Truck Platooning

Stephanie Okosa, Maxim Emile Speczyk, Muhammad Umer Bin Yaqoob[1]

Contents

[1] stephanie-chinenye.okosa@stud.hshl.de, muhammad-umer-bin.yaqoob@stud.hshl.de, maxim-emile.speczyk@stud.hshl.de

Abstract: This project presents an autonomous truck platooning system's design, implementation, and testing. This technological solution promises to revolutionise the freight industry by increasing efficiency and safety while reducing costs. The system, modelled using UML, incorporates a machine learning algorithm to select the platoon leader based on multiple parameters. It considers various scenarios like steering, lane change, braking, joining, and leaving the platoon, including potential communication failures implemented in UPPAAL. The project will culminate in a comprehensive simulation environment using Python.

1 Motivation

Autonomous vehicles have revolutionised our society, bringing with them a myriad of opportunities and challenges. Our team has embarked on a project to develop an autonomous truck platooning system that can simulate the steering, braking, lane change, coupling and decoupling of the trucks during motion. Truck platooning essentially involves several trucks equipped with advanced driver assistance systems. These trucks follow each other closely on a motorway, literally forming a convoy or 'platoon'. The importance of this arrangement lies in the many benefits it offers which include; improved road safety, reduced fuel consumption, reduced aerodynamic drag, lower CO_2 emissions and more efficient traffic flow.

2 Foundation

In the early stages of our project, each member of our team brainstormed and generated detailed scenarios for our autonomous truck platooning system. This step helped us to identify key functional requirements and gain a deep understanding of the problem from the user's perspective. We then used UML (Unified Modelling Language), to visually represent these scenarios and the system as a whole, detailing the relationships and data flow between entities. The UML designs served as blueprints for the architecture of our system, providing a solid foundation for model specification and final implementation, while facilitating efficient communication and shared understanding among team members and stakeholders.

2.1 Requirement Diagram

Initial requirements of the truck platooning system were discussed, the following requirements were iterated, and identified as critical for the system to function properly as shown in [1] below.

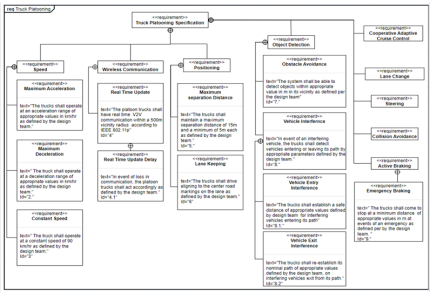

Fig. 1: Requirement Diagram
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/requirements/truckPlatooningRequirements.png

2.2 State Machine Diagrams

In this section we show the different scenarios states to enable us model the the timing behaviour in UPPAAL.

2.2.1 Braking Scenario

The autonomous truck platooning project focuses on synchronised braking mechanisms across the fleet. When the master truck initiates braking, every other truck must follow suit to maintain safe following distances and avoid collisions. This is achieved through efficient communication paths between the lead and following trucks as shown in [2]below.

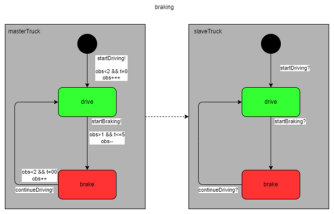

Fig. 2: Braking Scenario
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/state%20Machine/braking.png

2.2.2 Steering Scenario

This scenario in [3] ensures that the subordinate, or 'slave', trucks follows the lead or 'master' truck within the platoon at real time by, implementing a communication protocol wherein the leading, or 'master', truck transmits directional commands to the follower trucks.This facilitates seamless and synchronized transitions between directional states, e.g., initiating rightward or leftward shifts. The behaviour of this brake control system is illustrated by a state machine diagram in our system design.

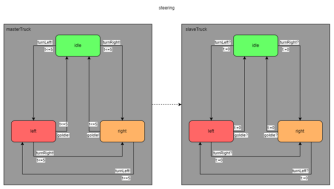

Fig. 3: Steering Scenario
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/state%20Machine/steering.png

2.2.3 Lane Change Scenario

Lane change coordination is key within the truck platoon fleet. The aim is to ensure that when the master truck makes a lane change, the slave trucks systematically reproduce the action to maintain the integrity of the platoon. These lane changes are managed by a highly responsive communication system between the trucks as shown in [4] below.

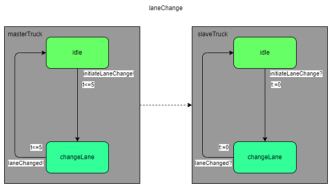

Fig. 4: Lane Change Scenario
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/state%20Machine/laneChange.png

3 UPPAAL IMPLEMENTATION

Here data, signal, events that are required for interactive communication between the trucks are identified, appropriate protocol for each scenario using timed automata specified, and modelled using UPPAAL. All scenarios described in this section model a platoon system where one truck acts as the 'truckMaster' and the two slave trucks, 'truckSlave' follow.

3.1 UPPAAL Braking Modelling

Here, the master truck leads the platoon, broadcasting driving instructions via three primary channels;'continueDriving', 'Brake', and 'continueBraking'. Depending on these signals, the slave trucks respond by maintaining speed, applying brakes, or continuing to brake. This synchronization enables the platoon to operate harmoniously and safely.

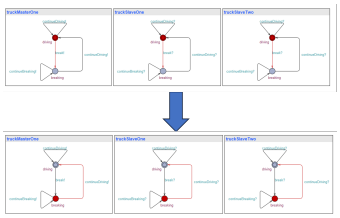

Fig. 5: UPPAAL Braking
`https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/braking.xml`

3.2 UPPAAL Steering Modelling

The steering behaviour of both trucks is defined by three states: idle, turnLeft and turnRight. The master truck's state transitions are determined by a global clock and its own synchronisation events, while the slave truck mimics the master's steering actions. The model thus demonstrates a coordinated steering system in a truck platoon, ensuring that the slave truck accurately replicates the directions of the master.

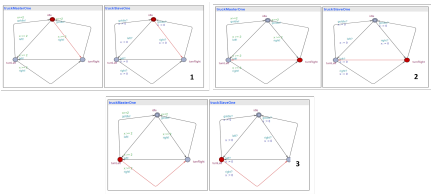

Fig. 6: UPPAAL Steering
`https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/steering.xml`

3.3 UPPAAL Lane Change Modelling

Lane change is initiated by the lead truck after a set time, prompting the following trucks to begin the same procedure. A 'Watchdog' entity monitors the operation, ready to raise an alarm if the lane change takes longer than expected. Upon completion, the lead truck signals the end of the lane change, upon which all trucks resume cruising. This system enables safe and synchronized lane changes within a truck platoon.

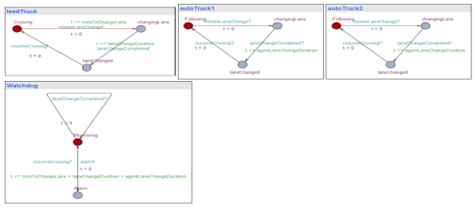

Fig. 7: UPPAAL Lane Change
`https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/laneChanging.xml`

3.4 UPPAAL Truck Coupling and Decoupling Modelling

In this scenario, each truck initially operates independently until it identifies a suitable environment for platooning, such as a motorway. When it detects a potential platoon, the truck sends a request to join, which is evaluated by the existing platoon, taking into account factors such as the size of the platoon, the compatibility of the new truck, and the potential benefits and risks. If the request is approved, the truck joins the platoon, adjusting its speed and positioning to form a coordinated and efficient transport unit. Critical to this operation is the constant communication between the trucks, which is monitored by a watchdog unit that ensures consistent data exchange and monitors for potential anomalies. If the watchdog detects a communication disruption within a truck that exceeds a certain threshold, the affected truck is removed from the platoon until the problem is resolved. In the event of a major malfunction affecting the entire platoon, the platoon disbands and each truck returns to independent operation, demonstrating the resilience and adaptability of the system.

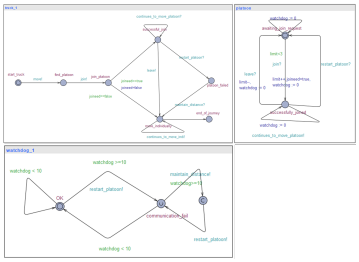

Fig. 8: UPPAAL Coupling and Decoupling
`https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/couplingAndDecoupling.xml`

4 Machine Learning Algorithm

To determine the lead vehicle of the platoon for simulating the various scenarios, machine learning algorithm is applied on data information of the trucks with predefined features to

train our model to autonomously identify the 'truckMaster' for each platoon. Two different methods are applied in order to determine the algorithm with a better prediction accuracy.

4.1 Decision Tree Classifier

Here, the decision tree classifier is used for classification of the 'truckMaster' from the 'truckSlave' in the platoon using a set of predefined features as shown in [9]. However the prediction score is 50% therefore, we try the isolation forest model to improve the prediction accuracy of the model.

	distance_route	max_route_match	fuel_consumption	body_characteristics	equipment_sensors	leader_vehicle
0	808	100	245	3769	91	0
1	899	42	308	2732	82	1
2	869	43	237	3576	72	0
3	823	35	303	2934	73	0
4	897	50	253	3649	57	0

```
# Define features and target
x = data.drop(['leader_vehicle'],axis=1)
y = data.leader_vehicle

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)

# Create a DecisionTreeClassifier and fit it to the training data
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf = clf.fit(x_train,y_train)

# Predict the test set results
y_pred = clf.predict(x_test)

# Print the accuracy of the model
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Plot the decision tree
plt.figure(figsize=(15,15))
clf = DecisionTreeClassifier(max_depth = 3).fit(x, y)
plot_tree(clf, filled=True, fontsize=9)
plt.show()
```

Fig. 9: Decision Tree Classifier
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/Leader%20vehicle%20selection/decisionTreeClassifier/DecisionTreeClassifier.py

4.2 Isolation Forest Model

The isolation forest model is based on decision trees and is used in identifying anomalies in a dataset by employing binary trees. The idea is to be able to spot the distinct trucks which would differ from the other trucks to be the 'truckMaster'.

	Relative Position	Speed	Acceleration	Gap	Sensor Data	Vehicle Characteristics	Behavior	Time Duration
0	1.470267	84.399199	0.587861	8.252372	Object Detected	Truck	Consistent	546.140477
1	16.462886	0.795604	2.117795	22.165324	Object Detected	Truck	Consistent	1460.835498
2	18.014825	75.953607	-9.773962	44.093501	Object Detected	Truck	Erratic	2326.726685
3	12.406913	24.578577	6.652652	38.285725	Object Detected	Truck	Erratic	215.892749
4	85.340544	63.854514	8.230661	36.547894	No Object Detected	Truck	Consistent	2039.669324

```
# Load the dataset from the csv file
data = pd.read_csv('truck_platoon_dataset.csv')

# Remove the 'Vehicle Characteristics' column
X = data.drop('Vehicle Characteristics', axis=1)
y = data['Vehicle Characteristics']

# Encode the categorical features using LabelEncoder
label_encoder = LabelEncoder()
for column in X.columns:
    if X[column].dtype == 'object':
        X[column] = label_encoder.fit_transform(X[column])

# Split the data into training and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig. 10: Isolation Forest Model [1]
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/tree/main/Leader%20vehicle%20selection/islolationForestModel

Fig. 11: Isolation Forest Model [2]
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/tree/main/Leader%20vehicle%20selection/islolationForestModel

With the random selection of data for fitting the model, the prediction accuracy is a 1.0. However, limitations lie in the overfitting of the test samples, as the model is not trained to consider all possible features, the model fitting does not have any dependency on the feature set as it is trained randomly i.e., no related data to find the 'truckMaster', and the 'Gap' feature data is incorrect as the value is expected to be constant in order to meet the 'fuel efficiency' requirement. In order to mitigate some of these constraints and optimize prediction, we can use pass optimization on the dataset, have some adjustments made to the feature parameters, as well as fitting the model with the feature that have correlation for identifying which is the 'truckMaster' of the platoon.

5 Truck Platoon Scenarios Simulation

Python is used as a simulation environment for implementing the various modelled scenarios in UPPAAL. The Python code provided uses the asyncio library to simulate a platoon of vehicles with a leader and several followers.

5.1 Braking Simulation

Each vehicle is represented by a class, with the leader switching between cruising and braking states based on specified time intervals, and the followers proceed. The simulation runs to completion with all vehicles operating simultaneously as shown in [12] below.

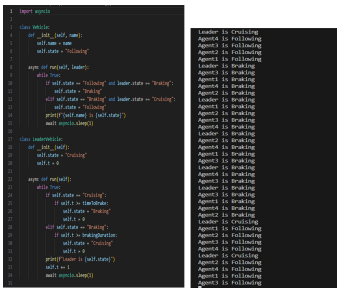

Fig. 12: Braking Simulation
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/Python%20simlutions/Braking.py

5.2 Steering Simulation

The leader truck cyclically changes its state (idle, turn left, turn right) every second and stores these states in a queue. The follower mimics the leader's states, but with a delay of 2 seconds. The script uses the asyncio library to allow the leader and follower to operate simultaneously in a non-blocking manner, providing a basic model of a delayed vehicle platooning scenario.

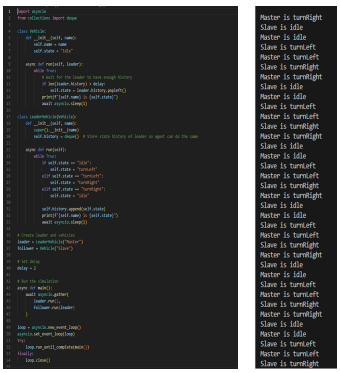

Fig. 13: Steering Simulation
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/Python%20simlutions/steering.py

5.3 Lane Change Simulation

Here, the leader changes lanes after a certain time, and the followers proceeds after a second delay. The state of each vehicle is monitored and updated using an asynchronous event loop from the asyncio library. After a successful lane change, the vehicles return to their initial state of following the leader, who returns to cruising mode as shown in [14] below.

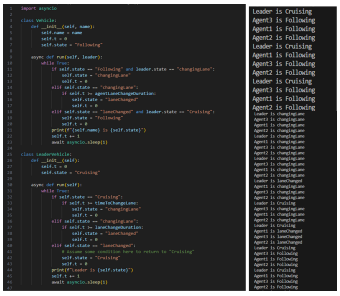

Fig. 14: Lane Change Simulation
https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/Python%20simlutions/Lanechangef.py

6 Declaration of Originality

I, Stephanie Chinenye Okosa, herewith declare that I have composed the present paper and work by myself and without the 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 have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.

Date & Place - Stephanie Okosa

I, Maxim Emile Speczyk, herewith declare that I have composed the present paper and work by myself and without the 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 have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.

Date & Place - Stephanie Okosa

I, Muhammad Umer Bin Yaqoob, herewith declare that I have composed the present paper and work by myself and without the 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 have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.

Date & Place - Stephanie Okosa