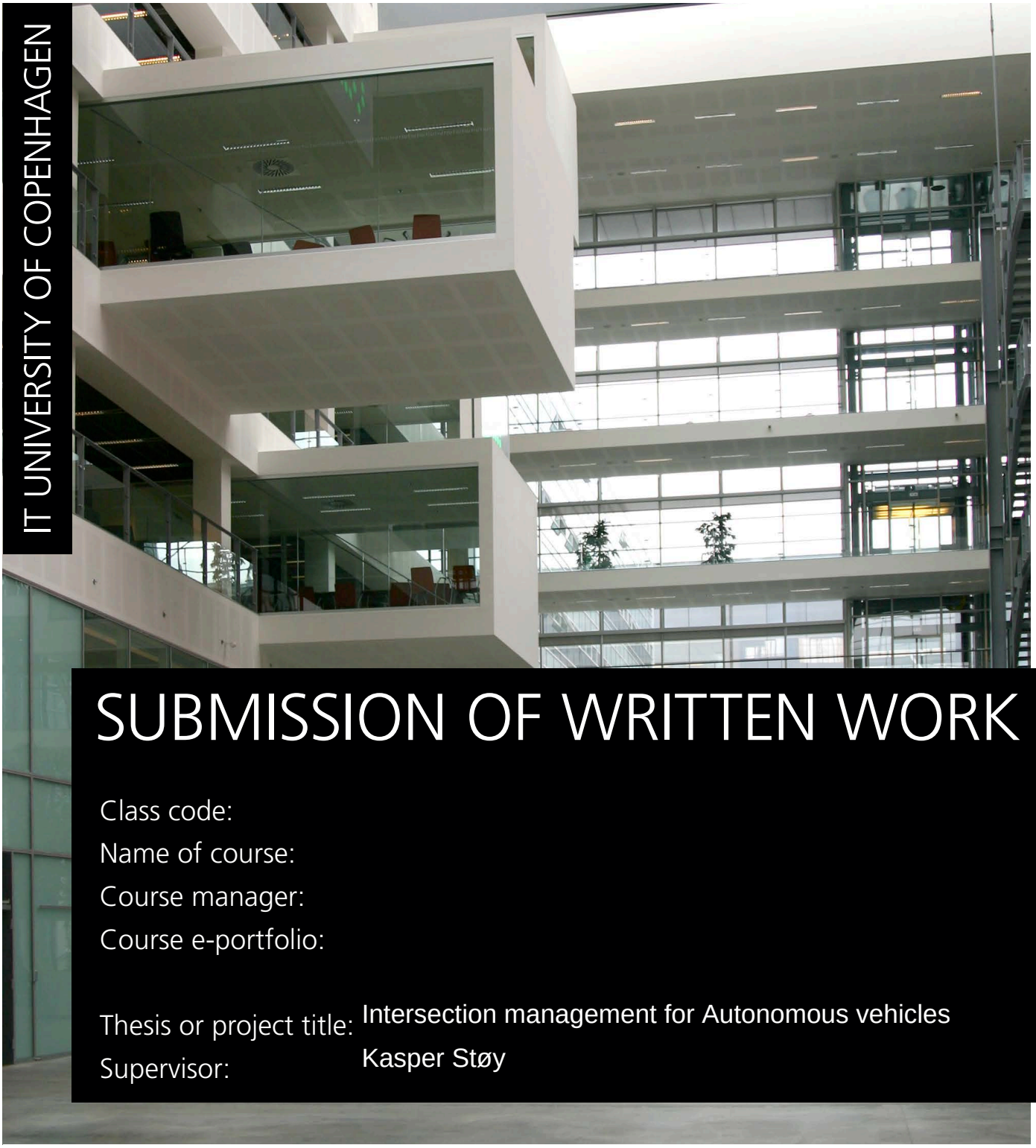
IT UNIVERSITY OF COPENHAGEN



# SUBMISSION OF WRITTEN WORK

Class code:

Name of course:

Course manager:

Course e-portfolio:

Thesis or project title: Intersection management for Autonomous vehicles

Supervisor: Kasper Støy

| | Full Name: | Birthdate (dd/mm-yyyy): | E-mail: |
|---|---|---|---|
| 1. | Ivan Naumovski | 17-07-1989 | inau @itu.dk |
| 2. | Martino Secchi | 04-08-1991 | msec @itu.dk |
| 3. | | | @itu.dk |
| 4. | | | @itu.dk |
| 5. | | | @itu.dk |
| 6. | | | @itu.dk |
| 7. | | | @itu.dk |

# Autonomous Car Intersection Simulation Project

Ivan Naumovski, inau@itu.dk
Martino Secchi, msec@itu.dk

supervised by
Prof. Kasper Stoy


December 12, 2016

# Contents

**Abstract**

This project aims to investigate some of the aspects involving traffic management with autonomous vehicles. In particular, we want to know how it can be possible to regulate self driving cars through an intersection, and if it is possible to achieve this without a centralised controller and the use of communication. We will run a simulation of an intersection with self driving cars and discuss our findings, comparing different methodologies and related work.

# 1 Introduction

The idea of having autonomous agents has existed in science fiction literature for hundreds of years and can also be traced back to ancient mythology. The ideas and concepts of autonomous agents are hence not something entirely new, but rather something that has been refined over the course of a lot of years. However the technology has just recently reached a state where a lot of the underlying challenges to autonomous agents can be solved somewhat efficiently. Self driving cars are becoming more and more a reality, and they will be an important presence in the near future. By automising vehicle control, traffic management can arguably be made more efficient in many ways, starting with throughput but also in other aspects, like fuel management and emissions control.

In this paper we will focus on what it takes to make autonomous automobiles efficient in passing intersections.

Initially we had a look at what other researchers had been doing the recent decade and quickly realised that there was a pattern. The same groups keep appearing in the literature when having the focus on autonomous vehicles and intersections.

To put everything in context we will start by presenting some of the challenges that are present when trying to efficiently solve the task of managing vehicles in an intersection. Followed by different control paradigms. These paradigms will be presented in relation to existing solutions to match the theory to their practical implementation.

## 1.1 Motivation

The theory we have touched upon is from multiple fields of engineering. One field is within traffic engineering, namely intersection management. The other is within the field of robotics, namely robot control. First we will briefly explain why intersection management is deemed relevant followed by a similar section for why robot control paradigms.

### 1.1.1 Intersection Management

The field of intersection management tries to deal with the challenges arising when trying to organize multiple entities to efficiently pass through a critical zone.
The challenge to solve is to go from origin to destination while not colliding with other entities.

In the real world these challenges are solved by defining a set of rules and designing the critical zones such that it is trivial to determine from where entities might enter and leave the zone and what entity has priority over the others.

Different templates exist for solving these tasks efficiently and these templates can differ from nation to nation, however our starting point is rooted in the way that traffic is handled in Europe which is kind of homogenous.

The commonalities used are traffic-legislation and intersections. Traffic-legislation contains a set of rules that makes it easy to determine what entity has a higher priority within critical sections (intersections).
Intersections are also designed in such a way that agents have an easier task at deciding priorities. This includes tools such as traffic lights, road markings and/or signs.

The intersection models that we are interested in are 4 way cross-intersections both with and without the use of traffic lights. This is due to the fact that these types of intersections are the most common ones in our local area. Additionally a smaller intersection still follows the same rules and can be seen as a 4 way intersection where traffic never arrives from or travels to one of the ways.

### 1.1.2 Control Paradigms

We use the term agents for robots or other actors moving within an environment.
We use the term environment as being synonymous to the world in which the agent is confined. The following will contain short presentations of the three canonical ways of doing robot control, based on the theory presented in the textbook The Robotics Primer[1], and chapters 12 to 15 on the subject.

*Reactive Control* which is also coined Sense-Act is a paradigm that acts directly on sensory input.
In our research we did not stumble upon any systems using purely reactive controllers, which we found interesting. This is due to the fact that the simplicity of the interaction with the environment should make pure reactive controllers quite safe. Simple schemes for collision avoidance are trivial to implement using pure reactive controllers.
An example of such a controller can be expressed quite simple using only a frontal sensor. The rule could be phrased as while this sensor does not register

anything in front of the vehicle for a certain distance accelerate else brake. The strength of this paradigm is that it is highly reactive to the environment. There are no complex layers inbetween.

In contrast to the above paradigm there exits a paradigm named *deliberate control*, or alternatively Sense-Plan-Act. Compared to the reactive control it introduces a layer of planning. The additional step however requires more information about the environment.

Any change to the environment would also invalidate the current plan as this is not factored in. This is computationally heavy since it senses, plans a step, executes the step and starts over. A plan can consist of multiple steps which are required to reach a goal.

The third canonical way of doing robot control is a hybrid of the two above, and hence it is also referred to as *hybrid control*. It resides somewhere in the middle where it has abstractions of routines using reactive controllers - but uses a higher level mechanism to combine these for advanced control schemes.

Most of the research we discovered on intersection management systems for autonomous vehicles tends to prefer centralised systems that can handle traffic requests. Particularly relevant is the research of two major groups in this area: the researchers at the University of Texas at Austin, and an international group of researchers composed by members of the Massachusetts Institute of Technology (MIT), the Swiss Institute of Technology (ETHZ), and the Italian National Research Council (CNR).

At the University of Texas, Kurt Dresner and Peter Stone developed AIM, Autonomous Intersection Management, a reservation-based system built around a detailed communication protocol able to coordinate movement of self driving cars through intersections [2]. Through a simulation they are able to demonstrate the potential of this system to outperform current intersection control mechanism: traffic lights and stop signs. In the simulation, the intersection center is divided into a n x n grid of reservation tiles. Through a "first come, first served" policy, approaching vehicles make a request to the system to reserve the space-time they need to cross the intersection. Their trajectory is then computed by the system and if the requesting vehicle at any time occupies a reservation tile that is already in use, the request is rejected. The vehicle will then continue requesting until it can pass.

A similar approach is also followed by the international group of MIT, ETHZ and CNR researchers. They developed a centralised slot-based intersection [3]. In their simulations, cars adjust their velocities in approaching the intersection in order to arrive and cross at a given slot of time that is made available for them. This system also involves communication between vehicles and a centralised controller.

Additionally, Dresner and Stone, define certain criteria that need to be met

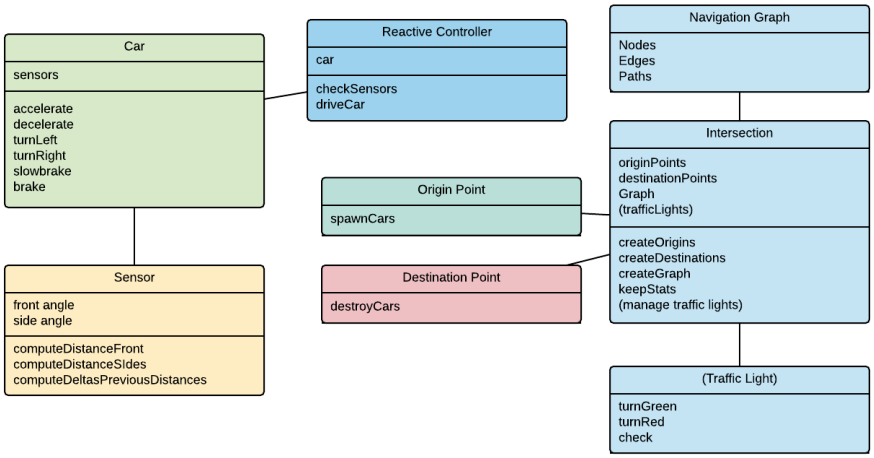



Figure 1: Overview of Unity3D software package

for the interactions between intersection and multiple agents to be deemed succesfull. They name properties such as *autonomy for every agent*,*realistic sensor models*, *deadlock avoidance*,*safety* and *efficiency*, which we adopt into our model.

They mention more properties - such as protocol standardization, low communication complexity and Incremental Deployability, which we deem less important due to our approach.

## 1.2 Problem Statement

The previously obtained insights about robot control, intersection management and previous research led us to the following plan of attack:

> The goal is to explore the possibility of building a decentralized system which can guide autonomous cars through intersections. The current research and solutions we have found are focused around centralized systems.
>
> Centralized systems rely alot on scheduling and coordination in a 'global' scale, while the decentralized system we envision would be built on a reactive approach which is centered around local state.
>
> We would like to build a prototype using the decentralized approach and compare results to an existing centralized solution.

# 2 Implementation

Post-research we were in a limbo, since the landscape of tooling that can be used for setting up experiments in this field of studies is quite vast. However it was decided early in the proces that it would be mainly software simuation, due to the fact that robots have a higher cost and getting the amount of agents that makes these types of studies interesting is somewhat higher than a dozen of robots.

## 2.1 Simulation Tools

Performing simulations is not trivial. Different criteria should be met depending on the accuracy of the simulation that is to be performed. In particular one should decide what level of abstraction fits the scenario, as 1:1 simulations are significantly harder to model, but might not necessarilly add value matching the increased complexity.

This introduces the question, what modelling complexity is sufficient for modelling our problem?

While the real world has three dimensions, we only need a two dimensional model. The height dimension is not required since we can consider every agent to occupy a space in two dimensional space. Restricting the simulated world to only two axis makes the model alot simpler to model. If this was a simulation for aircrafts the height dimension would make a lot more sense to include.

Two tools have been considered for doing the simulated world. One was the GazeboSimulator, which is a widely adopted tool in the robotics community. Another was doing a simulation using game development tools, in particular Unity3D.

While both tools provide the functionality we need such as physics simulation in 2D/3D environments, the degree of tools supported out-of-the-box varies. Gazebo has robotic simulations as being its main function, and hence has alot of useful tools for robotics integrated into it. Unity3D on the other hand is a general-purpose tool so it requires some tailoring to get the same functionality as Gazebo.

Unity3D however is a strong competitor due to the fact that we have prior experience using this tool. It is easy to go from concept to an actual prototype rapidly. And last but not least has a thriving community of professionals, academics and hobbyists aspiring to make the software great.
Our solution is built using Unity3D.

The choice, albeit it might seem arbitrary, is based on the fact that *Gazebo* has a very steep learning curve.

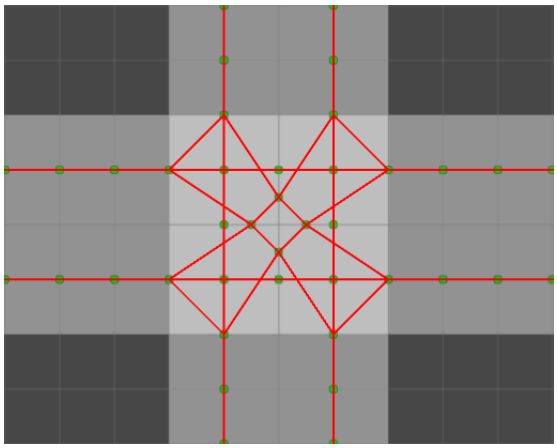


Figure 2: Image of the underlying graph representation for a four way inersection.

## 2.2 Software components

As we have settled on using Unity3D we work within its boundaries. It is component driven, meaning every object in the world consists of a collection of components and a transform - the transform is the objects spatial information. These components can be either colliders, renderers, or custom scripts, and objects can contain other objects - they can be composite.

The implementation was done by splitting the work in three major blocks. One was *the vehicle model*, another was *the intersection model* and the last *the reactive controller* for the car. An overview of our software package is depicted on figure 1.

### 2.2.1 The World Model

As observed in the real world, multiple intersection models exist, three lane and 4 lane intersections are the most common ones. The style can be either with the roundabout approach, which essentially rules out the left turn complication or a traditional 4 lane intersection where going left crosses the opposing traffics lane.

In our model we mainly focus on a single lane 4 way intersection.
We tried different intersection representations as can be seen on figures 2 and 3.

We represent these intersection models using an underlying directed graph. A car agent in our environment is seeded with information about its origin and its destination. These points are vertices in our graph. Every combination of origin and destination pair has one path.

While this does not provide alot of flexibility in straying from the path, the simplicity makes the model really easy to implement and adjust.

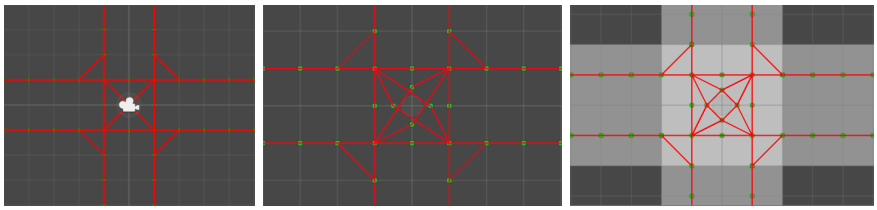


Figure 3: Image of the previous versions of the underlying graph representation.

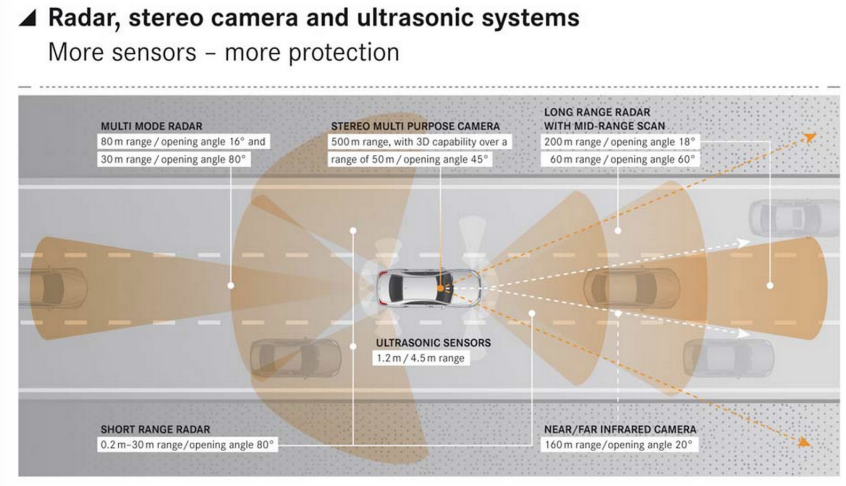



Figure 4: Image of the sensors integrated into a car.

Once we reached the test phase we also realised that we needed to change the values to real world values.

We translated our cars to being a fixed meter value. This value was not arbitrarly chosen, it was based of values from the range in which cars of that class belong to, namely compact car. Compact category covers cars such as Ford Focus, lengths are between 4.1 and 4.45 meters for hatchbacks and a bit more for sedans, saloons and station wagons. The simulation car size was 0.8 units. We went with 4.4 meters as being the real world value, and hence got a scaling factor of 5.5. Using the scaling factor we also translated the simulated vehicles speeds to km/h, which is quite helpful for the comparisons.

**The traffic lights**

This is an attempt to mimic traffic light regulated intersections. The lights change at intervals chosen in regards to our experiences with intersections. These values can vary alot from intersection to intersection - and are highly influenced by traffic flow during the day. We did experiments with different interval windows.

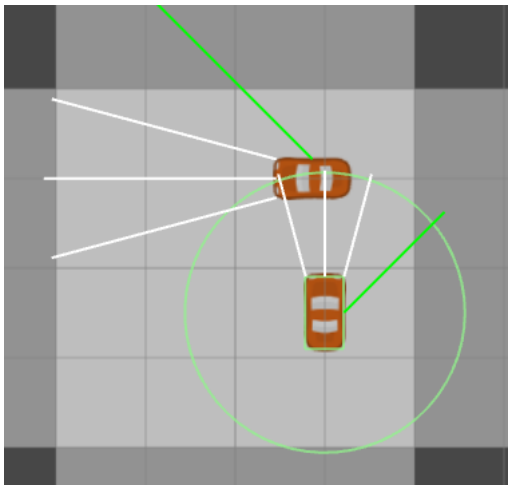


Figure 5: Image of Unity sensor model.

### 2.2.2 The Car Model

The way the car has been modelled is using a interface which essentially contains actions such as accellerating or decellerating, braking, and turning left or right. This abstraction simplifies the step of going from a virtual model to a physical one. The interface provides loose coupling between the controller and the underlying vehicle implementation. This seperation makes it simple to develop multiple types of controllers or vehicles.

**The sensors**
Studying autonomous vehicles, such as the Tesla's we realised how much sensing is actually integrated in these marvels. As seen on figure 4, a lot of sensors exist for achieving autonomous control in vehicles. The figure is not representative for all autonomous vehicles - but can be seen as a good starting point for understanding the complexity of the vehicles.

We focused on mimicking the frontal sensors and the cross-traffic sensors. We investigated the products of a well-known vendor, SICK, which is used by many different companies within the field. The sensing is done similar to the current generation of autonomous vehicles. We mimick 190 degree LMS5xx series for the frontal sensor. This is a sensor which uses lasers. It provides a high degree of precision. It functions in a lot of different conditions and works in the range of 0 to 80 meters.

### 2.2.3 The Reactive Controller

This was developed incrementally. Initial versions were done using only collision avoidance. The vision was to have multiple different controllers and be

able to hotswap these for the simulation. Later versions also included constructs for priority such as the right hand rule and a finer grained distinction between moving objects and their movement in relation to the agent.

Due to the simplification of only using one frontal sensor compared to the amount of sensors others might use, we have a finer grained way to interpret the input from this abstraction. The 180 degreee cone is split into zones, which essentially mimics the same division of the frontal senses as seen on figures 4 and 5. We took the abstraction further and limited the range of detection based on velocity of the vehicle. The faster it goes the farther it looks ahead. This reduces the risk of vehicles being overly cautious and stopping in intersections with lots of slow moving vehicle, especially if it is still possible to pass through. Additionally the braking distance is not as big at lower velocities - which makes this approach viable.

The controller has been reiterated multiple times: initially it was only doing collision avoidance. This was achieved by acting directly on the frontal sensor values. The earliest versions only had one threshold. If anything was closer than the given threshold the car would not move. While this avoids colliding with anything infront, it does not handle following a car infront very well. We did another version which trails behind the car in front. This is done using the delta distance rather than raw distance - if both vehicles move at the same speed the delta stays zero. If the gap closes it gains a negative value and if the gap increases it gains a positive value.

# 3 Experiments

To compare performances of the system over different paradigms and with different parameters, we ran multiple experimental simulations. The parameters that we changed over the different tests are from three classes: the paradigm, the speed of the cars, and the complexity.

For paradigm it is meant the use of a traffic light regulated intersection versus a pure reactive approach. As for complexity, we can increase it by allowing the cars to turn in order to reach any destination, or we can use a simpler system where cars can only go straight.

The car speed is changed over three base cases: slow ( 30 km/h), medium (40 km/h) and fast (60 km/h). It is worth mentioning that the advised speed when turning the car in 90 degrees turns inside urban areas is around or below 30 km/h.

When simulating with traffic lights enabled, we test two intervals of time that regulate the duration of the green light. We have an interval window of approximately 1 minute, that reflects a realistic scenario according to our observation of real life intersections. We also test a much shorter window of 20 seconds, that should improve performance under the assumptions that:

1. autonomous vehicles can react faster than human drivers, so they can start

moving immediately after the traffic light turns green compared to human drivers that could be slow or distracted;

2. shorter windows of time allow the traffic to be managed evenly, since traffic is evenly distributed between the different directions.

### 3.1 Experimental results

We measure performance of the systems mainly in terms of throughput, namely the amount of cars that get through the intersection per minute. However, we also account for other factors, like the amount of collisions, the amount of deadlock situations, the time at which these occur and the frequency of these events.

A deadlock is the condition in which cars have been stuck in the intersection zone with no one entering or leaving for a period of time (about 10 seconds).

To test the implementation of our system we also account for information such as the number of cars spawned, number of cars "lost" (not reaching the correct destination) and the number of cars present in the intersection zone at the occurrence of a deadlock or collision.

#### 3.1.1 Throughput

The graphs in figure 6 show the different throughputs of the system running with different factors.

## 4 Discussion

Discuss experimental results. Shortcomings of model. Improvements for further tests.

## 5 Conclusion

## References

[1] Maja J. Mataric. *The Robotics Primer*. MIT Press, 2007.

[2] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. 1905.

[3] Remi Tachet, Paolo Santi, Stanislav Sobolevsky, Luis Ignacio Reyes-Castro, Emilio Frazzoli, Dirk Helbing, and Carlo Ratti. Revisiting street intersections using slot-based systems. 1905.

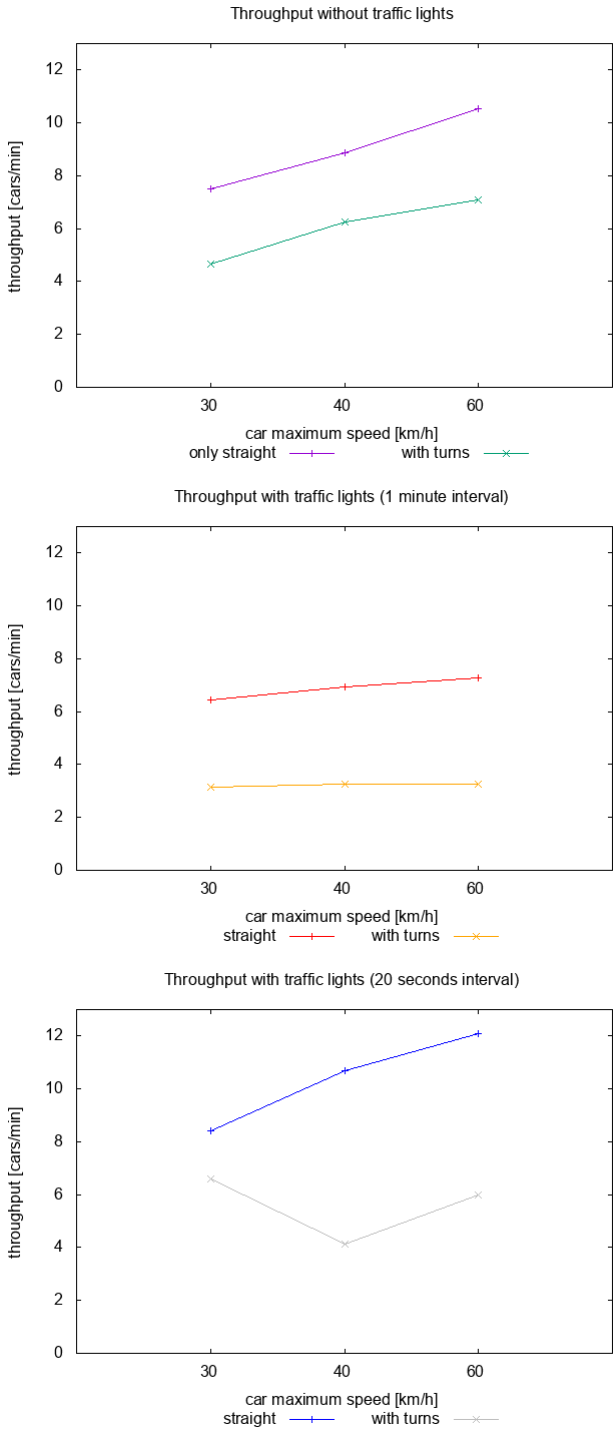



Figure 6: Throughputs in comparison