Czech Technical University in Prague
Faculty of Electrical Engineering

BACHELOR THESIS

Cooperative Collision Avoidance of Road Vehicles

Prague, 2011 **Author: Pavel Janovský**

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 26.5.2011

Janovský

podpis

Acknowledgements

First and foremost I thank to Ing. Jiří Vokřínek and Ing. Antonín Komenda from the Agent Technology Center on Czech Technical University for a guide throughout this project and valuable advices on the project design.

Furthermore I thank my colleagues Martin Schaefer, Aleš Franěk and Karel Jalovec for an effective cooperation on designing and implementing the simulator.

Abstrakt

Multi-agentní systém je přístup k řízení, který byl úspěšně použit při simulaci vyhýbání letadel. Používá různé kooperativní a nekooperativní metody řízení letadel ve třídimenzionálním prostoru. Highway project byl založen za účelem vytvoření dálničního simulátoru a implementace některých řídících metod použitých při vyhýbání letadel.

Naším prvním úkolem v highway project a prvním tématem této práce je implementace dálničního simulátoru založeného na platformě Alite vytvořené v ATG Center. Realizovali jsme všechny části multi-agentního systému. V této práci je popsán přístup pomocí multi-agentního systému a reprezentace světa v dálničním simulátoru.

Druhým úkolem a zároveň hlavním tématem této práce je implementace kooperativních metod a jejich použití při řízení dálniční dopravy. Navrhli jsme a realizovali první metodu, cooperative basic method. Druhá, cooperative peer to peer algorithm, byla vyvinuta pro vyhýbání letadel v ATG Center, ale musela být upravena a realizována pro použití při řízení dálniční dopravy. Obě kooperativní metody a jejich porovnání s nekooperativními metodami a metodami strojového učení jsou popsány v této práci.

Abstract

Multi-agent system is an approach to control which was successfully used in the simulation of airplanes collision avoidance. It uses various cooperative and non-cooperative methods to control the airplanes in three dimensional space. The highway project was established to create a highway simulator and implement some control methods used in the airplanes collision avoidance.

Our first objective in the highway project and the first topic of this thesis is an implementation of a highway simulator based on an Alite toolkit developed in ATG Center. We implemented all the parts of a multi-agent system. In this thesis we described the multi-agent system approach and the world representation in the highway simulator.

The second objective and also the main topic of this thesis is an implementation of cooperative control methods and their use in a highway traffic control. We developed and implemented the first control method, cooperative basic method. The second one, cooperative peer to peer algorithm, was developed for airplanes collision avoidance in ATG Center, but had to be modified and implemented for use in the highway traffic control. Both cooperative methods and their comparison with non-cooperative and machine learning methods are described in this thesis.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Pavel Janovský

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Kooperativní metody vyhýbání silničních vozidel

Pokyny pro vypracování:

1. Nastudujte tématiku kooperativního vyhýbání silničních vozidel.
2. Seznamte se s agentní platformou a simulátorem.
3. Navrhněte metody kooperativního vyhýbání vozidel.
4. Implementujte a validujte navržené metody pomocí simulace.

Seznam odborné literatury:

[1] Šišlák, D.; Volf, P.; Pěchouček, M.; Suri, N.; Nicholson, D.; Woodhouse, D.: Optimization-based Collision Avoidance for Cooperative Airplanes, 2009.
[2] Vokřínek, J.; Jiránek, J.; Hodík, J.: Agent-based Highway Traffic Simulation

Vedoucí bakalářské práce: Ing. Jiří Vokřínek

Platnost zadání: do konce zimního semestru 2011/2012

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 4. 2. 2011

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Pavel J a n o v s k ý

Study programme: Electrical Engineering and Information Technology

Specialisation: Cybernetics and Measurement

Title of Bachelor Project: Cooperative Collision Avoidance of Road Vehicles

Guidelines:

1. Study the theme of cooperative avoidance of vehicles.
2. Get to know the agent platform and the simulator.
3. Design methods of cooperative avoidance of vehicles.
4. Implement and validate designed methods by simulation.

Bibliography/Sources:

[1] Šišlák, D.; Volf, P.; Pěchouček, M.; Suri, N.; Nicholson, D.; Woodhouse, D.: Optimization-based Collision Avoidance for Cooperative Airplanes, 2009.
[2] Vokřínek, J.; Jiránek, J.; Hodík, J.: Agent-based Highway Traffic Simulation

Bachelor Project Supervisor: Ing. Jiří Vokřínek

Valid until: the end of the winter semester of academic year 2011/2012

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická
L.S.

prof. Ing. Boris Šimák, CSc.
Dean

Prague, February 4, 2011

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multi-agent systems bring a new way of looking on distributed systems. The basic idea is that a system composed of many autonomous intelligent entities can work better and more efficient than a centralized one. These entities we call agents. There are many approaches to design an agent, some will be discussed further.

Agents Technology Center (ATG) on Faculty of Electrical Engineering is researching in the field of agent-based computing, multi-agent systems and agent technologies. One of ATG's working projects is AgentFly. The AgentFly is described in the AgentFly project page (ATG, 2011*a*):

> AgentFly is a multi-agent system enabling large-scale simulation of civilian and unmanned air traffic. The system integrates advanced flight path planning, decentralized collision avoidance with highly detailed models of the air planes and the environment.

Our first objective was to implement a similar simulator for highway traffic. On this first part of the project I worked with my two colleagues, Martin Schaefer and Aleš Franěk. We implemented a simulator with entities moving in an environment. Our agents represent cars on the highway. Their goal is to go to the end of the highway and try to optimize the drive. The collision avoidance is specified clearly. Every agent should do it's best not to crush. When the collision avoidance is achieved, agents can concentrate on optimizing their drive. The optimization can be achieved in many ways, agents can try to make their drive time as short as possible or to change lanes as rarely as possible. The specific optimization we used will be discussed further.

Our objective was also to implement some of the AgentFly control methods to highway traffic simulation. There are several approaches to the control of the agents. Cooperative

approach allows agents to share their data with one another. On the contrary non-cooperative approach allows each agent to see only his own data. The third approach we decided to implement is machine learning, where the agents should learn from their previous mistakes.

My objective in the second part of the project was to implement the cooperative drive. My two colleagues had different objectives which we can see in figure 1.1. Because we will compare all implemented methods we will very briefly focus on the summary of their work too.

In this thesis we will describe multi-agent systems and our approach to the agent and environment implementation, we will describe the event based simulator we used, then we will focus on various control methods with great emphasis on cooperative drive. We will describe two implemented cooperative methods: basic cooperative algorithm and peer-to-peer algorithm. Finally we will demonstrate and discuss test results of each control method.

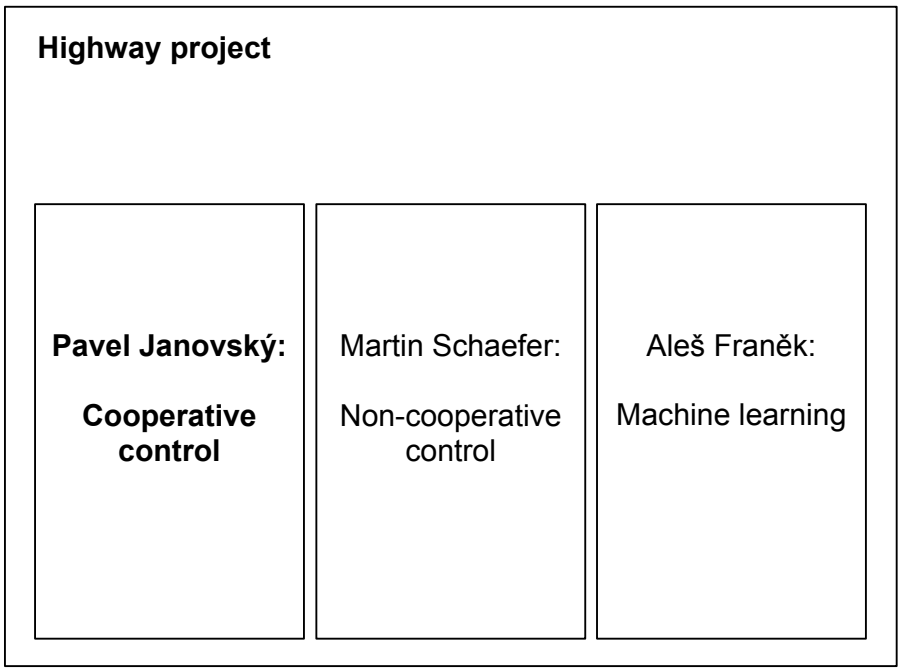

Figure 1.1: Objectives definition in highway project

Chapter 2

Multi-agent systems

Multi-agent systems are simulated or real systems which are trying to research and benefit from a decentralized approach on a control of some community. They are trying to understand the behaviour of a large group of intelligent units, that can interact with each other in various ways. A good example of a system with this complexity could be animal communities, for example ants or bees. These communities show us that the whole complex works much better than a sum of it's parts. There is no reason why such a system shouldn't work in our modern world. Our highways are kind of a multi-agent systems too.

The highway is an environment on which the cars are moving to their destination. The difference between the real world and multi-agent systems is that in many multi-agent systems entities are trying to cooperate with each other to achieve an optimized solution not only for themselves, but for the whole community. On the real highway drivers don't consider the full system optimization, they basically just care for themselves. This leads to a question whether a multi-agent system could help to improve the real system behaviour. It could optimize the traffic flow, remove traffic jams, reduce fuel consumption. Almost every car nowadays uses a GPS navigation. Also other systems like adaptive cruise control, which helps to keep a constant safe distance to a car ahead, automatic parking systems and many others are becoming components of many modern cars. These systems could be used with a set of sensors to implement a multi-agent system in the real world. The biggest problem apart of the technical ones would be that then drivers would have to obey the commands from an onboard computer, even if it meant to slow down to lower speed than the car could in the current situation go.

Multi-agent systems often look very similar to examples in the nature. There is always an environment in which various entities are operating. For more information about multi-agent systems see for example (KUBÍK, A., 2004).

2.1 Environment

The environment stores information about the world, in which the entities are moving, in specified data storages. A storage is a place where all data about agents and environment is saved. The environment is defined by it's borders and then specified by parameters according to the given simulation. In our project the environment stores in it's highway storage all information about the current highway. The highway representation will be discussed much more in chapter 3. The environment also stores all information about the entities. Every time an entity moves it uploads it's position to an environment car storage. Every agent has access to the storage, uploads and downloads information in it.

2.2 Agents

The agent represents a logic or brain of the entity. It's task is to gather specific information from the environment storages and based on it to decide what to do next. There are several types of agents. Agents can cooperate with other entities, share some information with each other, or just work with it's own data. It can gather data and decide about it's behaviour every discrete time of the simulation (we will focus on the event based simulation representation in chapter 3) or it can once produce a plan of actions, which will then be applied for a longer period of time while the agent only waits.

In our project we at first implemented a basic agent which was not cooperating with other ones and had to decide about it's behaviour every simulation step. Then we focused on agents that are able to produce plans and we implemented them as cooperative, non-cooperative and learning agents. This work focuses on cooperative agents. Two methods of cooperative control will be discussed in chapter 4.

There is a standard approach to interact with environment. Every agent has it's set of sensors, only through them it can gather information from the environment. It also has it's set of actuators and only trough them it can affect the environment.

Figure 2.1 shows the diagram of data flow in the highway project.

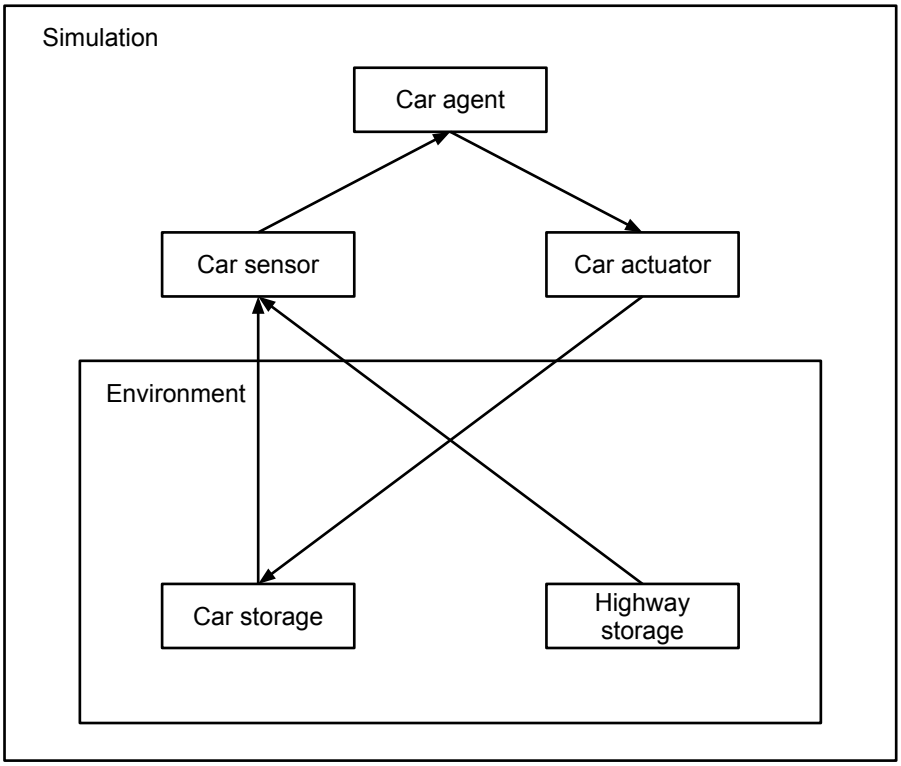

Figure 2.1: Diagram of data upload and download in highway simulation

2.2.1 Sensors

The sensor is an object created by an agent and connected to the environment storages. It contains various methods that are finding data specified by the agent in the storages. The most basic sensors (by sensor we mean the specific method in the Sensor class) returns the position and direction of the entity. The direction is a vector, the position is a point. We will focus on our implementation of the sensors in chapter 3.

2.2.2 Actuators

The actuator is an object also created by an agent and connected to the storages. It contains methods that don't return any value, but change the storage information according to the agent requests. The most basic actions are to change speed and change direction. The actuators don't change the position of an entity, that is done by the storage itself. We will focus on our implementation of the actuators also in chapter 3.

Chapter 3

Representation of the world

In chapter 2 we discussed the general approach to multi-agent system simulations, in this chapter we will focus on the specific implementation of the highway world.

3.1 Alite

The highway project is using a software toolkit developed in ATG center called Alite. The toolkit is described in the Alite project wiki (ATG, 2011*b*):

> Alite is a software toolkit helping with particular implementation steps during construction of multi-agent simulations and multi-agent systems in general. The goals of the toolkit are to provide highly modular, variable, and open set of functionalities defined by clear and simple API. The toolkit does not serve as a pre-designed framework for a complex purpose, it rather associates number of highly refined functional elements, which can be variably combined and extended into a wide spectrum of possible systems.

3.2 Event based simulation

The simulator we used is based on sending and handling events. This is controlled by an environment event processor. The simulation runs in steps called after a specified time period. In each step of the simulation the current positions of the entities are uploaded

to the storages. Also the storage calls various loops, that are used in all the layers (see section 3.5). When everything is done, the storage creates a new step event which will be handled by the storage as another simulation step. The step time is a constant which we set to 100 milliseconds. That means the positions of the cars are changed ten times per second, the cars are moving through discrete positions in distances at most 3.6 meters. Because this distance is lower than a length of a car, the cars can't jump through each other without noticing a crash.

The advantages of this architecture are that we don't have to use threads (whole simulation runs in one thread), also we can create new loops and change the period of time after which the loops or even the simulation steps will be called.

3.3 Highway environment

The environment has two important functions. The first is to provide an event processor. This means every object that wants to create events needs an access to the environment. The second function of the environment is to provide the access to the storages. In our project those are car storage and highway storage. We can get the basic idea of how the simulation architecture works from figure 2.1.

3.3.1 Highway storage

The highway storage is a passive storage. When the simulation is created, the highway structure and all it's parameters are uploaded here from highway editor. The highway structure is composed of points. Between each two points is a line called segment. The highway is composed of straight segments, but the length of each segment could be arbitrarily small. This creates an effect of round curves. The highway representation is shown in figure 3.1. The storage then provides all information about the highway to car entities through their sensors. No data is uploaded to the storage from car actuators or any other sources.

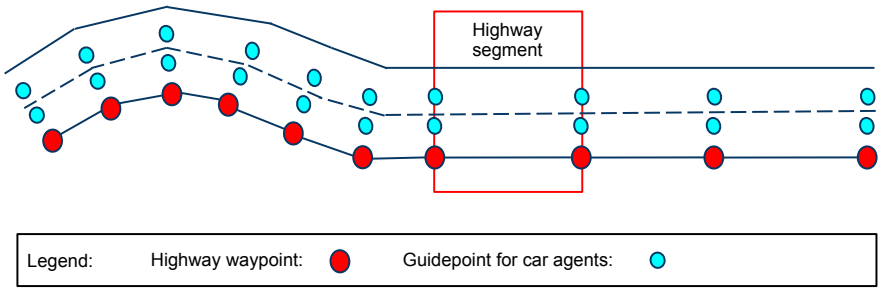

Figure 3.1: Highway structure

3.3.2 Car storage

The car storage manages all data about the entities. Every simulation step it uploads new parameters like cars positions in the world and on the highway, cars directions and speeds. In our project the car storage literally stores the car entities with all it's parameters (we will focus more on car parameters in section 3.4). The storage allows the car agents to get the specific data through the sensors and also to upload new data through the actuators.

3.4 Car entities

Each car entity is a representation of a real car in the simulation world. We have four types of cars in our project (car, van, bus, truck). Each car type has specified parameters (dimension, visibility radius, maximal speed). Every car has it's own car agent with set of sensors and actuators. When a car entity is created, it is stored in the car storage. In our project we can create cars in two ways: we can create all cars when the simulation starts and put them on different positions on the highway or we can continuously create cars during the simulation and put them on the highway start.

3.5 Three layer architecture

The architecture of car control system is divided into three layers. The lowest one is a pedal layer, the center is a way-point layer and the highest is a maneuver layer. Each

layer has it's own interface through which it gets commands from a higher layer and actuators through which it sends commands to the lower layer. To get the idea of the data flow in layers see figure 3.2 from (SCHAEFER, M., 2011). This structure was implemented so the car could create a plan of actions and then execute it continuously during a longer period of time. Because the cooperative, non-cooperative and machine learning control would't be possible without this structure, we will focus on each layer separately with emphasis on the maneuver layer.

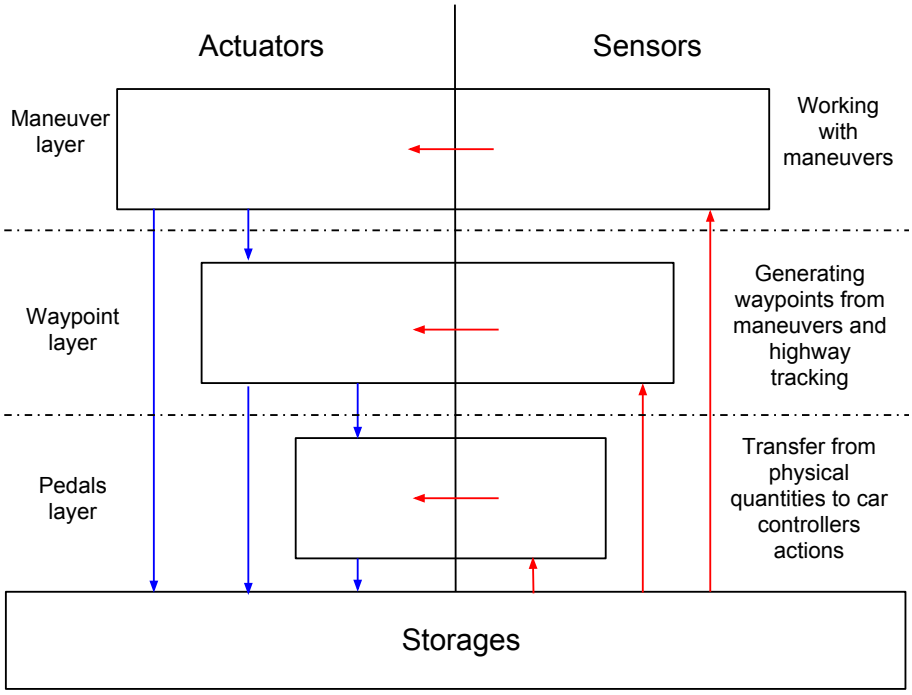

Figure 3.2: Diagram of data flow in layers of each car entity, source: (SCHAEFER, M., 2011)

3.5.1 Maneuver layer

The maneuver layer is responsible for creating a plan of maneuvers which the car will be performing. A maneuver is an object with strictly defined input parameters and calculated output parameters. Parameters of each maneuver are:

- **Input position** is a distance from the highway start.
- **Output position** is calculated according to maneuver input position, speed and acceleration.

- **Input lane** is a number of lane (starting from the right lane).
- **Output lane** is calculated according to a maneuver type.
- **Time** is an absolute time stamp of the maneuver. Output time is calculated as an input time plus time duration of the maneuver, which is a constant set to three seconds. This time period represents a time to change lane on a real highway.
- **Speed** is considered in meters per second. Output speed is relevant to input speed and maneuver acceleration.
- **Acceleration** is each maneuver type constant.

All car maneuvers then differ only in these parameters. We implemented following maneuvers:

- **Straight maneuver** has zero acceleration. The lane and speed are not changed. The car is in a uniform motion.
- **Acceleration maneuver** has defined non-zero acceleration. The lane doesn't change. Each simulation step the car speed is increased by a small increment according to the acceleration. The car is in a uniformly accelerated motion.
- **Deacceleration maneuver** is an equivalent to the acceleration maneuver, the only difference is a defined negative acceleration.
- **Lane left maneuver** is a maneuver that increases car lane by one. This is achieved by using a bezier curve. The acceleration is zero, the speed is not changed.
- **Lane right maneuver** is an equivalent to the lane left maneuver. The only difference is that the output lane number is decreased.

The maneuvers are organized in a queue which represents a plan. In the plan all maneuvers are attached, that means all input parameters of each maneuver have to correspond with the output parameters of the previous maneuver. The approach to building a plan depends on each control method. In most cases the plan is built by searching a state space with an A star algorithm. When a plan is constructed, the maneuver layer starts to send commands to the way-point layer to execute each maneuver.

3.5.2 Way-point layer

The waypoint layer has two objectives. The first one is to generate way-points according to the current maneuver, the second one is to navigate the car towards them. The navigation is done by sending commands to the pedal layer. These commands are to change speed and to change direction of the car.

When a way-point layer gets a maneuver to process it generates way-points according to maneuver input and output parameters. A way-point is an object with specified position (here a position is a point with three coordinates) and time. The way-point layer has to consider the direction and curves of the highway. The way-points are generated with defined time distance. That means the distance between each two way-points depends on the car speed.

When the maneuver is processed, the layer starts to navigate the car towards the way-points. Otherwise (for example when the layer doesn't get any maneuver to process) the layer navigates the car to go straight according to the highway profile. This situation is emergency and should't take too long.

3.5.3 Pedal layer

Pedal layer gets commands to change speed or direction of the car from a way-point layer. It is connected to the simulation physics. The physics contains information about car maximal speed, maximal acceleration and maximal angle the car can turn. The layer sends commands to the physics which checks whether the command doesn't exceed car limits. If the command is verified the physics changes the current parameter of the car.

Chapter 4

Cooperative control

4.1 General characteristics of cooperative drive

Cooperative drive is an approach to multi-agent systems control. The cooperation between the agents has two most important aspects: sharing data and cooperative deciding about agents behaviour.

Sharing the data means each agent has an access to all information about other agents. The agent can see positions of other agents wherever on the highway, their speeds and accelerations, all their sensor inputs and mainly their plans of maneuvers. As we said in section 3.5 each agent creates a plan of maneuvers and then navigates using it. During the creating of a new plan the agent gathers plans of other agents and with this knowledge it tries to create a non-conflicting plan. The approaches to solve the plan creating problem can differ, in this thesis we will describe two of them.

Cooperative deciding will be described in the following section.

4.2 Cooperative planning

Cooperative deciding and planning means that when an agent creates a plan containing time and position conflicts with plans of other agents, it starts to cooperate with them and more agents then try to solve the conflicts of the first agent. There are also many approaches to solve agents cooperating, in the *cooperative basic* method there is just a basic cooperation, in *cooperative peer to peer* method the agents cooperate iteratively in pairs.

4.2.1 Planning with A star algorithm

In both cooperative methods the agent is checking it's plan every simulation step. When the plan is empty, it starts a search of an optimized path using A star algorithm. The A star algorithm is browsing the state space of the agent to find an optimized path.

The state space is composed of nodes and edges. A node represents a state of the agent. In highway project it is an object with defined position, lane, time and speed. An edge is a maneuver (for information about maneuvers see section 3.5). The A star algorithm browses the nodes and expands them, that means it creates all possible maneuvers starting in the current node.

Two important parameters of the A star search algorithm are: a cost of an edge (maneuver) and a heuristic distance of a node.
The cost of an edge is a number which should represent a length or a difficulty of the edge. Each node then has a cost composed of costs of each edge that lies on the path from the start node to the current node. The path-cost function we used is shown in table 4.1. The heuristic distance of a node is an estimated distance from the node to the goal node. This function must be an admissible heuristic, that means it must not overestimate the distance from the current node to the goal node. The heuristic function we used is shown in table 4.1.
A total cost of each node is then a sum of the path-cost and the heuristic distance. The A star algorithm always expands the node with the lowest total cost.

A star cost function	difference between the maneuver input and output time
A star heuristic distance	time to achieve the goal node when driving at maximal speed

Table 4.1: Parameters of the A star search algorithm in highway project

4.2.2 Creating conflicts

When the algorithm expands a node, it checks each new maneuver if it creates conflicts. Each cooperative method works with the conflicts differently, therefore we will focus on the conflicts resolving during the methods description. In highway project there are three types of conflicts:

- **Conflict with highway** means the maneuver leads out of the highway. This happens when the maneuver output lane is lower than zero or the output lane is closed in the current segment.

- **Conflict with maximal speed** appears when the maneuver output speed is higher than the maximal speed of the current car type.

- **Conflict with plan of other car** appears when the maneuver has time and position intersection with maneuver of another car. Both of following conditions must be met:

  Time intersection means the maneuvers share time.

  Position intesection means the maneuvers intersect in some space. It is measured with one meter accuracy.

  This type of conflict is represented by an object with specified time of the conflict, names of both cars and their conflicting maneuvers.

4.2.3 A star limits

Because the A star algorithm can in some situations be very time-consuming, we used two parameters to limit the state space: open list size limit and depth limit. The open list is a list containing all nodes that the algorithm has expanded to and is going to expand. Especially when the car speed is low or the highway is long, it's size can be huge, which leads to a long processing time. This is the reason why we added a limit to the size of the open list. For the same reason we added a depth limit, which says how deep the algorithm can search. If any of the limits is exceeded, the algorithm stops and returns the path to the last node. This approach finds an optimized plan which doesn't lead to the end of the highway. This plan is not globally optimal, but it is acceptable, because we are certain that the rest of the plan exists.

4.3 Cooperative basic method

4.3.1 General description of cooperative basic method

The cooperative basic is a method which we developed and implemented. It's biggest difference with the peer to peer method is that the agents are cooperating only in a state of emergency and their cooperation is very limited. Sharing plans works similarly to the peer to peer algorithm. The control method is working, but it has an unresolved problem with deadlocks. In a situation when two cars are driving as close as the maneuvers allow them none of the car agents can create a new plan. We will focus on this problem further.

4.3.2 Cooperative basic planning

The agent creates a plan only when it's plan is empty. This happens always on the highway start and then sometimes when the A star algorithm was unable to create a plan which would lead to the highway end. The agent gathers through it's sensors all data needed to represent it's current state by a start node. From this start node the A star algorithm starts to browse the state space and finds the path. When each node is expanded, the algorithm checks each new maneuver. If a maneuver creates a conflict with the highway, maximal speed or plan of other car, it is banned and the node it leads to is not added to the list of opened nodes.

We can see the situation in figure 4.1. Car 1 and Car 2 have already created their plans (black dash-dotted arrows), Car 3 just appeared on the highway and now begins to create a new plan. It creates it's start node and the A star algorithm starts to expand the nodes. To simplify the situation, we assume the Car 3 can perform only three maneuvers: lane left, straight and lane right. Whenever the algorithm finds a conflict in a maneuver, the maneuver is banned (red dotted arrows). The algorithm then creates a set of non-conflicting maneuvers (green dashed and blue arrows). When the algorithm stops (the goal node was achieved or the limits were exceeded, see section 4.2.3) it returns the path to the last node (blue arrows). The situation with Car 1 shows that the maneuvers can intersect in position if they don't intersect in time too.

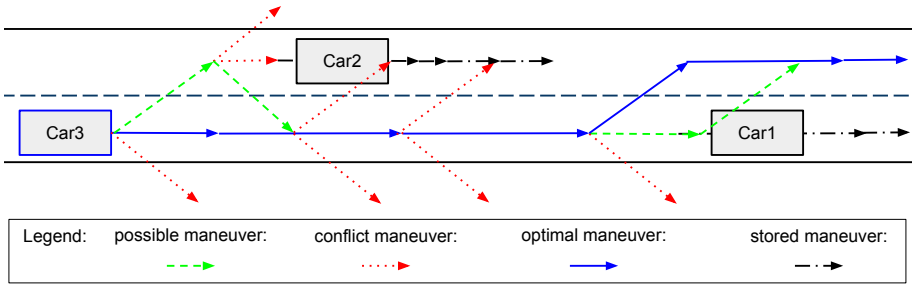

Figure 4.1: Cooperative basic planning

4.3.3 Iterative state space depth contraction

The A star algorithm used in this method is using only non-conflicting maneuvers. The highway state space is limited by the highway borders, which means the A star is sometimes not able to create a plan to the highway end. The path doesn't have to exist. We resolved this situation using iterative state space depth contraction. When the A star fails to create a path, the depth limit is reduced by half. This can go on until the depth limit is one, which means the algorithm finds and returns only one maneuver.

4.3.4 Cooperation on resolving conflicts

In this method we only used very limited cooperation. The agents cooperate only in a state of emergency, which occurs when an agent is unable to create a plan. In that situation the agent finds the nearest conflicting car and erases it's plan. Then the agent is able to create a plan and in another simulation step the other agent tries to create it's plan. When this algorithm fails, it repeats again.

4.3.5 Problems of the cooperative basic method

The main problem of the cooperative basic method is an existence of deadlocks. The A star algorithm makes the cars go as fast as they can. This creates situations where the cars are going as close as they can. The A star planned their path so they wouldn't crash, but because it didn't plan their path to the highway end, one of the car has to create a new plan. The situation is depicted in figure 4.2. Car 2 is so close to the planned

maneuver of Car 1 that it can't create any plan. All maneuvers that Car 2 creates either lead out of the highway or create a conflict zone, which is a part of the highway space locked by more than one maneuver. The lane right maneuver of Car 2 creates conflict with the straight maneuver of Car 1 because the change-lane maneuvers lock both used lanes. We tried to use a safe deacceleration maneuver, which would be created in this situation no matter the conflicts, but we didn't manage to resolve the problem.

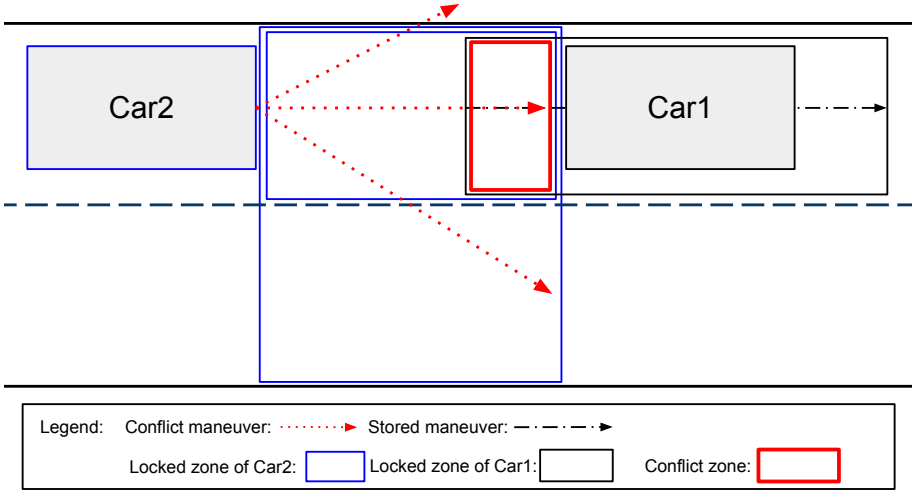

Figure 4.2: Cooperative basic deadlock situation

4.4 Cooperative peer to peer method

4.4.1 General description of cooperative peer to peer method

The peer to peer algorithm was developed in ATG center and described in (Volf, P. et al., 2007):

> Algorithm starts with selecting the soonest conflict. It generates a set of possible flight trajectories for each airplane using predefined manoeuvres (e.g. turn left, turn right, turn up, turn down, speed up, slow down). This manoeuvres are constructed to avoid the conflict. Each generated flight trajectory is tested for conflict with all other airplanes. If a collision with any airplane is found that would happen sooner than the currently solved one, such trajectory is removed from the generated set. Afterwards, sets for both airplanes are com-

bined to create all possible pairs of trajectories (cartesian product of these sets). Each pair is checked whether mutual collision persists. If there are some valid pairs, the best one is selected. Such selection can be based for example on sum of the utility value of each trajectory to reach maximum social welfare. If there is not any valid pair, the manoeuvres are applied again to generate wider range of trajectories. Newly generated and checked flight plans are added to the previously generated set. This process is repeated until a valid pair of generated trajectories is found and applied. When single collision is solved, the next soonest collision is selected.

4.4.2 Peer to peer algorithm in the highway environment

The described algorithm can be used on the highway with some changes. The most important difference between the highway environment and the environment for airplanes is a size of the state space. The airplanes are not limited by the environment in any way, they can even use three dimensional maneuvers. The cars are limited by the highway borders and by the highway lanes. This leads to a situation when the peer to peer algorithm doesn't work. This situation and a solution will be described in section 4.4.5. In the following sections we will describe the specific use and implementation of the algorithm in the highway project and compare it with the cooperative basic method. The diagram of the peer to peer algorithm used in highway simulation is shown in figure 4.5.

4.4.3 Cooperative peer to peer planning

The agent creates a plan when it's plan is empty and also much more frequently when it is resolving a conflict. We will focus on the resolving of conflicts in the following section. As in the cooperative basic method the agent gathers all the information it needs to create a start node which represents it's current state and starts an A star search. The use of the A star algorithm in the highway project is very variable. In peer to peer algorithm it allows all maneuvers, that don't lead out of the highway or don't exceed the car maximal speed. When a maneuver creates a conflict with a plan of another car, the conflict is stored in a car storage and the maneuver is allowed. This means that with each created plan the algorithm creates a set of conflicts sorted by the conflict time. In this first step of the peer to peer algorithm the agents plan an optimized path, which contains conflicts.

The situation is depicted in figure 4.3. While Car 3 is creating a plan, it stores all conflicts, which are filtered after the path is created, so only the conflicts of maneuvers in the final path remain. In this situation the conflict with Car 1 remains in the conflict list of Car 3. For information about conflicts see section 4.2.2.

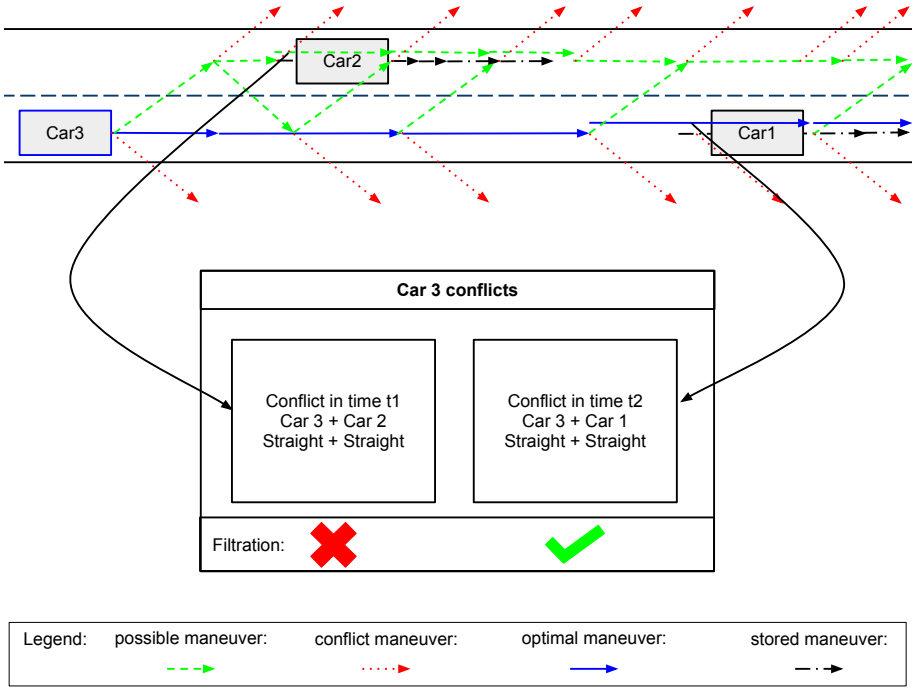

Figure 4.3: Cooperative peer to peer planning

4.4.4 Cooperation on resolving conflicts

We implemented the algorithm according to the article (Volf, P. et al., 2007). When an agent creates a plan, it finds a soonest conflict in it's conflict list. Then it divides the plan into a part before the conflict, which will remain, and a part after the conflict, which will be altered. From the last node of the path before the conflict we create all possible maneuvers. The output parameters of these maneuvers are used to construct start nodes of an A star algorithm, which then plans paths using the same approach as described in the previous section. This way we create a set of possible plans for the current agent. The same procedure is used to create a set of possible plans of the conflicting car.

For these sets we apply a cartesian product. That means we create pairs of possible plans of both cars. These pairs are tested for a presence of a conflict with lower conflict

time than a time of the conflict we are currently trying to resolve. If this conflict is found, the pair is banned. Otherwise the pair is evaluated. The value is represented by a time period to another conflict.

The algorithm then chooses the pair with a highest value and connects each plan to the parts of the old plan before the conflict.

This approach guarantees that all conflicts created during the run of this one step will have higher conflict time. In another step the agent will solve other soonest conflict an so on until the agent has no stored conflict.

We described the pure cooperative peer to peer algorithm, which works well in the AgentFly project where it creates plans of airplanes. We can use it in the highway project with one change which is described in the following section.

The diagram of the peer to peer algorithm is shown in figure 4.5.

4.4.5 Safe maneuver

The approach to avoid a conflict for the agents is to either change lane or change speed. Let's assume that none of the two conflicting cars can change lane (it would create a conflict which would happen sooner then the current one). Then they have to change their speeds, which ideally means the first car will accelerate and the second car will slow down. The first car (we mean the car in front of the second car) often can't accelerate because it is going it's highest speed. This means the conflict has to be solved by the second car. But the second car doesn't have to be able to resolve the conflict. The second agent creates a deacceleration maneuver from the last non-conflicting node of it's plan, but this maneuver can create conflict with the same conflict time as the time of the conflict it is trying to resolve. When this happens, the original peer to peer algorithm fails.

The solution to this problem is a safe maneuver. When a pair of two conflicting cars isn't able to resolve the conflict, the second agent (the one which is back and has lower driven distance) creates a safe deaccelerating maneuver starting from the current car state. Then the agent creates a plan starting with the safe maneuver, using an A star algorithm. Then the peer to peer algorithm starts again. If it is still unable to resolve the soonest conflict (which is different from the one before the application of the safe maneuver), the agent adds a second safe maneuver after the first one and creates the plan again with the A star algorithm. This process can go on and on until a non-conflicting plan is created.

The situation is depicted in figure 4.4. Car 2 is trying to create a non-conflicting plan, which it can create only by applying a safe maneuver. The whole peer to peer algorithm with the safe maneuver application is shown in figure 4.5.

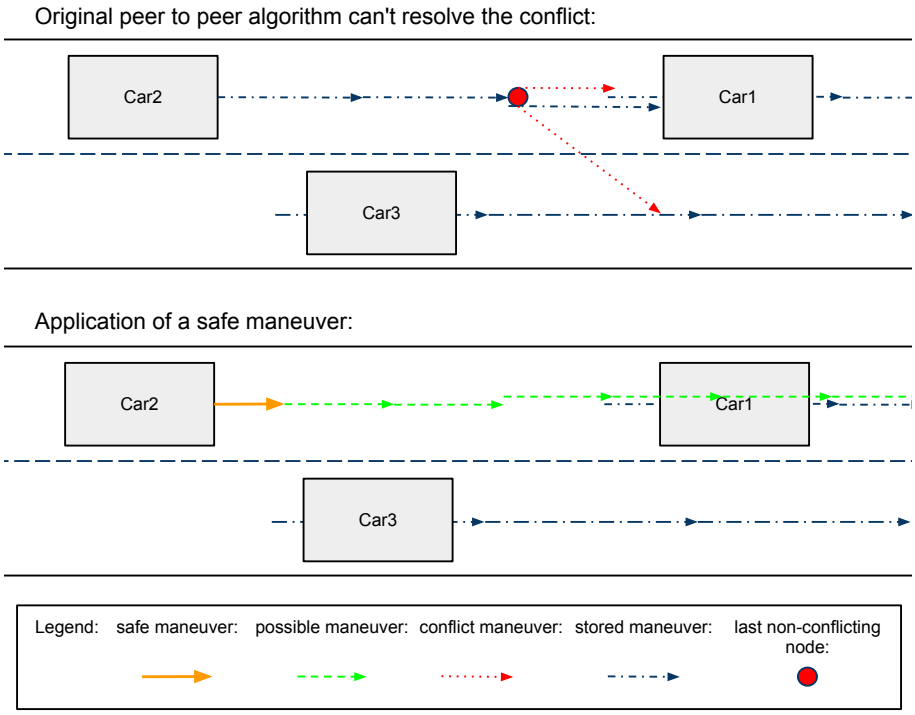

Figure 4.4: Cooperative peer to peer - safe maneuver application

4.4.6 Problems of the cooperative peer to peer method

The cooperative peer to peer algorithm with an application of a safe maneuver should theoretically work without any problems. Even so we encountered problems caused by a limited state space. The resolution of the conflicts is based on the time and space separation of whole maneuvers. Each maneuver takes three seconds and during that time the space of the maneuver is locked for other cars. This combined with a limited space of the highway creates the biggest difference to the environment of the airplanes and also brings the biggest problem to resolving the conflicts. There are occasionally situations when an agent is unable to create a non-conflicting plan.

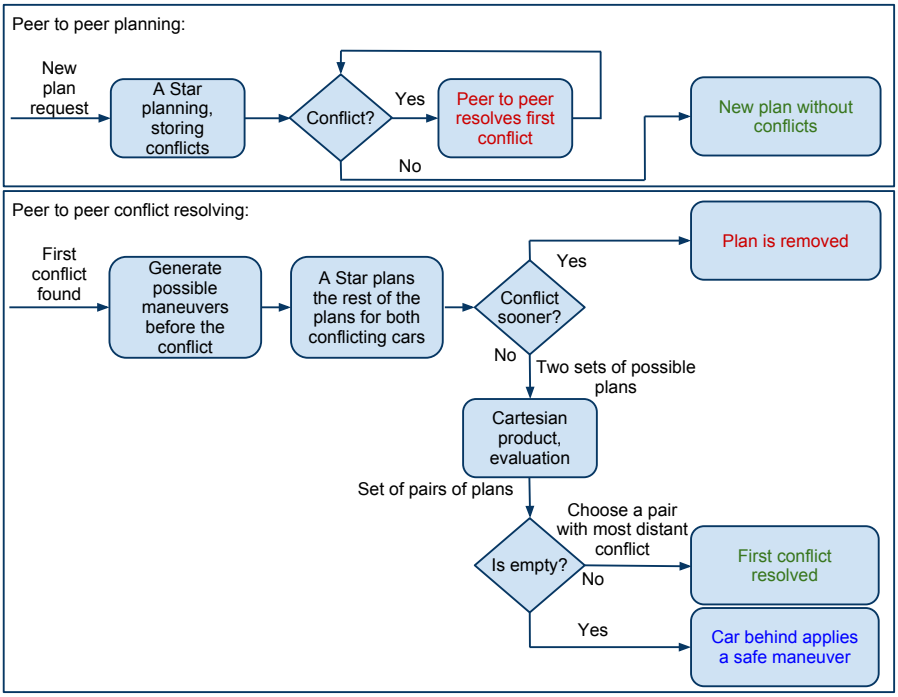

Figure 4.5: Cooperative peer to peer algorithm diagram

4.5 Cooperative algorithms comparison

We have developed cooperative basic method that is not using all the benefits of the cooperative drive. The cooperation works mainly based on sharing the plans. Peer-to-peer algorithm originally developed for airplanes by (Volf, P. et al., 2007) has been adopted and extended by safe maneuver to ensure proper deconfliction in our scenario. In the simulation the peer to peer algorithm works much better, because it uses much more benefits of the cooperative drive and also because the basic method has unresolved problems with deadlocks. For information about the cooperative basic method see section 4.3, the cooperative peer to peer method is described in section 4.4. Table 4.2 shows main differences between the two cooperative methods.

	Cooperative basic	**Cooperative peer to peer**
Sharing	Plans of maneuvers	Plans of maneuvers
Cooperating	in emergency, limited	full cooperation iteratively in pairs
A* planning	creates only maneuvers without conflicts with plans of other agents	creates all maneuvers without environmental conflicts, stores conflicts with plans of other agents
Conflict with environment detection	conflicting maneuver leads of the highway or exceeds maximal speed of the car	conflicting maneuver leads of the highway or exceeds maximal speed of the car
Conflict with other cars plans detection	doesn't create maneuver conflicting with plans of other agents	creates all maneuvers, stores conflicts with plans of other agents
A* failure	iterative state space depth contraction	cannot happen

Table 4.2: Comparison of parameters of cooperative algorithms

Chapter 5

Related control methods

In this work we describe implementation of cooperative drive - one of the three approaches to the multi-agent control of the cars (for more information about objectives definition see the introduction and figure 1.1). We focused on cooperative control in chapter 4. In this chapter we will very shortly discuss the other two approaches (mainly because we will compare the test results of those methods, non-cooperative control and machine learning are not my objectives in highway project or in this thesis). For comparison of test results of all implemented methods see chapter 6.

5.1 Non-cooperative control

The non-cooperative control does not plan sequence of maneuvers but only one next maneuver. New maneuver is accepted when the situation it leads to is safe, which matters only on the actual situation. The maneuvers are created according to preferences, which makes the agents go as fast as possible.

Agents can see directional and brake lights of other cars, they also estimate the speed of other cars. With these inputs the agent creates a virtual plan of the other cars.

The non-cooperative drive is closer to a real highway traffic than a cooperative drive. Because the agents using cooperative drive can use much more information and actions, which comes with the cooperation, it should achieve better results in the simulation. On the other hand the advantage of the non-cooperative drive could be an independence on the communication between the cars. The control will work even when the communication fails.

We can see the non-cooperative planning algorithm in figure 5.1, which was taken from (Schaefer, M., 2011). There is also much more information about the non-cooperative methods used in the highway project.

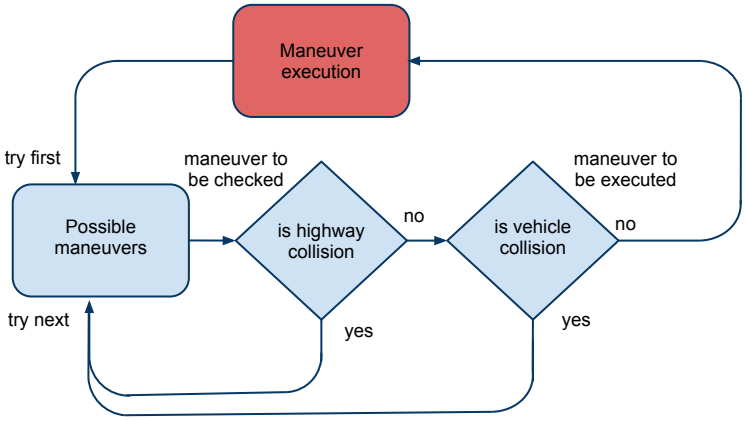

Figure 5.1: Diagram of non-cooperative planning process, source: (Schaefer, M., 2011)

5.2 Machine learning

The machine learning uses a completely different approach than the cooperative or non-cooperative drive. The cars are in various states, which are represented by sets of parameters. Most important parameters describing each state are the speed, cars in the surroundings and a shape of the highway in the current position. These parameters can only have a value of true or false. Each state is evaluated according to the current car speed after the car enters it. Also a state where a car is crashed is evaluated as the worst.

In the beginning of the learning process the agents have no clue how they should behave. As the simulation continues, agents start to upload new values to each state they have been to and then they prefer the states with higher values. This approach should theoretically lead to a learned system of agents, that prefer non-conflicting maneuvers with the highest possible speed.

For much more information about machine learning methods used in the highway project see (Franěk, A., 2011).

Chapter 6

Measurements, tests

In this chapter we will define the parameters of the simulation. We will compare the two cooperative methods using several tests. Then we will choose the cooperative method with the better results and compare it with the results of the related control methods.

6.1 Definition of the input and output parameters

There are many approaches to measure various parameters of the highway simulation. We have to exactly define the input and output parameters of the simulation.

The input parameters of the measurements are:

- **Simulation time** - the total time of the simulation
- **Initial number of lanes** - number of lanes before the narrowing
- **Final number of lanes** - number of lanes after the narrowing
- **Length before narrowing** - length of the highway in meters before the narrowing
- **Length after narrowing** - length of the highway in meters after the narrowing
- **Car types ratio** - ratio of the occurrence of all car types (car, van, bus, truck)
- **Delta time** - the time interval after which the cars are created and inserted to the highway beginning

We choose to define the output parameters according to (VOKŘÍNEK, J. et al., 2009), we added the last two. The output parameters of the measurements are:

- **Number of finished cars** - the number of cars that successfully reached the end of the highway
- **Number of crashed cars** - the number of cars that either came off the highway or crashed with other car
- **Average speed** - the average speed of all finished cars
- **Numbers of each maneuver** - numbers of straight, accelerating, deaccelerating, turning left and turning right maneuvers
- **Number of sent messages** - the number of cooperation between the pairs of cars in cooperative drive
- **Number of A Star operations** - the total number of all A Star runs during a simulation

6.2 General tests definitions

In the two sets of tests some input parameters will be similar. It will be mainly the shape of the highway and the ratio of the car types. The basic scenario is a narrowing. We decided to test two types of narrowings that are common on real highways, narrowing from three to two lanes (Test 2) and narrowing from two to one lane (Test 3). We also tested the control without narrowing on highway with two lanes (Test 1). The car types ratio also represents a real ratio of car types on the highways. The general parameters are shown in table 6.1. Figure 6.1 shows the pictures of the simulation.

Input parameters	**Test 1**	**Test 2**	**Test 3**
Initial lanes number	2	3	2
Final lanes number	2	2	1
Length before narrowing	1500 m	1000 m	1000 m
Length after narrowing	0 m	500 m	500 m
Car types ratio (car:van:bus:truck)	(5:1:1:1)	(5:1:1:1)	(5:1:1:1)
Delta time	6 s	6 s	6 s

Table 6.1: General input parameters of the measurements

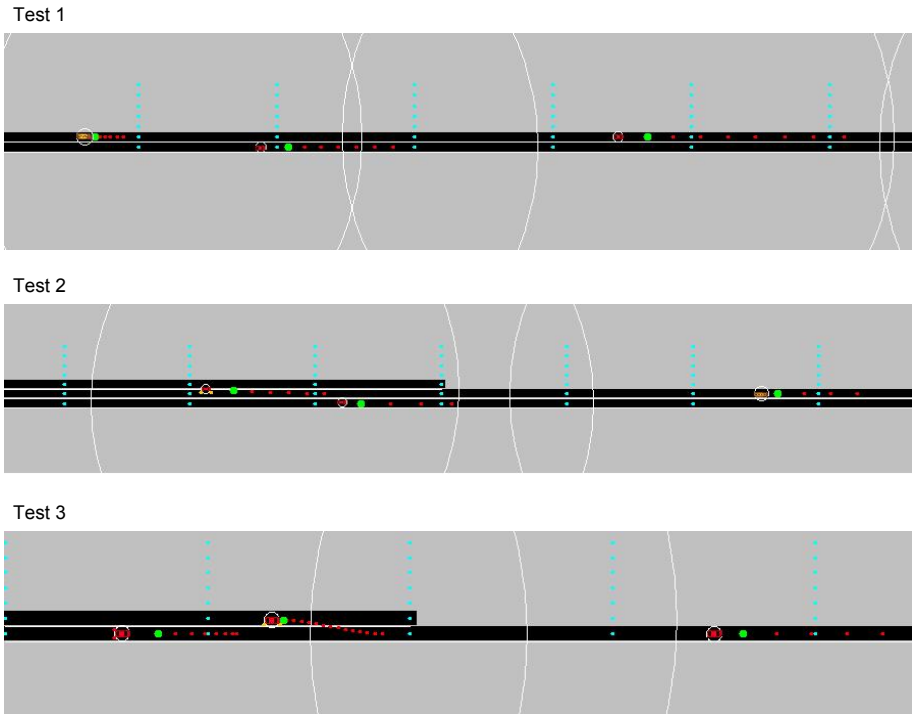

Figure 6.1: Tests definitions

6.3 Cooperative methods comparison

We will measure the cooperative basic and cooperative peer to peer methods output parameters. We described this two methods in chapter 4.

6.3.1 Specific tests definitions

The general test definitions were described in table 6.1. The specific parameter for this set of measurements is a simulation time, which will be 5 minutes.

As we said in section 4.2.3, the A Star search algorithm has several parameters. Values of these parameters differ according to the current cooperative method. The A Star parameters for the two cooperative methods are shown in table 6.2. The depth limit is set to 10, which means the cars are planning 30 seconds into the future.

A Star parameters	**Cooperative basic**	**Cooperative peer to peer**
Open list size limit	2000	1500
Depth limit	10	10

Table 6.2: Parameters of the A Star algorithm for cooperative methods

6.3.2 Results

The results of both methods measurements are shown in table 6.3 (Test 1), table 6.4 (Test 2). We will not measure Test 3, because the basic method is not able to ensure a free highway without crashes. Using Test 1 we also measured other parameters of the control with various traffic density. These are: average speed, number of sent messages and number of A Star operations. We can see outputs of these measurements in figure 6.2, figure 6.3 and figure 6.4. An evaluation follows.

Test 1	**Basic**	**Peer to peer**
Number of finished cars	32.5	34.8
Number of crashed cars	8.25	0
Average speed	86.25 km/h	81.6 km/h
Number of straight maneuvers	618.25	569
Number of acceleration maneuvers	326.75	572.4
Number of deacceleration maneuvers	57	121.8
Number of lane left maneuvers	15.25	0
Number of lane right maneuvers	17.75	0
Number of sent messages	139.25	154
Number of A Star operations	1146.5	1456.2

Table 6.3: Results of Test 1 for cooperative basic and peer to peer methods, average from 5 measurements

Test 2	**Basic**	**Peer to peer**
Number of finished cars	38.25	36
Number of crashed cars	4	0
Average speed	99.75 km/h	76.2 km/h
Number of straight maneuvers	730.75	564
Number of acceleration maneuvers	353.5	716
Number of deacceleration maneuvers	47.25	173.8
Number of lane left maneuvers	14.5	0
Number of lane right maneuvers	43.75	16.2
Number of sent messages	168.25	225.8
Number of A Star operations	1417	1981.2

Table 6.4: Results of Test 2 for cooperative basic and peer to peer methods, average from 5 measurements

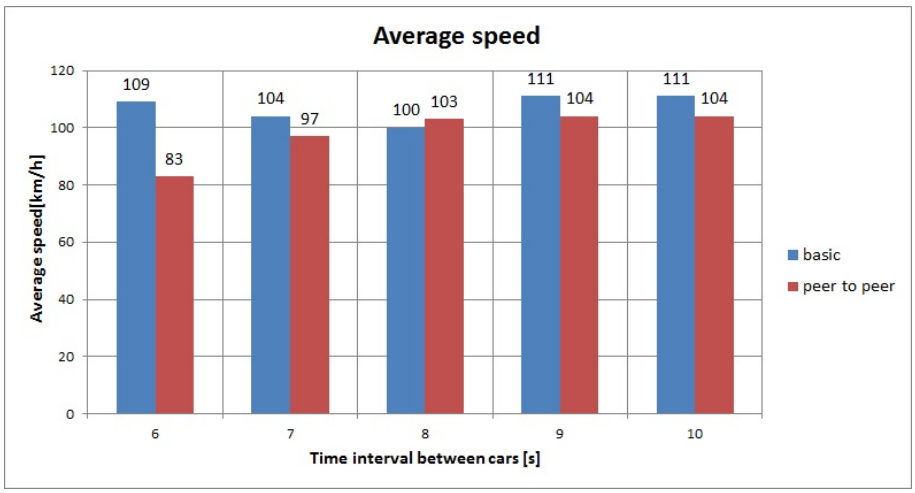

Figure 6.2: Diagram of average speed in cooperative basic and peer to peer methods for various traffic density

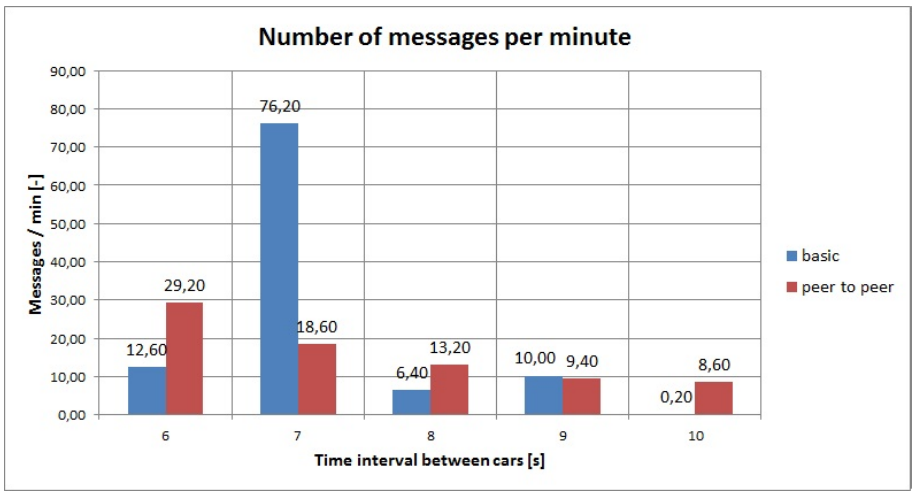

Figure 6.3: Diagram of message communication in cooperative basic and peer to peer methods for various traffic density

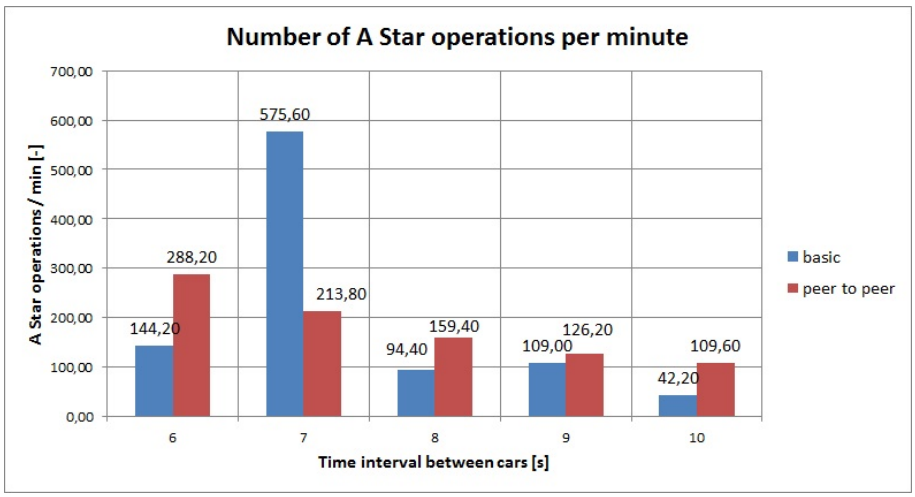

Figure 6.4: Diagram of A Star planning count in cooperative basic and peer to peer methods for various traffic density

6.3.3 Tests evaluation

The most important output parameter is the number of crashed cars. The peer to peer method ensured a non-conflicting simulation. The cooperative basic method wasn't able to ensure a non-conflicting simulation. The reason was described in section 4.3.5.

The second most important parameter is the average speed. The basic method was better than the peer to peer method, but the price for that were the crashes. The average speed is generally quite low, because the cars are placed on the highway with an initial speed, which is half the maximum speed. We can see the average speed with various traffic density in figure 6.2.

The number of finished cars depends a lot on the average speed. In these two tests the numbers were almost similar.

When we compare the numbers of each maneuver, we can see that the peer to peer method creates much more acceleration and deacceleration maneuvers, the control is more dynamic. On the other hand the basic method creates much more lane left and lane right maneuvers, this is because the control creates only non-conflictning plans.

Figure 6.3 shows numbers of separate communications between the cars. We can see that the peer to peer algorithm is using the communication more often. The peer to peer communication and conflicts resolution is much more complex, which we can see in figure 6.4. With one exception (probably an occurrence of a deadlock) the peer to peer

algorithm uses the A Star to create a plan much more often.

In general the cooperative basic method is in this version not applicable to a traffic control, the peer to peer method has optimistic results.

6.4 Cooperative and related control methods comparison

We will compare the cooperative peer to peer method with non-cooperative and machine learning methods. Peer to peer algorithm was described in section 4.4, non-cooperative and machine learning methods were described in chapter 5.

6.4.1 Specific tests definitions

The general test definitions were described in table 6.1. The A Star parameters for peer to peer algorithm are the same as those shown in table 6.2. The specific parameter for this set of measurements is the simulation time, which will be 10 minutes.

6.4.2 Results

The results of all methods measurements are shown in table 6.5 (Test 1), table 6.6 (Test 2) and table 6.7 (Test 3). The figure 6.5 shows total average speed for various traffic density for all control methods, figure 6.6 and figure 6.7 show the dynamic of control of all methods. All three diagrams are measured using Test 2. An evaluation of all tests follows.

Test 1	**Cooperative**	**Non-cooperative**	**Machine learning**
Number of finished cars	78.6	92.8	70.4
Number of crashed cars	0	0	0.4
Average speed	84.6 km/h	112.2 km/h	68.2 km/h
Number of straight maneuvers	1209	66	468.2
Number of acceleration maneuvers	1208	1487.8	1225
Number of deacceleration maneuvers	250.8	51.2	431.8
Number of lane left maneuvers	0.4	12.2	75.2
Number of lane right maneuvers	0	16.6	71.8

Table 6.5: Results of Test 1 for cooperative, non-cooperative and machine learning methods, average from 5 measurements

Test 2	**Cooperative**	**Non-cooperative**	**Machine learning**
Number of finished cars	79.2	86	45.2
Number of crashed cars	0	0	1.2
Average speed	77 km/h	108 km/h	40 km/h
Number of straight maneuvers	1164.2	268.2	533.6
Number of acceleration maneuvers	1534.4	1440.4	1084.4
Number of deacceleration maneuvers	384.2	533.2	1060.6
Number of lane left maneuvers	0	3.8	117.2
Number of right lane maneuvers	40.4	41	132.2

Table 6.6: Results of Test 2 for cooperative, non-cooperative and machine learning methods, average from 5 measurements

Test 3	Cooperative	Non-cooperative	Machine learning
Number of finished cars	48.2	67.6	43.4
Number of crashed cars	0.4	0	0
Average speed	62 km/h	79.6 km/h	49.5 km/h
Number of straight maneuvers	657.8	1149	266
Number of acceleration maneuvers	1265.4	1085.6	898.8
Number of deacceleration maneuvers	401	1372.6	403
Number of lane left maneuvers	0.6	5.4	83
Number of lane right maneuvers	34.4	28.2	104.2

Table 6.7: Results of Test 3 for cooperative, non-cooperative and machine learning methods, average from 5 measurements

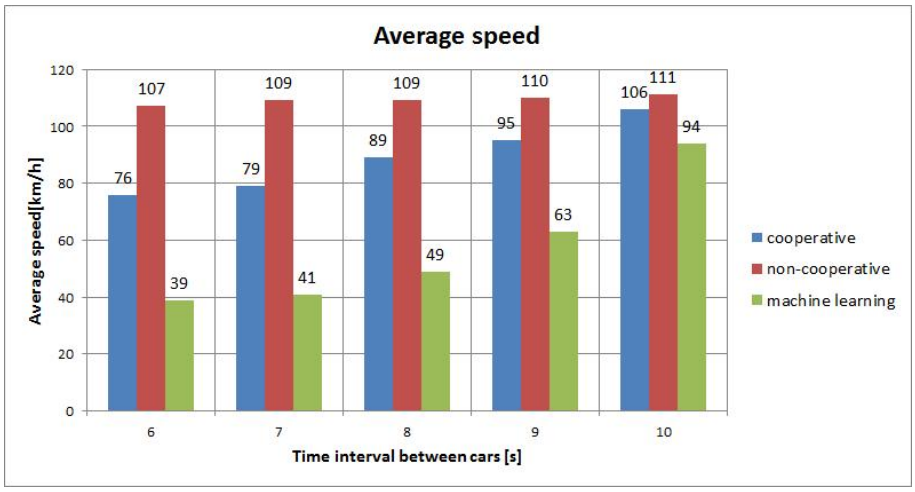

Figure 6.5: Diagram of average speed for cooperative, non-cooperative and machine learning methods for various traffic density

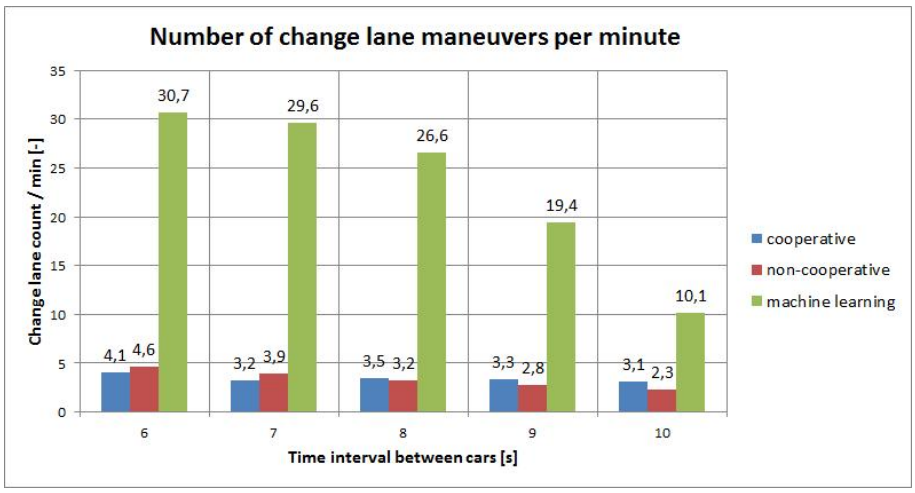

Figure 6.6: Diagram of change lane maneuvers for cooperative, non-cooperative and machine learning methods for various traffic density

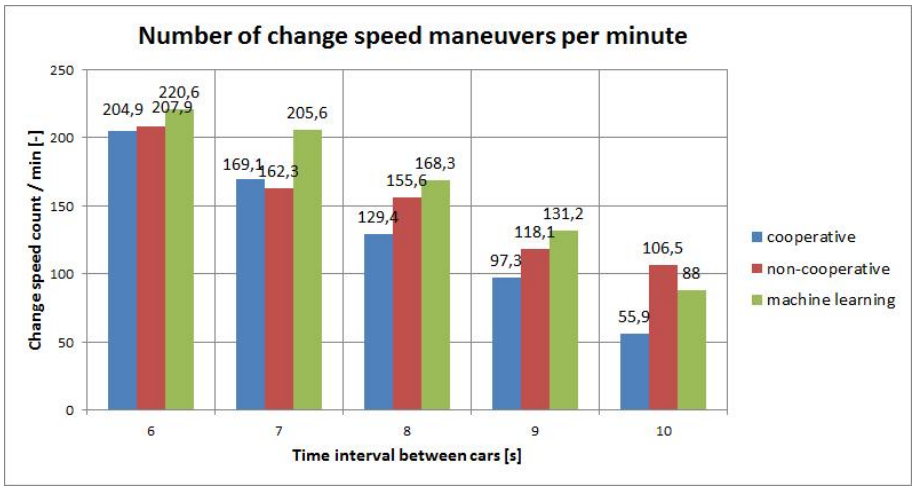

Figure 6.7: Diagram of change speed maneuvers for cooperative, non-cooperative and machine learning methods for various traffic density

6.4.3 Tests evaluation

We will focus mainly on the comparison of cooperative and non-cooperative methods. We can compare them using the number of crashed cars, number of of finished cars, average

speed and control dynamics. We can also get interesting conclusions from observing the simulations.

With one exception both methods are able to ensure a simulation without crashes. The exception happened during the cooperative drive in the beginning of the highway, a new car applied a safe maneuver, during which the other cars are not allowed to cooperate with it (This condition prevents an occurrence of deadlocks in peer to peer planning). In further work this problem could be solved.

The average speed is always bigger for the non-cooperative drive. This can have various reasons. The cooperative peer to peer algorithm uses and limits the state space more, also a frequent use of a safe maneuver (see section 4.4.5) brings a relatively intensive speed reduction. Also the non-cooperative drive doesn't deliver all the cars, it gives more space to cars in the lane which is not ending. That increases their speed. This three reasons probably lead to a fact, that the cooperative control can deliver less cars than the non-cooperative. We can see the difference in average speed in figure 6.5. As we said before, the average speed is generally low, because the cars are placed on the highway with an initial speed, which is half the maximum speed.

The number of finished cars is significantly dependant on the average speed. It is lower for the cooperative control.

Figure 6.6 and figure 6.7 show the dynamic of control. We can see that the number of change lane maneuvers is almost similar, the non-cooperative drive is more dynamic in change speed maneuvers, the reason is the basic principle of cooperative and non-cooperative control.

A big advantage of a cooperative control is a fact, that it delivers all the cars. The problem of the non-cooperative drive is that often the cars in the ending lane stop before the narrowing and are not able to turn right and get behind the narrowing. This fact can be observed from the simulation. It also means the cars in the lane that continues don't have to slow down because of the cars in the ending lane, which in the end increases the average speed of all cars. To get more information about the non-cooperative simulation see (Schaefer, M., 2011).

The machine learning results are slightly less satisfying. The reason is a fact that the learning methods used in a traffic control were implemented and examined for the first time. It will need more further work to achieve similar results as the cooperative or non-cooperative control. The numbers of machine learning change lane maneuvers are higher than for other methods. The reason is in the learning process and is not certain. For more information about machine learning methods see (Franěk, A., 2011).

Chapter 7

Conclusion

In the first part of the highway project we implemented an event based simulator. We created an environment containing storages, which store information about all the agents and entities and also the highway data. We also implemented a set of sensors and actuators, which the agents are using to orientate themselves and to move in the environment. Then we created a three layer structure of sensors and actuators with a defined application programming interface. The highest layer we prepared for connection to cooperative, non-cooperative and machine learning planners.

The main achievement of this work is implementation of two cooperative planning algorithms: cooperative basic method and cooperative peer to peer method. The basic method was developed by me and has several advantages, but it also has an unresolved problem with deadlocks. The peer to peer method was created originally for airplanes and we had to modify it by applying a safe maneuver.

The further work on the cooperative drive could be an optimization of the A Star search algorithm, which is used very often and makes the peer to peer algorithm very time-consuming. Also the A Star algorithm and generally the system of maneuvers uses a set of parameters, which can affect the results of the simulations. More research and specification of these parameters could lead to a more optimized and more effective algorithm.

The highway simulator we designed and implemented can be used for many other planners using various control methods. Also various physical models can be connected and used instead of our simple physics.

Although the implemented algorithms aren't able to ensure a simulation without crashes under all circumstances (during thirty simulations of peer to peer algorithm there was one crash), they have many advantages over a centralized model of a highway system.

When used on real highways, a control units of each car wouldn't be dependent on any central control unit, only on a short range communication with other cars. Data sharing is becoming a trend of this time and thanks to that the cooperative control methods could become a major topic in the real traffic control.

Bibliography

ATG (2011*a*), 'Agentfly project page'. ⟨http://agents.felk.cvut.cz/projects/agentfly/⟩.

ATG (2011*b*), 'Alite project page'. ⟨http://merle.felk.cvut.cz/redmine/projects/alite/wiki⟩.

Franěk, A. (2011), 'Machine learning methods for colision avoidance of road vehicles'.

Kubík, A. (2004), *Inteligentní agenty*, Brno: nakladatelství Computer press. ISBN 80-01-03208-6.

Schaefer, M. (2011), 'Noncooperative colision avoidance of road vehicles'.

Vokřínek, J., Jiránek, J. and Hodík, J. (2009), 'Agent-based highway traffic simulation'.

Volf, P., Šišlák, D., Pěchouček, M. and Prokopová, M. (2007), 'Convergence of peer-to-peer collision avoidance among unmanned aerial vehicles'. ISBN 80-01-02095-9.

Appendix A

Source codes

The highway 2 project source codes are enclosed on the CD. The implemented cooperative methods are stored in following directory structure:

- highway2/classes/cz/agents/highway2/planner/cooperative/:
  cooperative basic, cooperative peer to peer methods
- highway2/classes/cz/agents/highway2/planner/plan/:
  cooperative plan