AgriSense

Cognitive Behavior

Authors

Akhilesh Kakkayamkode
Angel Mary
Arjun Veeramony
Kritika Premkumar
Vipin Vijayakumar

Affiliations

HOCHSCHULE HAMM-LIPPSTADT

Fachhochschule Dortmund
University of Applied Sciences and Arts

Image Regresseion

We employs a cognitive model using the ResNet-18 neural network architecture on a rover equipped with Nvidia JetRacer and a camera for autonomous navigation and traffic sign recognition. This advanced setup allows for efficient path following and accurate identification of traffic signs, leveraging deep learning for high-level feature extraction and decision-making.

01

Data Collection

Collected 1,020 images annotated with (x, y) coordinates indicating the ideal driving path on the road. Afterward, we manually removed any incorrectly marked or improper images.

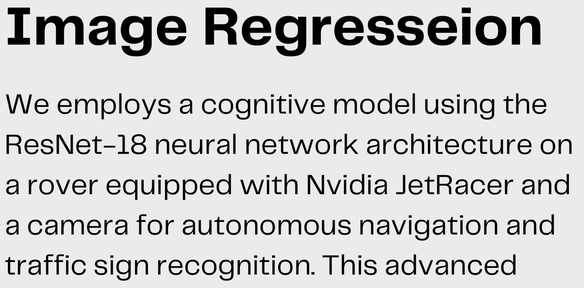


Fig: Unmarked and marked images

02

Feature Extraction

Utilized ResNet18, a convolutional neural network, implemented using the PyTorch library, to extract meaningful features from the collected images.

03

Model Training

The extracted features were then used to train a regression model. We trained the model for 10 epochs using PyTorch library, achieving an accuracy of 97.5%.

04

Model Optimization

For optimization, we used the TensorRT library to enhance inference speed and efficiency on embedded platforms. Leveraging TensorRT, we transformed the trained PyTorch model into a format compatible with NVIDIA GPUs, ensuring accelerated performance without compromising accuracy.

05

Deployment

The optimized model, now in TensorRT format, was seamlessly deployed on the JetRacer platform. This deployment strategy not only streamlined integration but also facilitated real-time predictions of the continuous (x, y) coordinates for the ideal driving path. The efficient deployment on JetRacer underscores our commitment to delivering robust and responsive solutions for autonomous navigation applications.

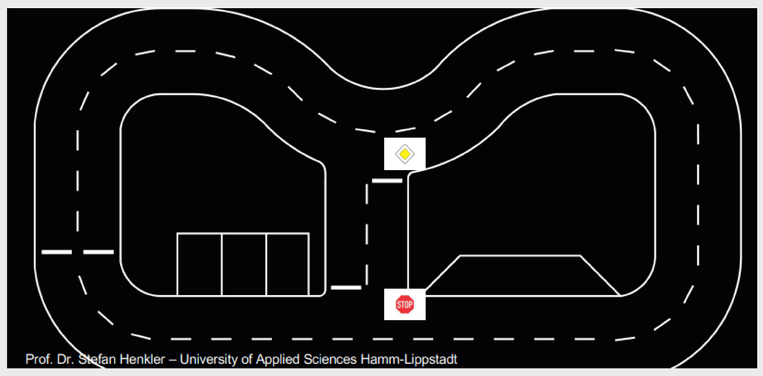


Fig: Simplified model of Race track

Image Classification

We tackled a binary classification problem to classify two different categories: priority roads and stop signs. The training phase faced challenges like varying lighting conditions and required transfer learning.

Fig: Traffic sign for Stop

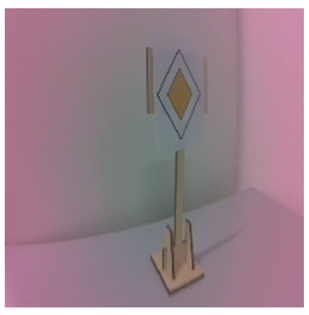


Fig: Traffic sign for Priority road

We trained the classification model for 10 epochs, achieving an accuracy of 98.56% on the validation set. The accuracy on the test set will be calculated by us.

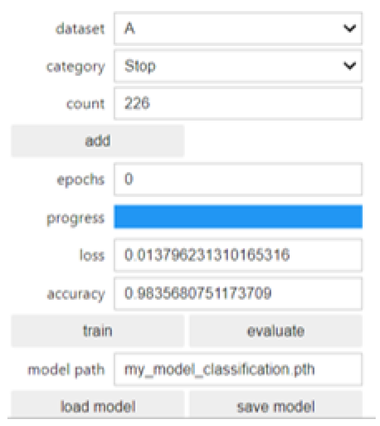


Fig: Training information widget

AgriSense

System Specification with COSENS

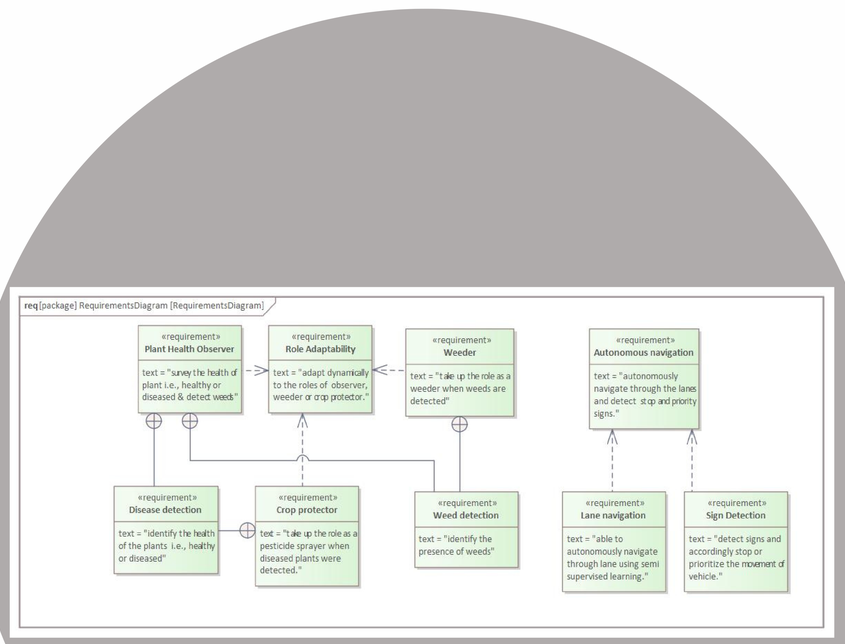
Authors

Akhilesh Kakkayamkode
Angel Mary
Arjun Veeramony
Kritika Premkumar
Vipin Vijayakumar

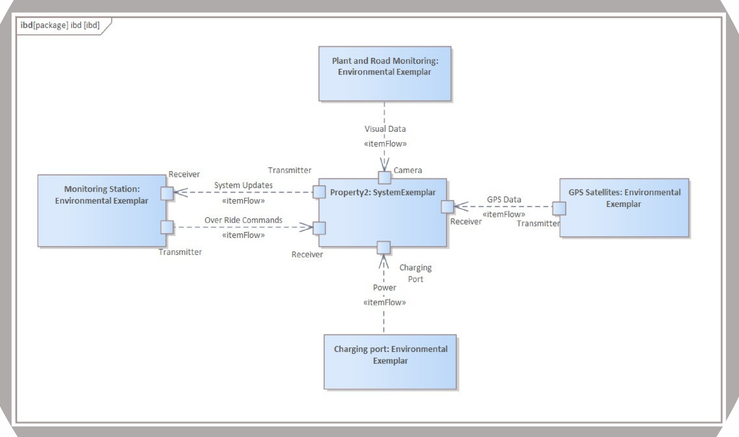
Affiliations

HOCHSCHULE HAMM-LIPPSTADT

Fachhochschule Dortmund
University of Applied Sciences and Arts

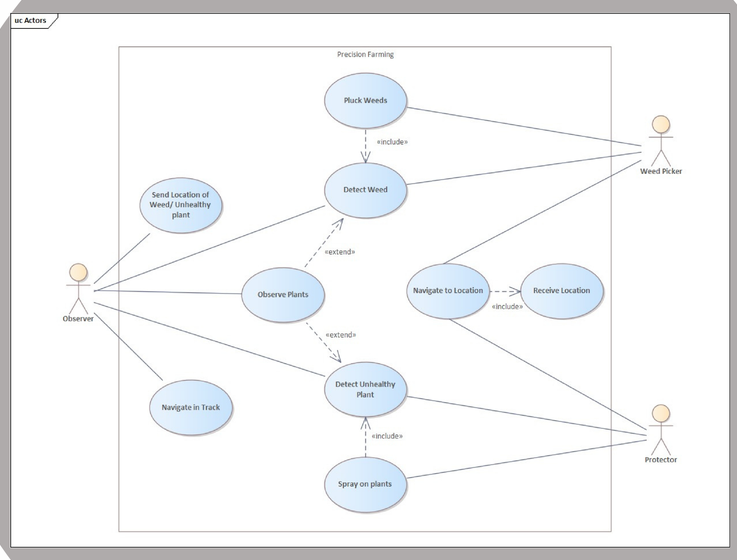


Requirements diagram

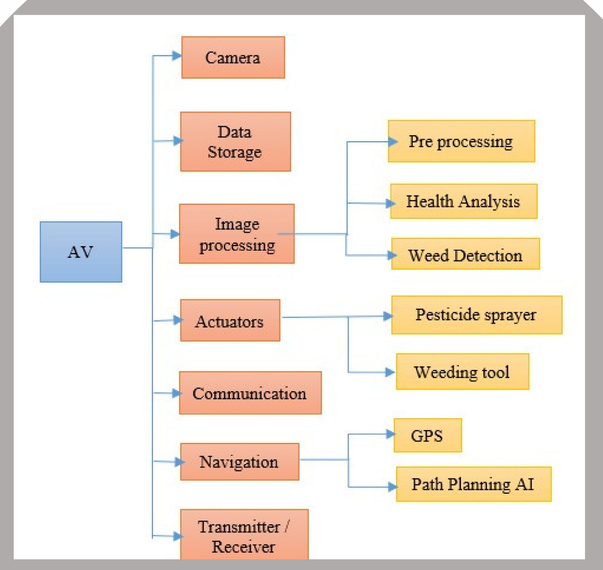


Internal block diagram

Specification Technique
CONSENS



Usecase diagram



Block definition diagram

AgriSense

Mechatronics UML Specification

Authors

Akhilesh Kakkayamkode
Angel Mary
Arjun Veeramony
Kritika Premkumar
Vipin Vijayakumar

Affiliations

HOCHSCHULE HAMM-LIPPSTADT

Fachhochschule Dortmund
University of Applied Sciences and Arts

Model Checking

The model-checker is an automated method that allows for the verification of specific properties of the system, such as reachability of desired states, absence of deadlocks, or the verification of temporal properties. It exhaustively explores the state space of the modelled system, enabling the verification of whether the specified properties are satisfied by the model or not. Since testing the entire system for all possible states would be extremely enormous, only crucial properties of the system have been extracted and modelled. Properties such as deadlock freedom and reachability have been taken into consideration here and verified.

Roles

The key roles in the system have been identified as the navigation, protector and observer.

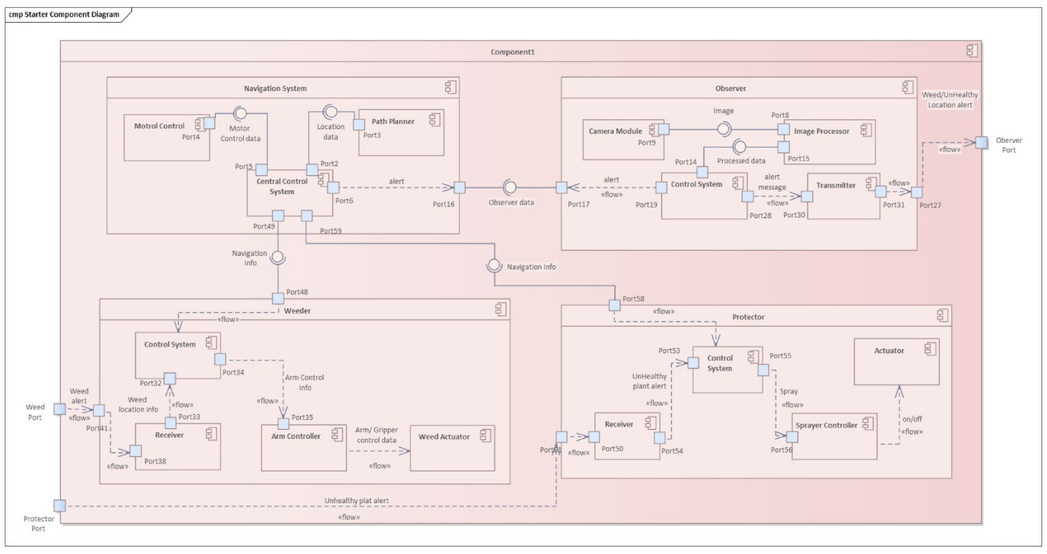


Fig: Component Diagram

01

Navigation

The navigation role begins only once it receives the event to Start and continues to move until the desired position is reached(for simplicity, only the horizontal axis is considered).Upon reaching, the transition is made to the LocationEnd state and the event Reached is generated to signal the Protector to start its work.

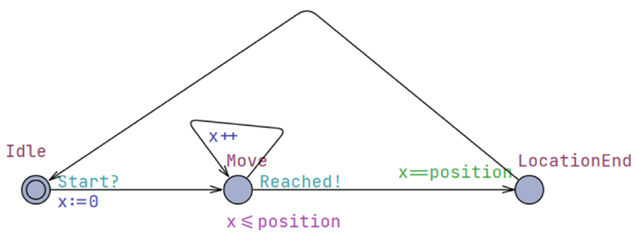


Fig: Automata for Navigation
Modelled in UPPAAL

02

Observer

The observer is modelled to detect issues regularly and informs the protector on the location that needs to be travelled to begin “protection”. The transition from SendData to the Initial location is done only once the protector confirms that the data is received, to avoid message loss.

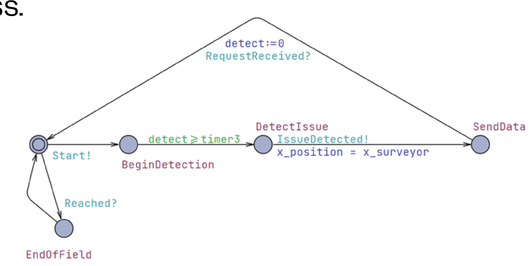


Fig: Automata for Observer
Modelled in UPPAAL

03

Protector

Protector begins its work once it receives the information that an issue has been detected. In order to ensure the synchronization, the protector informs the detector that the request has been received, there after the observer is allowed to continue with the survey.

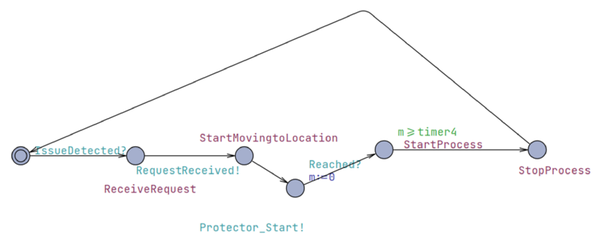


Fig: Automata for Protector
Modelled in UPPAAL

These three roles synchronize with the help of channels and work together to achieve the desired functionality. Upon modelling the system using UPPAAL, using the Model checker inbuilt within UPPAAL, key properties such as deadlock freedom and reachability of certain states have been checked. Another property to ensure the functionality is to check that the detector sending the data would imply that the Protector would receive the request.

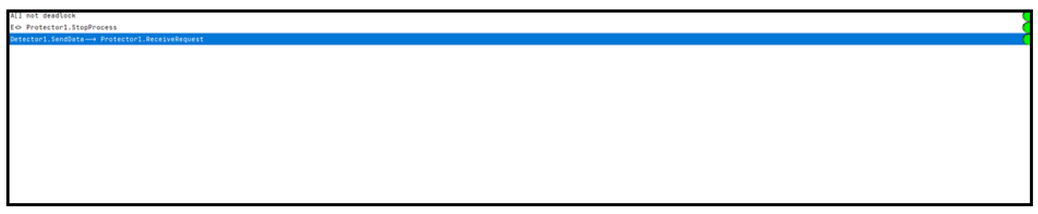


Fig: Model Checker Verification Result