataset's integrity.

The subsequent phase involved loading and preprocessing the data. A function, load_data, was defined to load the training and validation datasets using the ImageFolder class. This

function applied a series of transformations, including resizing, converting to tensors, and normalizing the images, thereby preparing them for model training.

To visualize the data, a denormalization function was implemented. This function reversed the normalization process, allowing the images to be viewed in their original form. A batch of images and labels was then obtained from the training loader. One image was selected, denormalized, and converted to a NumPy array for visualization using matplotlib.

Overall, a comprehensive environment for image classification was set up, organizing the data and providing essential tools for loading, preprocessing, and visualizing the images. This structured approach ensured that the dataset was well-prepared for the subsequent model training and evaluation phases, facilitating accurate and efficient image classification.

6.2 Model Architecture

A pre-trained ResNet-50 model was employed and modified to enhance its suitability for the specific task of image classification. Initially, the computational device was selected by determining the availability of a GPU (CUDA) or CPU, ensuring optimal performance based on the hardware capabilities. The ResNet-50 model was loaded to leverage the benefits of transfer learning. The final fully connected layer of the ResNet-50 model was then customized to match the classification requirements of this experiment, with the number of output features adjusted to correspond to the two target classes: priority and stop signs. The model was subsequently transferred to the selected device to ensure that all computations were executed on the appropriate hardware. The loss function was defined using 'nn.CrossEntropyLoss', which is well-suited for classification tasks, and the optimizer was set up using Stochastic Gradient Descent (SGD) with specified learning rate and momentum parameters. This comprehensive setup facilitated the preparation of the ResNet-50 model for efficient training and accurate classification, leveraging the advantages of pre-trained weights and appropriate optimization techniques.

6.3 Training the Model

In this segment of the experiment, a function was developed to train and validate the modified ResNet-50 model for image classification. The function, train_my_model, is designed to manage both the training and validation phases over a specified number of epochs. Initially, the function determines the appropriate computational device (GPU or CPU) and moves the model and loss function to this device to ensure optimal performance. During each epoch, the training phase begins with the model set to training mode, which enables features like dropout and batch normalization. The training data, organized in batches, is iterated over, with each batch being transferred to the selected device. The optimizer resets gradients, and the model processes the input images to produce outputs. The loss, calculated using the

criterion, is then backpropagated to update the model parameters via the optimizer. The cumulative training loss is averaged over the entire dataset for reporting.

Subsequently, the validation phase sets the model to evaluation mode, disabling features such as dropout. The validation data is also processed in batches, without gradient computation to ensure efficiency. The model's outputs are compared with the actual labels to compute the validation loss and accuracy. These metrics are averaged over the validation dataset to provide a comprehensive evaluation of the model's performance.

After each epoch, the training loss, validation loss, and validation accuracy are printed, providing continuous feedback on the model's progress. Upon completing the specified number of epochs, the function returns the trained model, now optimized for the task of classifying priority and stop signs. This structured approach ensures that the model learns effectively from the training data while being rigorously evaluated on the validation data, facilitating accurate and reliable image classification.

6.4 Evaluation

In the final stage of the experiment, the trained ResNet-50 model is evaluated using a separate test dataset to measure its performance on unseen data. The process begins with the preparation and loading of the test data. The path to the test dataset is specified, and a series of transformations, including resizing, tensor conversion, and normalization, are applied to the images. The test dataset is then loaded using the ImageFolder class, and a DataLoader is created to iterate over the data in batches.

The model's predictions are made within a no-gradient context to conserve memory and enhance inference efficiency. Each batch of test images is transferred to the appropriate device (GPU or CPU), and the model processes these images to generate outputs. The class with the highest score for each image is determined and stored in a list of predictions.

The file paths of all test images are collected by traversing the test directory. These paths, along with the corresponding predictions, are compiled into a DataFrame. The numeric predictions are mapped to their respective class labels using a predefined dictionary. A list of true labels, serving as the gold standard, is created to facilitate the calculation of the model's accuracy. This accuracy is computed by comparing the model's predictions with the gold standard labels.

Additionally, the trained model's state dictionary is saved to a file for future use, ensuring that the model can be reloaded and utilized without retraining. To further analyze the model's performance, a confusion matrix is computed, which provides detailed insights into the counts of true positive, true negative, false positive, and false negative predictions. This matrix is visualized using the ConfusionMatrixDisplay class with a blue color map to enhance interpretability.

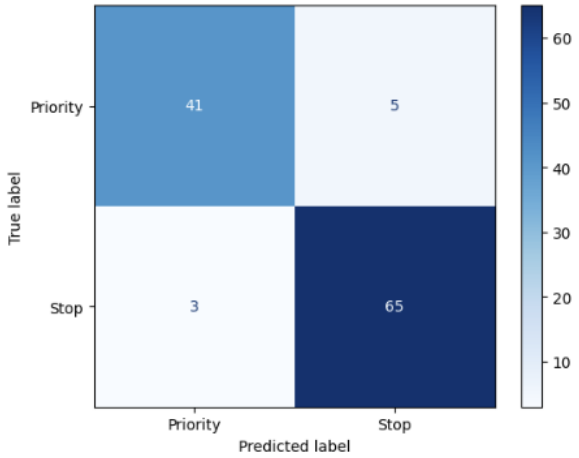

Fig. 16: Confusion Matrix

The confusion matrix visualized in the final stage of the experiment provides a detailed evaluation of the ResNet-50 model's performance on the test dataset. The matrix is a 2x2 grid where each cell represents the count of true positive, true negative, false positive, and false negative predictions for the classes "Priorityänd SStop."

- **True Positives** (Top-Left Cell): The model correctly identified 41 images as "Priority"when they were indeed "Priority."
- **False Positives** (Top-Right Cell): The model incorrectly identified 5 images as SStop"when they were actually "Priority."
- **False Negatives** (Bottom-Left Cell): The model incorrectly identified 3 images as "Priority"when they were actually SStop."
- **True Negatives** (Bottom-Right Cell): The model correctly identified 65 images as SStop"when they were indeed SStop."

The color intensity in the matrix indicates the number of instances in each cell, with darker shades representing higher counts. This visual representation helps in quickly identifying the strengths and weaknesses of the model. The high number of true positives and true negatives suggests that the model performs well in distinguishing between "Priorityänd SStopßigns. However, the presence of false positives and false negatives indicates areas where the model's accuracy could be improved.

This comprehensive evaluation process, encompassing data preparation, model inference, result compilation, accuracy calculation, model saving, and performance visualization, ensures a thorough assessment of the ResNet-50 model's classification capabilities on the test dataset.

7 Conclusion

In conclusion, the autonomous agricultural vehicle system offers a promising solution to the challenges faced in traditional farming. Its ability to perform complex tasks autonomously, coupled with its adaptability to various environmental conditions, positions it as a valuable tool for future agricultural practices. Continued research and development in this field will further enhance the capabilities of such systems, paving the way for more sustainable and efficient farming methods.

8 Affidavit

We, Dinh Quan Nguyen, Nhat Quang Nguyen, Nhat Lam Nquyen, Ilker Kurtulan and Igor Risteski herewith declare that I have composed the present paper and work by myself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form has not been submitted to any examination body and has not been published. This paper was not yet, even in part, used in another examination or as a course performance.

Dortmund, July 16, 2024

Dinh Quan Nguyen,
Nhat Quang Nguyen,
Lam Nguyen Nhat,
Ilker Kurtulan,
Igor Risteski

Bibliography

[1] S. Becker, S. Dziwok, C. Gerking, C. Heinzemann, W. Schäfer, M. Meyer, U. Pohlmann, " The MechatronicUML Method: Model-Driven Software Engineering of Self-Adaptive Mechatronic Systems " in: Proceedings of the 36th International Conference on Software Engineering (Posters), ACM, New York, NY, USA, 2014

[2] Steffen Becker, Christian Brenner, Christopher Brink, Stefan Dziwok, Christian Heinzemann, Renate Löffler, Uwe Pohlmann, Wilhelm Schäfer, Julian Suck, Oliver Sudmann, " The MechatronicUML Design Method – Process, Syntax, and Semantics ", Germany, 2012

[3] Dziwok, S., Heinzemann, C., Tichy, M. (2012). Real-Time Coordination Patterns for Advanced Mechatronic Systems. In: Sirjani, M. (eds) Coordination Models and Languages. COORDINATION 2012. Lecture Notes in Computer Science, vol 7274. Springer, Berlin, Heidelberg.

[4] Jürgen Gausemeiera, Tobias Gauksterna, Christian Tschirnera, ” Systems Engineering Management Based on a Discipline-Spanning System Model ” in: Conference on Systems Engineering Research (CSER’13).Eds.: C.J.J. Paredis, C. Bishop, D. Bodner, Georgia Institute of Technology, Atlanta, GA, March 19-22, 2013.