Path planning for cyclists

Simulating bicycles in urban environments

Simon Rosman

Master thesis

ICA-3754014

simon.rosman@gmail.com

Utrecht University

Supervisor I

Dr. Roland J. Geraerts

Utrecht University

Supervisor II

Dr. Ir. A. Frank van der Stappen

Utrecht University

August 26, 2015

Abstract

In the field of path planning and crowd simulation, many models have been developed for simulating pedestrians, cars, and autonomous vehicles. However, no such models exist for bicycles. We present an analysis of the characteristics of bicycle riding and provide a model for simulating people riding bicycles. The method performs in real time, supports collision avoidance between cyclists and pedestrians, and handles dynamic obstacles such as parked cars.

Contents

1 Introduction

Bicycles are a commonly used mode of transportation in both urban and rural environments. They enable a person to move around quickly and at low energy cost, in particular when compared to walking. The motion of bicycles, i.e. its stabilizing- and steering-characteristics, is complicated. This is due to a few factors: firstly the mass of a person riding a bicycle exceeds that of the bicycle itself by far. Secondly, sudden changes in speed, e.g. when accelerating quickly, apply large forces onto the steering mechanism. These characteristics make bicycles an excellent subject for research. A model for simulating bicycles has many applications, such as games, driving simulators, traffic management, and road-safety design. There are virtually no models for simulating cyclists. We present a model that simulates cyclists by following a given global path. The model supports collision avoidance with cyclists and other road users such as pedestrians.

The dynamics of riderless bicycles has been a subject of research for quite some time. In the late 19^{th} century, the first works were published. In 1899, Whipple [Whi99] presented one of the first models of the self-stability of a bicycle in the form of linearized equations, which proved to be almost accurate. Meijaard et al. [Mei+07] have created an improved stability model which is now commonly accepted as the standard benchmarking model for bicycle dynamics. However, we are interested in much more than just the mechanical properties of a bicycle, and its handling dynamics. We are approaching this subject from an angle of path planning and crowd simulation. This means that we also want to investigate how a cyclist behaves in its environment. The (relatively) low speeds and the responsiveness of a bicycle enable cyclists to perform complex maneuvers quickly and allow them to ride in close proximity of other cyclists that would not be within the safety margins for other kinds of vehicles. We therefore cannot use methods that are tailored to other vehicles, such as cars. On the other hand we also cannot use the methods that are used for simulating crowds of pedestrians since there are non-holonomic constraints that need to be satisfied [AKL05], i.e. that a bicycle cannot move sideways, which we will elaborate later on. However, cyclists and pedestrians also share some characteristics. First, both cyclists and pedestrians do not move in exact straight lines, but present oscillating behavior. Moreover, they display behavior of group compression when an environment is densely crowded or when jams occur [Nar+09]. Finally, they both have many (shared) degrees of freedom and can perform numerous interactions, e.g. stepping on/off a bicycle, allowing for an infinite action-space. Therefore we might be able to base our method on existing planning methods for crowds while incorporating the non-holonomic characteristics of car-like systems.

1.1 Research questions

A brief look into the literature shows us that it might be possible to build our research upon existing methods, which we will further investigate in Section 2. While most of the literature focuses on the planning problem for single moving vehicles, we want to support interactivity between cyclists and other road users. Therefore, we present the following research questions.

1 *What model is best suited for the path following problem of a bicycle with an active rider?*

2 *How can this model be modified to support collision avoidance*

   (a) *for static obstacles (e.g. a street layout);*

   (b) *for dynamic obstacles such as other bicyclists, pedestrians, and cars,*

   such that realistic everyday traffic scenarios can be simulated?

3 *How can we validate the results of such a model such that the similarity between real cyclists and the model's results can be determined?*

If we can provide adequate answers to these questions, we should be able to create an appropriate solution for simulating heterogeneous traffic situations involving cyclists. In the upcoming sections we will look into the existing methods in more detail. We will first provide a brief introduction into the kinds of bicycles that we are interested in.

1.2 Common types of bicycles

There are many types of bicycles in existence, some of which are known by different names per country. We will provide a brief description of the bicycle types that are the most rudimentary, and are in our opinion the most commonly used. Additionally, we will present some special types of bicycles that are used less frequently. The first two types, the *European city bike* [Bal+00] (also known as *"Stadsfiets"* in Dutch), and the *Racing bike* have a shape and geometry that is quite alike. The former can be driven while being seated upright whereas the latter is being driven in a more huddled position, where the upper body is bent towards the ground for more aerodynamic efficiency. The *mountain bike* has a sturdier frame and is equipped with coarse-treaded tires, suitable for rough terrains. *Mountain bikes* are usually also equipped with spring-loaded wheel suspension to ease in taking jumps. The *recumbent bike* [FW81] is a slightly divergent type of bicycle that is also used in some countries, such as the Netherlands. The rider is positioned close to the ground and laying down facing slightly upwards, which allows for more comfort during long trips while reducing air resistance. The aforementioned bicycles also exist in dimensions suitable for children, but we will not consider this difference in our research and assume a bicycle model for adults.

Researchers have also created experimental bikes for the purpose of analyzing their riding characteristics. Some have been proven to be next to unridable, e.g. the *rear-steered bike* [AKL05], while others improve the maneuverability by having two steers, such

as the *sideways bike*. A final example is the *center-steering bike* for it has no steerable wheels at all but articulated steering, i.e. the bike has no steerable axes, but has its frame pivot in the center. While this improves the turning ratio, it causes problems in balancing. We will not consider these experimental types in our research.

1.3 Generic bicycle model

We will assume a generic bicycle model with a set of commonly accepted parameters, as depicted in Figure 1.

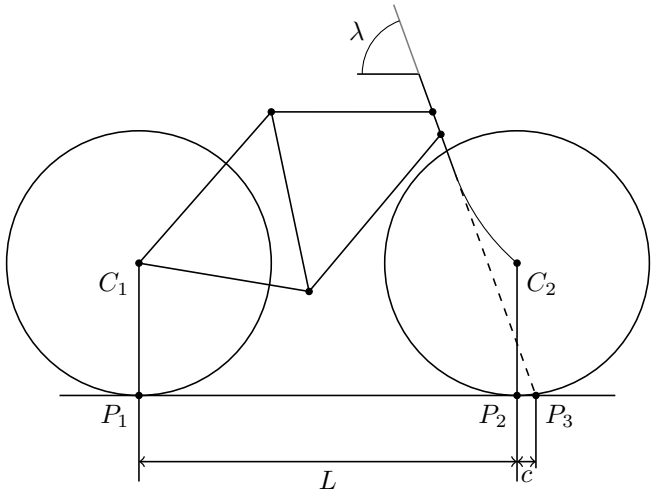


Figure 1: *An overview of the geometry of the generic bicycle model [AKL05]. Points 'P1' and 'P2' denote the contact points of the wheels, whereas point 'P3' denotes the intersection of the frontal axis and the ground plane. The trail 'c' is also affected by the angle of tilt of the steer axis 'λ'. 'L' is the wheel base.*

The generic bicycle model represents a two-wheeled, frontwheel-steered bicycle. The bicycle has a tilted front axis, and a bent frontal fork providing the offset controlling the trail length. Common values for trail c range from $3 - 10cm$, wheel radius $r_i = P_i - C_i = 35cm$, wheel base $L = 1$ m, and steer angle $\lambda = 70°$ [AKL05] where $\lambda = 90°$ would represent a vertical steering wheel without tilt.

1.4 Structure

The following sections are organized as follows. First we will provide an overview of existing related works in Section 2. Next, in Section 3, we will elaborate on the methods and techniques that are best suited to our research questions. Section 4 presents the implementation of our method, followed by an analysis of its validity in Section 5. Finally, we conclude on the presented work and the results, describe the limitations of our research, and provide suggestions for future works in Section 6.

2 Related works

In this section related works in the fields of bicycle dynamics, curved path finding, and collision avoidance are presented. We want to determine how realistic everyday traffic scenarios involving cyclists can be simulated. A first question considering this problem that arises is how cyclists plan their route? Using common sense we can establish that, in general, a cyclist plans the shortest route on a city level that avoids collisions. Since the topic of finding global paths on a city level, i.e. find the shortest set of roads connecting two points is mostly solved, we focus on the issue of local path finding. When we dig further into local path finding and collision avoidance, the next question that arises is how bicycles steer? The final question that we encounter is how bicycles work to begin with? The following sections will describe the research on these topics, starting with the dynamics of a bicycle, and advance up to interaction between bicycles.

2.1 Bicycle dynamics

The first work we mention is Whipple's model [Whi99]. In 1899, Whipple presented the first generic model describing the stability of a bicycle. Although the work contained some errors, we include this work for historic purposes as it conducts the start of bicycle research, and most later works used its equations as a basis. Much more recently, Meijaard et al. have presented a series of equations for the dynamics of bicycles [Mei+07]. Their work is now considered as the accepted standard [Moo12]. Their work describes the motion of a riderless bicycle whilst our work considers bicycles with an active rider. However the main issue is, since their method computes exact solutions for self-stability, that it is much too low-level and meticulous for simulating cyclists, and cannot be used to plan motion between two positions. The authors already acknowledge this mentioning that their model is *unnecessarily and inappropriately complex for understanding basic features of active rider control* [Mei+07, p. 1958], for understanding how cyclists control their bicycles. Nonetheless, we can use their work for determining and verifying the constants of our own model. Astrom, Klein, and Lennartsson present various models for bicycle steering and stability. Their simplest model assumes an untilted steering axis, a constant speed, and a rider rigidly attached to the bicycle frame [AKL05]. They provide more models with increasing complexity, adding the dynamics of the front fork, self-stabilization, manual control, and gyroscopic effects. The focus of the paper, however is not on bicycle steering. They do look at bicycle steering, but use a dynamics model with complexity similar to Meijaard et al. [Mei+07], and only look at the effect of steering on the resulting path and bicycle tilt. They define a transfer function from steering angle ϕ to tilt angle φ. Astolfi present a generic method for stabilizing the controls for a non-holonomic system [Ast96]. Yavin uses a simpler dynamics model for controlling the bicycle, and focuses on steering the bike towards a given point [Yav98]. However, he assumes a riderless bicycle, and does not consider planning a path towards a goal.

Moore investigates humans controlling bicycles [Moo12], but focuses on keeping the bicycle stable. He uses a crossover model based on human pilots [MK74] to control

the steering torque and body leaning angle applied by the rider based on sensed tilt angle. His model [Moo12] supports path following, given an existing path, but only as a byproduct of the stabilization. Kooijman, Schwab, and Moore investigate human aspects of bicycle control. They establish that the upper part of the body is not used to keep the bicycle stable during motion but is only used in sharp corners to account for the sudden change of direction [KSM09]. Additionally, they discovered that the steer is only turned in small angles ($\phi \approx 3°$) at most speeds. When sharp corners are to be taken the steer is turned to a maximum of $\phi \approx 15°$ [KSM09] [Moo12]. These observations will make it easier for virtual characters to be correctly animated when riding a bicycle as only their arms and legs will have to be moved, which can be achieved through inverse kinematics. The range of steering angles provided can be used as constraints for our path-following method which tries to find a local path that satisfies non-holonomic constraints and follows a given global path. [Sha08] investigates Rider control theory for the bicycle, and shows that the bicycle's dynamics have a much smaller effect on bicycle steering than the control empowered by its rider. This argues that we do not need a complex dynamics model to simulate bicycles with active riders.

Numerous other works provide more details on the bicycle's dynamics, as well as slightly different bicycle models, or special cases, including locked steering [RM71] [FSR90] [Faj00] [Moo12]. Since Kooijman, Schwab, and Moore and others show that we do not need a complex bicycle model for simulating cyclists, the next problem we should solve is: *How does a bicycle steer?* As this problem is closely related to the problem of finding a path that satisfies the non-holonomic constraints, the next subsection elaborates on research on existing path finding and following for motion planning with non-holonomic constraints.

2.2 Path finding and following

Several approaches for finding constrained paths exist. Dubins presents a method for finding a path with constrained curvature [Dub57]. However, the model produces open-loop trajectories that assume constant speed and ignore kinodynamic constraints. Furthermore, the model assumes an obstacle-free environment. Getz and Marsden present a dynamics model for stabilizing a bike that incorporates path following [GM95]. The method determines yaw and roll angles given a path and computes the lean and speed settings for the bicycle. However, they do not consider how to find such a path that avoids obstacles. Bickford presents a method for path following which uses a *linear quadratic regulator* (LQR) to stabilize the bicycle [Bic13]. The method is able to track constant curvature paths, but does not focus on finding a valid path and assumes a collision-free path is given.

Other works do not focus on bicycle planning but focus on general curves and trajectories. Nienhuys and van der Stappen [NS04] present a method for checking the intersection of a needle curve with a polygonal mesh. This method could potentially be used to compute a discrete approximation of a curve in free space; However, free space is usually not explicitly represented as a mesh. Additionally, the method assumes a fixed nee-

dle angle while the steering angle of a bicycle is not fixed. Alterovitz et al. have developed a method to compute the path of a steerable needle [AGO05]. Given input configuration parameters orientation *(i.e. yaw)*, rotation *(i.e. roll)*, and location, the method computes the path the needle will follow through soft tissue. Their method supports deviations from the optimal path caused by tissue deformations. Bicycles, however, generally move on a solid surface. Webster et al. present another method for steering needles [Web+06]. Their method supports a 3D coordinate system, and thus it has degrees of freedom, which are not supported by a 2D method, such as ours. The method presented by Agarwal et al. [Aga+02] is able to find a curved path between two configurations q_1 and q_2 and given curvature limits, if one exists. However, it is only able to operate in a convex polygon that is free of obstacles. As a result, using solely this method is not sufficient if q_1 and q_2 do not reside in the same convex part of the navigable space, in which case a global path through these convex polygons has to be planned using other methods. Issues may rise when transitioning from one to another convex part of the global path. Additionally the path's curves only have a unit radius, whereas bicycles have a variable turn radius.

The methods presented thus far are not suitable for planning a path through a large-scale environment containing obstacles. Therefore, we will now elaborate on methods that can find global paths that satisfy non-holonomic constraints. Lazard, Reif, and Wang have shown that the 2D case of motion planning with curvature constraints is NP-hard for an environment containing polygonal obstacles [LRW98]. Since finding valid paths using continuous methods is complex, sampling-based methods, such as Kuffner and LaValle's RRT-connect [KL00], are popular [LaV06]. This sampling-based planning method is suitable as a global path planner, i.e. to determine whether a path between two points exists. It can cope with non-holonomic constraints. However, the method produces paths that are non-optimal and jagged. Therefore, smoothing should be applied before using the resulting paths. LaValle and Kuffner use an RRT-based approach to plan with kinodynamic constraints [LK01]. Several attempts have been made to produce smooth paths, such as RG-RRT [SWT09], LQR-RRT [Per+12], and POSQ-RRT [PA14]. However, many RRT-based methods produce open-loop paths that cannot deal with dynamic obstacles without replanning. Bereg and Kirkpatrick [BK05] present a method to find corridors with limited curvature that satisfy non-holonomic constraints. Their method is able to operate in a polygonal environment. However they assume unit curvatures. Other methods used for crowd simulation, such as created by Jaklin, Cook, and Geraerts, can find a path in a polygonal environment [JCG13]. Their method can find paths with a guaranteed clearance to obstacles and supports terrain preferences, but it is not designed to cope with non-holonomic constraints.

None of the above works is completely suitable for solving the problem of path finding and following for bicycles. However, apart for solving this problem for a single cyclists, we also have to consider the problem of simulating multiple cyclists, as mentioned in Subsection 1.1, which we will discuss next.

2.3 Interaction between bicycles

Collision avoidance for moving obstacles in general is a topic that has been widely discussed and written about. Since the dynamics between cyclists are closely related to this topic, we will first discuss whether some of these works can be directly applied. For example, Helbing and Molnar's vector-based force model may be taken into consideration [HM95] for implementing the dynamics between bicycles. It is easy to understand and implement as it computes a weighted sum of attracting and repulsing forces, which is then used to steer a character. However, this method is extremely prone to deadlocks and oscillating behavior. Additionally, it cannot handle non-holonomic constraints. Tanner et al. [TLK01] and Abdessemed et al. [ABM04] present methods for navigating non-holonomic vehicles that support collision avoidance for static and dynamic obstacles. These methods plan a path around local obstacles in a reactive manner. However, since they do not focus on avoiding other vehicles, their methods are prone to deadlocks in narrow spaces. Lindemann et al. use a similar approach for collision avoidance [LHL06]. Their method ensures a smooth path, but does not consider moving obstacles.

Other methods are better suited towards avoiding collisions between multiple bicycles. Frazzoli, Dahleh, and Feron follow an RRT-based approach [FDF02]. Their method plans a global path using an incremental Probabilistic Roadmap, and uses RRT to plan local paths that avoid moving obstacles. The incremental roadmap contains milestones, which are used to ensure that a local path does not lead into a dead end. Tanner, Loizou, and Kyriakopoulos present a method for planning motion for multiple non-holonomic vehicles [TLK03]. Their method avoids collisions for multiple vehicles simultaneously using potential fields, but is tailored to multiple robots that work in cooperation. Additionally, the method approximates all robots and obstacles as points in order to use a potential field-based algorithm.

Kuwata et al. use a slightly different RRT-based approach [Kuw+08]. Their method steers a car towards a goal position using an RRT to plan around local obstacles. They use a sampling technique biased to sample points in front of the car, and filter the resulting trajectories based on their feasibility. The sampling strategy can be controlled such that it supports lane following, overtaking, taking U-turns, and parking. The method is however tailored to autonomous vehicles, and, therefore, adheres to safety characteristics that do not apply to cyclists. Therefore, this method, when used to simulate cyclists, will produce unrealistic results because the cyclists will keep too much safe distance, and will not be able to overtake within one lane. Xu et al. present a trajectory planning method for vehicles [Xu+12]. They discretize a road into segments and lanes, and plan a trajectory to change lanes when an obstacle blocks the current lane. Acceleration and steering constraints are taken into account when picking a trajectory to a different lane. The method is geared towards autonomous vehicles.

Other recent methods, such as RVO [Ber+11], may also prove useful. The RVO method [Ber+11] projects a cone in front of each character, which other characters try to avoid

[Ber+11]. Although this method is tailored to characters without non-holonomic constraints, its principles may be useful for collision avoidance for non-holonomic vehicles. This is attempted by Alonso-Mora et al. who present NH-ORCA [AM+13], a method for collision avoidance for non-holonomic vehicles. Their method uses the concepts of RVO [Ber+11], but is only successfully applied to differential-drive non-holonomic robots, i.e. robots that can rotate in-place.

2.4 Route planning

Route planning considers the task of navigating on a city level, i.e. which roads or bicycle paths are chosen to a destination. Although this is not a main topic in our research, some aspects of it may prove useful. Data from this field of research can also be used to validate our own method. Aultman-Hall et al. [AHHB97] investigate the routes taken daily by commuters through a large urban environment. Their work also reveals common preferences for cyclists, which may affect both global and local path planning. Hyodo et al. [HST00] investigate route choice of cyclists in an urban environment. They show that the path of a cyclist is not necessarily the absolute shortest path, and may be not even the cognitive shortest path, i.e. the shortest path that a person is aware of. They also present a model that can predict route selection using a genetic algorithm. Heinen, Wee, and Maat [HWM10] provide a thorough overview of existing literature on bicycle commuters and provide averages for trip lengths, trip distances, speeds, route choices, and present many more metrics. They also confirm Hyodo et al.'s [HST00] findings that there is no direct link between the shortest path and the selected path. CROW [CRO06], the Dutch authority for road design, has published guidelines for the design of bicycle roads. The work contains recommendations for safety measures, curbs, traffic-light setup, and road curvatures. While the work does not provide any models for planning bicycle routes, it does provide some established rules of thumb, such as side preferences employed by cyclists, which will explain in Subsection 3.1.

2.5 Traffic and cyclist behavior studies

A common method for validating road design is using Macroscopic Fundamental Diagrams (MFD's) in which the relationship between average speeds and average density is taken into account [DG08] [May90]. The traffic intensity q is computed using equation

$$q = k\,u, \tag{1}$$

where u is the speed, and k is the density of the traffic defined as the number of vehicles per kilometer of road. Geroliminis and Daganzo [GD08] show that such an MFD holds for cars in urban city traffic. They use data from a network of traffic sensors combined with GPS data from taxi rides. In another work [DG08] they verify that this relationship is not affected by traffic lights along a road section that are arbitrarily timed.

For cyclists, similar relationships also exist. This is shown by Papendrecht and Botma [PB89], who measured the lateral position of cyclists on various road segments along

common bicycle routes. They show that when traffic flow increases, the bicycle lane becomes more occupied laterally. They do not represent this relationship in an equation adhering to the form of eq. (1). Blankers [Bla12] argues that there is no correlation between traffic intensity and speed on small bicycle lanes when the majority of cyclists have the same common goal and share a high incentive for swiftness. Vansteenkiste et al. [Van+13] investigate the relationship between path width and cycling speed from a visual aspect. They show that smaller lanes cause cyclists to focus on the nearby path while wide lanes cause the cyclists to frequently gaze towards a goal position.

2.6 Data of bicycle motion

Authors such as Dozza and Werneke [DW14] use data of bicycle movements gathered using GPS trackers among other sensors to observe driving behavior. They have collected large amounts of data corresponding to some 1400km of bicycle routes. These data might seem useful for validating bicycle models, however the authors do not focus on the traversed trajectories, but on bicycle safety. As a result the precision of the trajectories is 2.5 m, which is quite coarse considering our purposes. Van den Ouden [Oud11] has also collected some trajectory data of a few short routes using GPS. However, the precision of his data is not given. In general, GPS data have not enough precision for comparison against simulated traversed trajectories. Kiewiet et al. [Kie+14] have also collected data of bicycle motion using a different approach. They used inertial sensors mounted on the bicycle frame and had a path taped on the ground that was followed by a cyclist. Although their data have better precision than GPS, they did not capture data of reference points, such as the distance to the verge or curb, as well as street layout. The traversed trajectories can be determined by integrating the inertial sensor data, but since this is an open-loop system, it is susceptible to rapidly increasing deviations over time.

2.7 Scope limitations

Additional literature is available on detailed aspects of the bicycle dynamics, but these are too intricate for our research. Therefore we will ignore the dynamics of bicycle frame elasticity, wheel suspension, tire deformation, air friction, and rolling friction. We will also not consider different shapes of tires, such as differences between toroidal and round tires. Furthermore, we will assume that there are no differences in terrain types. Finally, we will ignore rider lean, since it does not majorly contribute to the traversed path, and can be derived from the traversed path [AKL05].

3 Method

So far we have seen many methods that focus on the various aspects of motion planning, bicycle dynamics, collision-avoidance, and several works on data collection and method validation. We have established that none of these works focus on trajectory planning for bicycles. We have shown that integrating bicycle dynamics into a planning method is too complex for the purpose of simulating bicycles on a non-microscopic level, which is

also stated by Meijaard et al. [Mei+07]. Therefore, we can ignore these dynamics for the most part. Several works do focus on generating paths with curvature constraints, some of which even focus on paths suitable for bicycles, such as by Dubins' [Dub57] and by Getz and Marsden [GM95]. However, these methods only focus on computing the curves and do not consider obstacle avoidance. Furthermore, all of these methods produce paths with constant radii whereas the curves in bicycle trajectories have a variable radius. As for collision avoidance, many methods already exist, of which ORCA [Ber+11] is one of the best performing ones. It has only briefly been applied to motion with non-holonomic constraints [AM+13]. For determining whether collision-free paths exist, many methods can be used, such as the *Indicative Route* method [KGO09], which also guarantees a minimum clearance. We will use an approach based on an intuitive analysis of cycling behavior. Cyclists generally have a global route in mind when cycling from one location to another. Upon traversing this global route, obstacles and oncoming traffic have to be avoided. Since the field of view is usually limited to a couple of tens of meters, a cyclist can only start planning a local curve around obstacles and oncoming traffic as soon as they appear. The dynamics of the bicycle are controlled subconsciously as they have been mastered during the process of learning how to ride a bicycle.

Our approach uses a two-stage process for solving the problems of finding global paths and planning local curves. In the first stage, a global path free of collisions with static obstacles is computed that has enough clearance for our bicycle to fit through passages and to make turns. In the next stage, local curves are planned to avoid dynamic obstacles and oncoming traffic as well as to follow the global path. In most works from the fields of robotics, path planning and crowd simulation, these tasks are commonly split up for maintaining a structured overview and ease of computing, and we will do the same. We plan a local path within a planning window (which we will explain in Subsection 3.3) that satisfies the non-holonomic constraints of the bicycle. The local planner tries to find a path that is optimized for speed and tries to minimize deviations from the global path. In the final step, the local path is traversed by the bicycle and collisions with moving obstacles are avoided. When the local horizon has moved, the replanning step is repeated. This multistage approach ensures that the traversed path satisfies the bicycle's non-holonomic rolling constraints but has smooth curves with non-constant curvature. Furthermore, the resulting path is collision-free with respect to both static and dynamic obstacles. Finally, if the global path is the shortest one, the resulting path still belongs to the same homotopic category, thus respecting the preferences of the global planner.

We will now explain the approach and details of our method, starting with a brief explanation in Section 3.1 on the method's first stage, creating an *Indicative Route*. Next we will explain the differential model of the bicycle that is used for computing the non-holonomic motion in Section 3.2. This model describes some key variables, such as turn-radius ρ, which are used in the local planning steps and therefore we will elaborate on this before our actual method is explained. In Section 3.3, we focus on the main parts of our method, i.e. the local planning algorithm. In Section 3.5 we describe how

the ORCA method [Ber+11] can be used for avoiding collisions with other cyclists. We discuss some alternative approaches for path following in Section 4.2. In Section 4 we discuss the experiments that have been conducted to test the method, and discuss its validity in Section 5. Finally, we show and discuss some conclusions in Section 6.

3.1 Global planning

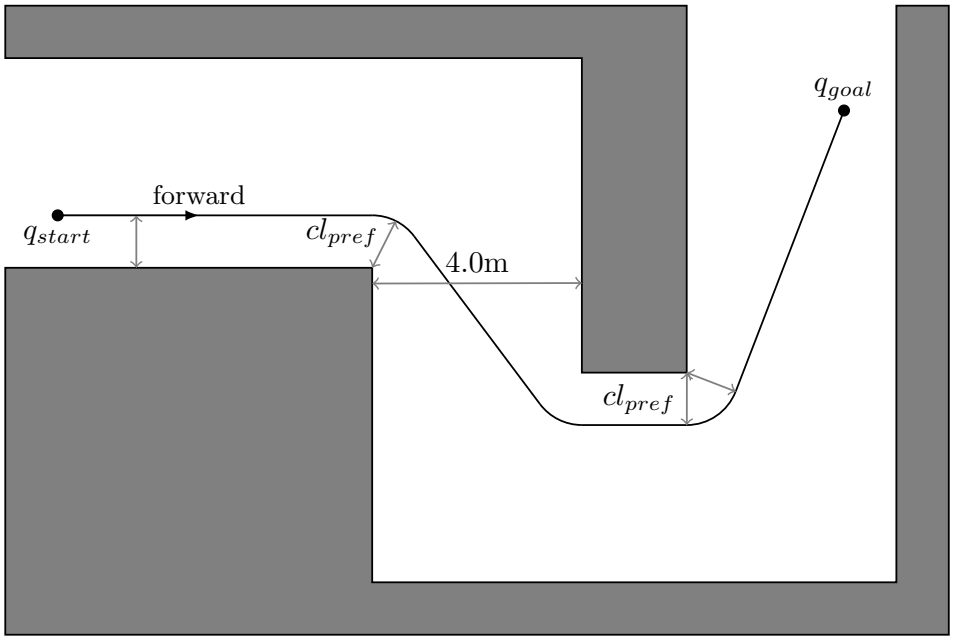
As mentioned in the section's introduction, our method takes a given global path as input. Global paths can be computed using global planning algorithms. A global planning algorithm computes whether a path free of obstacles exists between two points in a given environment, and computes such a path. To plan such a collision-free path, a description of the *navigable space* (also known as the *free space*) is required, denoted as C_{free}. When planning global paths, many criteria come into play such as path length, route preferences, minimum clearance, and movement constraints. In our case, the most important criterion is that a path must be traversable by a bicycle. However, this criterion is hard to satisfy on a global level. This problem is proven to be NP-hard for generic curvature-constrained shortest paths by Lazard, Reif, and Wang [LRW98]. Given the application of our algorithm, i.e. path planning for cyclists in urban environments, we will assume that a cyclist will be able to take every turn in a global path, and that the detailed planning of the non-holonomic movements can be solved on a local level. However, apart from non-holonomic movements, cyclists also have a preferred clearance to the side of the road or sidewalks, on which we will now briefly elaborate.

Astrom, Klein, and Lennartsson [AKL05] describe the concept of counter-steering: the process of briefly steering in the opposite direction when taking a turn. This helps the cyclist to enter a controlled state of falling, during which balance is maintained by the centrifugal forces caused by turning. A requisite for counter-steering is that there is some obstacle-free space available in the opposite direction of the turn. As a result, cyclists try to keep a safe distance from the side of the road, whereas social conventions and traffic-safety regulations require that they adhere to the side of the road. We will assume right hand driving in the remainder of this work. The cyclist unconsciously tries to keep balance between adhering to the roadside and keeping some room for counter-steering and keeping tolerance towards obstacles. This balance is found to be optimal at some fixed distances, depending on properties of the side of the road. A common property in urban environments is that the road is accompanied by a sidewalk, separated by a curb, in which case the cyclist will keep a distance from the curb of approximately 0.50m. In case there is no sidewalk or the curb is very low (a height of $\leq$ 0.07m.), the cyclist keeps a distance of approximately $0.25m$ [PB89] [MOW02, p. c. 4.1.5] [CRO06] [Bla12]. An overview of the different criteria for side preferences is given in Table 1. This is a criterion that can be handled efficiently by the MIRAN method [JCG13]. This method takes a preferred clearance, denoted as cl_{pref}, and the character width, and computes the shortest global path. The result is a corridor that is at least as wide as the character plus the preferred clearance on both sides of the character. Within this corridor, an *Indicative Route* is computed. The *Indicative Route* is a rough estimation of what the traversed

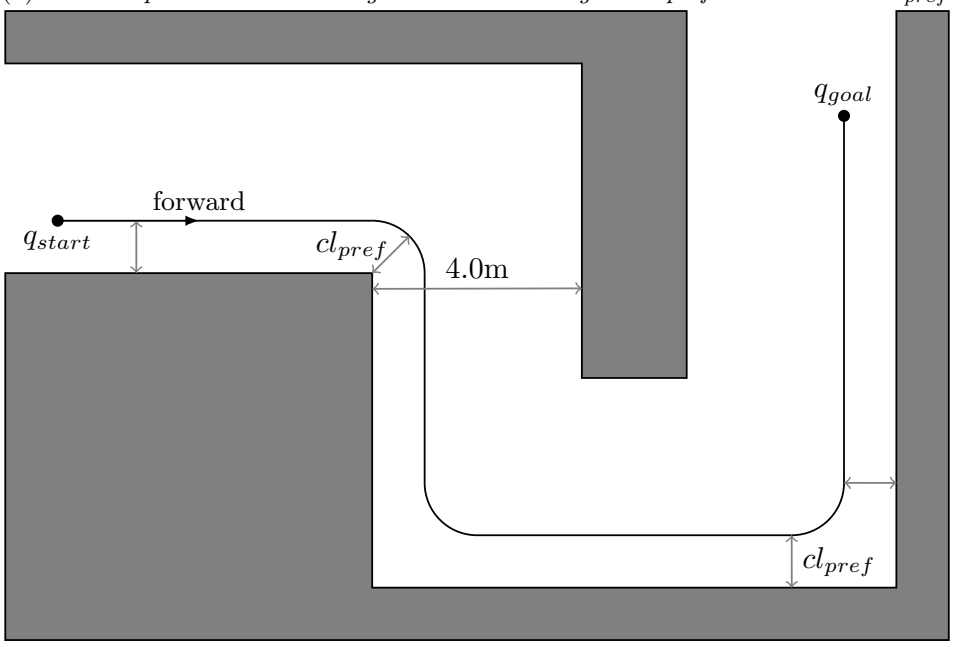
path should look like. An overview of several example *Indicative Routes* is shown in Figure 2. These exaggerated examples show various routes that can be created within the same environment using different settings for side preference. Figure 2a shows an example where the character does not have a side preference but has taken the shortest route possible instead. For cyclists this is an unlikely choice since they usually keep to the right, especially in corners with low visibility. Therefore, Figure 2b shows an example of a route traversed by a character that keeps a preferred clearance towards the obstacles on the right side of the road. However, by doing so the turns in the trajectory are quite sharp, thus requiring the cyclist to take the turns slowly. In reality, cyclists try to minimize braking in turns, causing wider curves in the traversed trajectory, as shown in Figure 2c. The relationship between speed and turn-radius is explained in Subsection 3.3.1.

Obstacle	**Distance**
Roadside without curbing (or max height of 0.07 m)	0.25 m
Roadside with curbing (default for urban environments)	0.50 m
High obstacles such as trees or lampposts	0.75 m
Walls	1.00 m

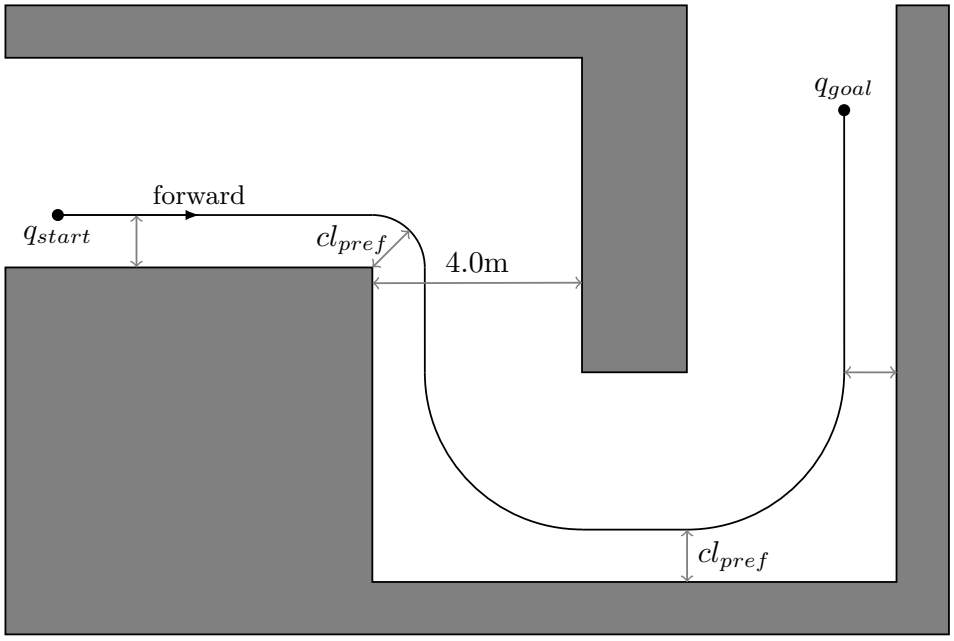
Table 1: *Cyclists maintain different criteria for side preference.*



(a) *Shortest path within the navigable environment given a preferred clearance* cl_{pref}



(b) *Side preference (with respect to obstacles)*



(c) *Side preference (with respect to the road width)*

Figure 2: *Several examples of global paths that keep a preferred clearance towards obstacles. The preferred clearance is denoted as* cl_{pref} *and the obstacles are marked in gray.*

3.2 Dealing with non-holonomic constraints

We now have an input path which we want to use for the local planner. First, however, we must select a kinematics model that can be used to compute the movements of the bicycle. The bicycle cannot move sideways and is an underactuated system, meaning that it has fewer controls than it has degrees of freedom. This is caused by rolling constraints imposed by the bicycle's wheels. A bicycle can achieve any orientation given enough room. However, because of these rolling constraints, an orientation can only be achieved by controlling the steer and moving forwards or backwards. Therefore, the bicycle is a non-holonomic system.

Our method must plan paths that satisfy these non-holonomic constraints, and, therefore, we select a kinematics model that can handle them. Many non-holonomic models exist [Whi99] [LSL98] [AKL05][Mei+07] [LaV06], some of which are even specifically tailored to bicycles. The model by Meijaard et al. [Mei+07] is considered the de facto choice for riderless bicycles but is fairly complex. Moreover, for cyclists the mass of the rider exceeds the mass of the bicycle by far [AKL05], thus changing the dynamics of keeping balance. The cyclist is able to keep balance without the internal balancing mechanisms of the bicycle. Therefore we can suffice with a much simpler model for simulating cyclists [Mei+07]. Astrom, Klein, and Lennartsson present such a model [AKL05]. The model is based on the generic bicycle model, as described in Subsection 1.3, but assumes that the steering axis is vertical, i.e. that head angle $\lambda = 90°$ and that trail $c = 0$. Furthermore they assume that the steering angle ϕ is not controlled by applying force to the handlebars but is controlled directly. Astrom, Klein, and Lennartsson's model incorporates the bicycle's roll (denoted as φ), i.e. the angle at which the bicycle is tilted sideways. However, the bicycle's roll angle φ can be directly derived from the steering angle ϕ via a transfer function [AKL05]. Therefore we can further simplify the non-holonomic model to ignore roll. The resulting model is identical to the car model as given by Dubins [Dub57], having a configuration q consisting of only a position (x, y), an orientation angle θ, and a steering angle ϕ, as depicted in Figure 3. The Dubins car model simplifies a car by applying the rolling constraints as if the wheels are in the center axis of the car. Therefore the model's geometry is identical to a bicycle's geometry.

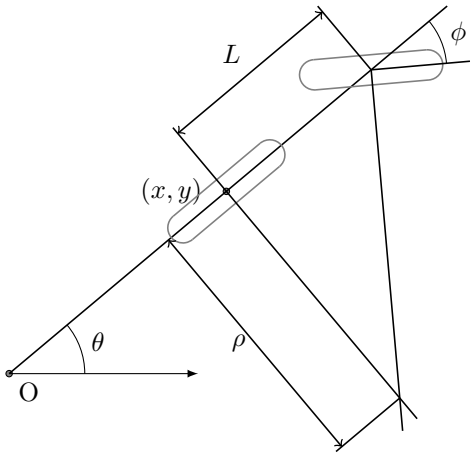


Figure 3: *An overview of the bicycle model as used for the implementation of Dubins' car model [Dub57]. 'L' denotes the wheelbase, 'θ' the bicycle's orientation, 'ϕ' the steering angle, 'ρ' the turn radius, and 'O' the origin.*

An implementation of the model is given by LaValle [LaV06, p. 595-599]. The model, though designed with four-wheeled vehicles in mind, is suitable to simulate a two-wheeled vehicle such as a bicycle since it does not consider the rolling constraints all four wheels independently. The car's steering angle ϕ controls the change in orientation over time. Dubins' model [Dub57] simulates the motion of the bicycle by assuming that there is a small time interval Δt during which the bicycle moves forward, approximately in the direction of θ. After each time interval, the bike's orientation θ is updated to account for the effect of the angle of the steer ϕ during the traversed trajectory. The position and orientation are updated using the following equation [LaV06, p. 597]

$$\begin{aligned} \dot{x} &= u_s \cos\theta \\ \dot{y} &= u_s \sin\theta \\ \dot{\theta} &= \frac{u_s}{L} \tan u_\phi, \end{aligned} \tag{2}$$

where u_s is the speed of the bicycle, u_ϕ is the steering angle, and L is the wheelbase, i.e. the distance between the two contact points of the wheels on the ground plane. The model's turning radius ρ is given by [LSL98] as

$$\rho = \frac{L}{\tan\phi}. \tag{3}$$

The steering angle ϕ is limited to the domain $-\frac{\pi}{2} < \phi < \frac{\pi}{2}$ as the bicycle cannot rotate in-place.

3.3 Local planning

The method operates on a given global path known as the *Indicative Route* [KGO09]. The properties of the *Indicative Route* ensure a desired clearance, which provides us with

a path that is feasible for a bicycle. As the bicycle moves along the *Indicative Route*, our method acts as a window, moving over a section of the global path, referred to as the local 'planning window'. The local planning window is based on the optokinetic reflexes of humans, also known as *Nystagmus* [Hon+68] [Bar93]. *Nystagmus* is the process of focusing the eyes on fixed points in an environment while moving. During this process a person chooses a fixed point and keeps this point in focus for a while using smooth eye movements. As the point moves towards the peripheral vision a new fixation point is chosen and the eyes move rapidly towards this new point. This reflex is also observed for cyclists by Vansteenkiste et al. [Van+13]. In our approach this focus point is called an attraction point (denoted as α_i). The attraction point is used as a target position within the planning window. Algorithm 3.1 provides an overview of the method's structure and an overview of the replanning step is provided in Algorithm 3.2.

Algorithm 3.1: Method overview. Within each new planning window a local path is planned and followed, until the goal is reached.

Input: Start and goal configuration q_{start} and q_{goal}, an *Indicative Route* A, a replanning threshold d, and a set C_{free} describing the navigable space.
Output: Traversed bicycle path Q

$\alpha_{last} \leftarrow q_{start}$ ▷ α_{last} is the last known attraction point.
$Q \leftarrow \varnothing$ ▷ Q is the bicycle path.
$q_i \leftarrow q_{start}$ ▷ q_i is the bicycle's configuration in each iteration.

while *distance*$(q_i, q_{goal}) > \epsilon$ **do**
  $\alpha_i \leftarrow$ computeAttractionPoint(q_i, A)
  if *distance*$(\alpha_i, \alpha_{last}) \geq d$ **then** ▷ d controls the planning window.
    checkAttractionPoint()
    if *replanLocalPath()* $=$ *SUCCESS* **then**
      $\alpha_{last} \leftarrow \alpha_i$
  $q_i \leftarrow$ followLocalPath()
  Q.add(q_i)

To follow the global path we use the MIRAN method [JCG13]. This method computes an attraction point α_i on the *Indicative Route*. Then the local planner plans a path towards α_i. The MIRAN method allows specifying a *look-ahead distance* σ which controls the size of the aforementioned replanning window. The attraction point is chosen such that it is within a direct line of sight which has a desired clearance cl_{min} towards obstacles, as shown in Figure 4. Furthermore, the attraction point is at a largest possible distance not exceeding σ (in meters) of the bicycle's current position. The lookahead distance σ is dependent on both the bicycle's speed s and the bicycle lane width. A possible model for this relationship between these factors is described by Vansteenkiste et al. [Van+13]. Since their model is tested only indoors on very narrow lanes, ranging from 0.1 m to 0.4 m, we will not use their model. Instead, we set lookahead distance σ to 15 m, based on the path length in their setup [Van+13].

Algorithm 3.2: The replanning method computes a set of Dubins' paths, and selects the shortest collision-free path.

Input: Intermediate start and goal configurations q_1 and q_2, speed s, and a set C_{free} describing the navigable space.
Output: Local target path P_{local}
Function: replanLocalPath()

```
Q_local ← {}                                  ▷ Q_local is the set of potential local paths.
φ ← computeMaxSteeringAngleForSpeed(s)        ▷ φ is the bicycle's steering angle.
ρ ← computeTurningRadius(φ)                   ▷ ρ is the bicycle's turn radius.
Q_local ← computeDubinsPaths(q_1, q_2, ρ)
if Q_local ≠ {} then
    sortPathsByLength(Q_local)
    for q_l ∈ Q_local do
        if pathIsCollisionFree(q_l, C_free) then
            P_local ← q_l
            return SUCCESS
    return COLLISION
return NO_PATHS_FOUND
```

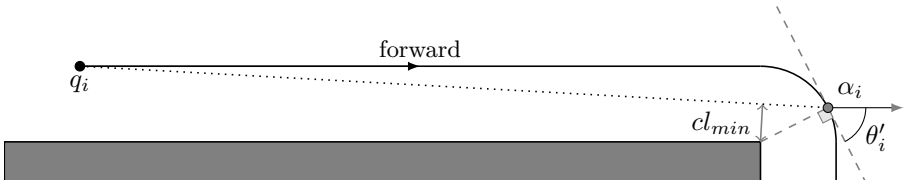


Figure 4: *The local planner derives the target orientation from the direction of the Indicative Route at the position of attraction point α_i.*

3.3.1 Computing local paths

Once the target position α_i is determined we can plan a curved path towards it. First however, we must define the target orientation θ'_i, i.e. the orientation in which the cyclist should arrive at point α_i. To determine the target orientation θ'_i, we compute the direction of the *Indicative Route* at the position of the attraction point α_i, as depicted in Figure 4. The *Indicative Route* computed by the MIRAN method [JCG13] is given as a discrete set of connected points. Therefore the target orientation cannot be computed in a continuous manner. The attraction point α_i is located either on an edge between two nodes of the *Indicative Route* or exactly at a node. Therefore the target orientation θ'_i can be derived from the direction from the attraction point α_i towards the next node.

Since we now have an initial configuration $q_1 = \{x_i, y_i, \theta_i\}$ and target configuration $q_2 = \{\alpha_{i_x}, \alpha_{i_y}, \theta'_i\}$, we can compute a path connecting these two configurations. We use Dubins' method [Dub57] to compute such a path since it can find the shortest path that takes into account the non-holonomic constraints of the cyclist, and takes into account that the cyclist can only move forward. Most other methods are tailored towards cars or autonomous robots and therefore assume a vehicle that can move both forwards and backwards. Dubins' method [Dub57] returns a path in constant time.

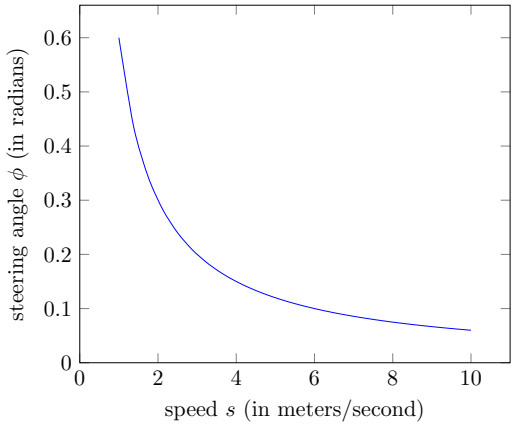


Figure 5: *A model of the relationship between speed and maximum steering angle.*

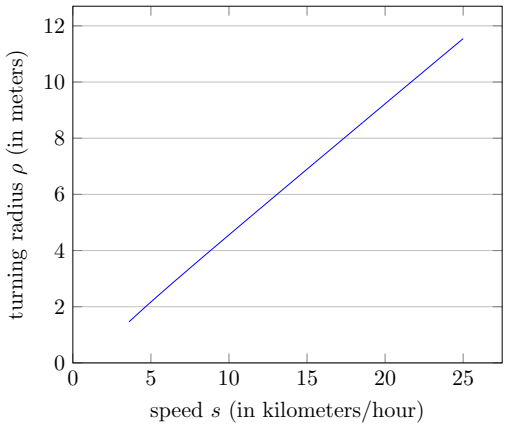


Figure 6: *A model of the relationship between speed and turning radius for a bicycle with a wheelbase 'L' of* 1.0 *m.*

To define the non-holonomic constraints the Dubins' method uses a configuration parameter ρ, which denotes the turn radius of the bicycle. The turn radius is derived from the steering angle, as described in Subsection 3.2. Cain and Perkins have investigated the relation between bicycle speed and steering angles and have experimentally confirmed their derivations an instrumented bicycle [CP10]. They provide an inversely proportional model of the relation between speed and steering, which we can use to determine the maximum steering angle ϕ for a given speed s, as displayed in Figure 5. The model

computes the maximum steering angle ϕ_{max} (in radians) using equation

$$\phi_{max} = \frac{0.6}{s} \tag{4}$$

where speed s (in meters/second) is limited to the domain $[1, 10]$. Combined with the equation for computing the turn radius by Laumond, Sekhavat, and Lamiraux [LSL98], the equation for computing the turning radius for a bicycle's given its speed is

$$\rho = \frac{L}{\tan \frac{0.6}{s}}. \tag{5}$$

A plot for a bicycle with a wheelbase of $L = 1.0$ m is given in Figure 6.

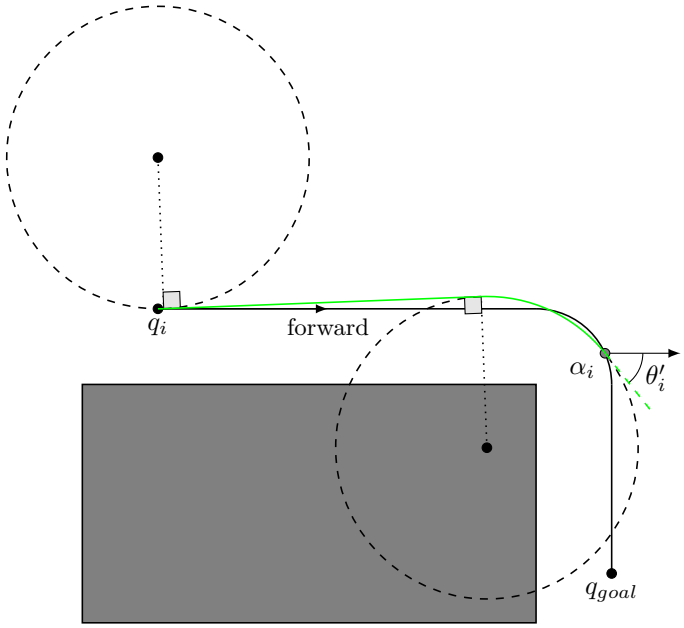
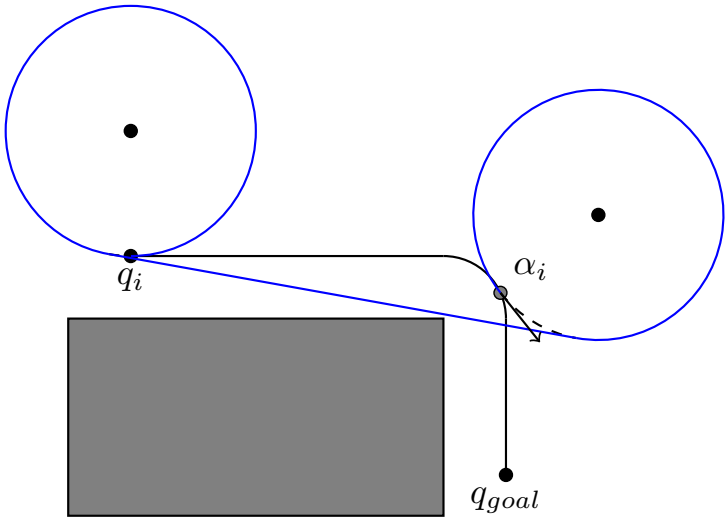
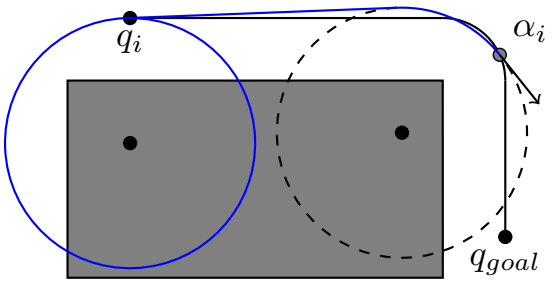


Figure 7: *A Dubins curve consists of at most three segments that have a constant curvature.*

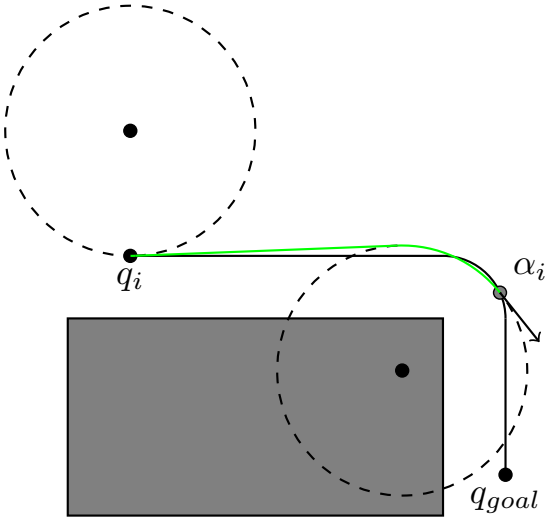
After the turn radius is determined, we can compute a Dubins curve between the two configurations q_1 and q_2. This results in a path consisting of at most three segments, which are either straight or circular of radius ρ (turning left of right). An example of a Dubins' path from configuration $q_1 = (q_i, \theta_i)$ to configuration $q_2 = (\alpha_i, \theta'_i)$ is shown in Figure 7 consisting of a left turn, followed by a straight section, followed by a right turn ending at the attraction point.



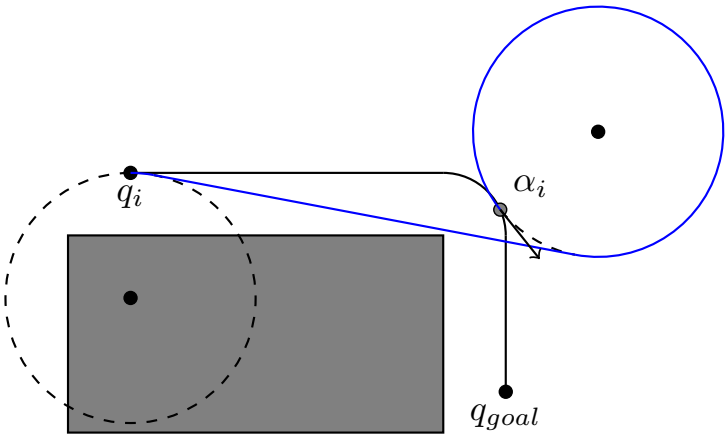
(a) *LSL*



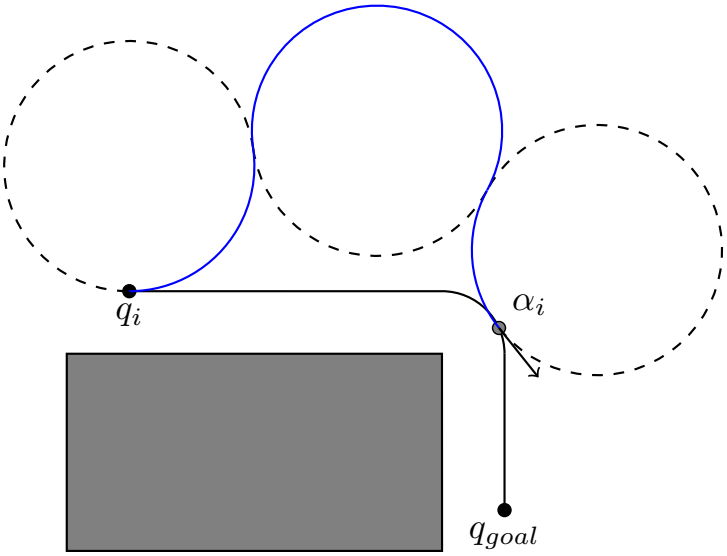
(b) *RSR*



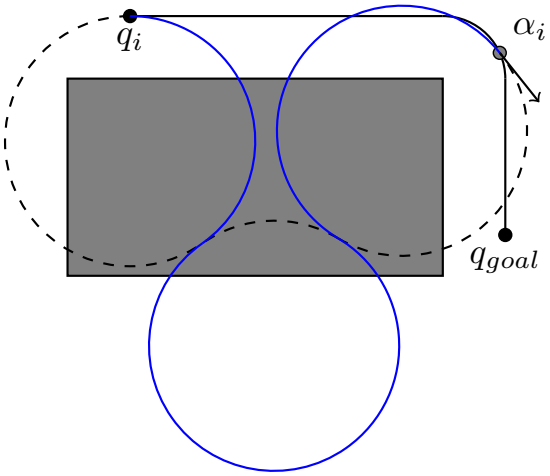
(c) *LSR*



(d) *RSL*



(e) *LRL*



(f) *RLR*

Figure 8: *All possible Dubins curves for a path between two configurations q_i and α_i. Each solution consists of three segments consisting of: Left turns (L), Right turns (R), or Straight segments (S). The unused segments of the turning circles are dashed. The shortest solution is marked in green.*

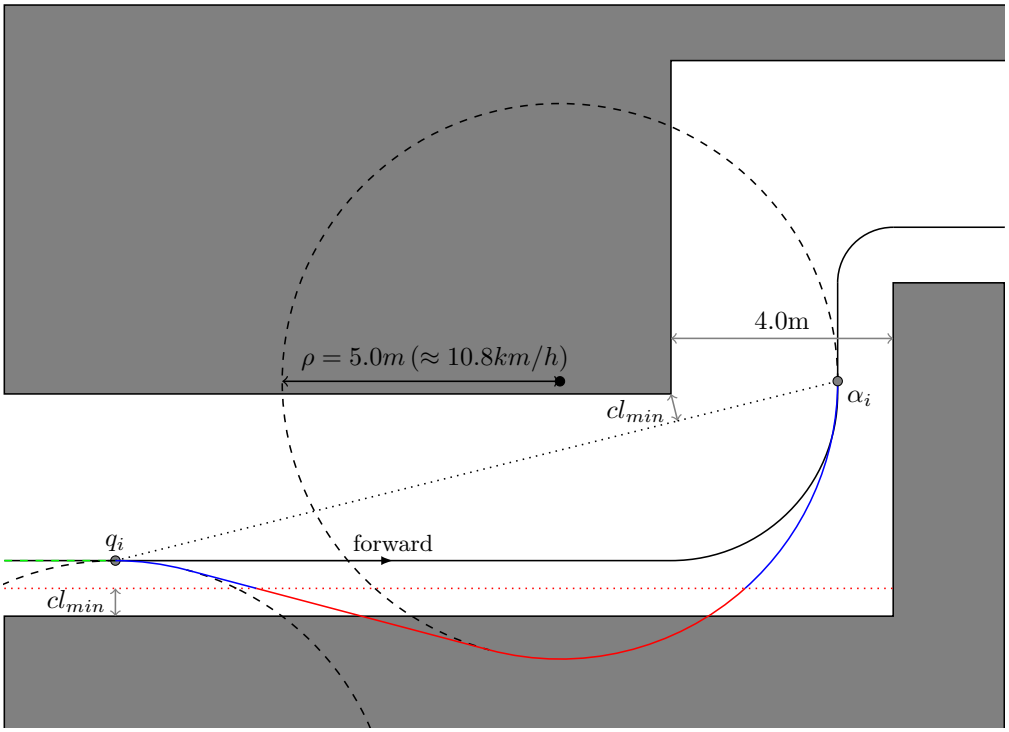


Figure 9: *An example of a potential RSL solution (marked in blue). The segments violating the minimum clearance, denoted as cl_{min} are marked in red. The path is rejected because it is not collision-free.*

The Dubins algorithm [Dub57] provides all possible combinations of three-segmented paths that lead to the target configuration, and selects the shortest one as the best solution. An example of a set of solutions is visualized in Figure 8. Since the Dubins method does not take into account obstacles, we must check each solution for collisions. If no collision-free solutions are available, such as in the example displayed in Figure 9, the path computed previously is followed and a braking force is applied to the bicycle to slow it down. This allows the bicycle to take sharper turns in the next replanning iteration. Once the bicycle has slowed down and has continued its planned curve around the corner, the attraction point α_i moves further along the *Indicative Route*, and a solution is found. This results in a path such as shown in Figure 10. Whenever a new solution is found during a replanning step while the bicycle is moving slower than its preferred speed, an acceleration force is applied until the preferred speed is reached.

3.3.2 Changing the goal orientation

When the goal position q_{goal} is almost reached, i.e. when it is in a direct line of sight from the bicycle's position, a situation might occur in which the exact orientation θ at the goal position q_{goal} becomes unfeasible. This can occur if the bicycle has deviated from the

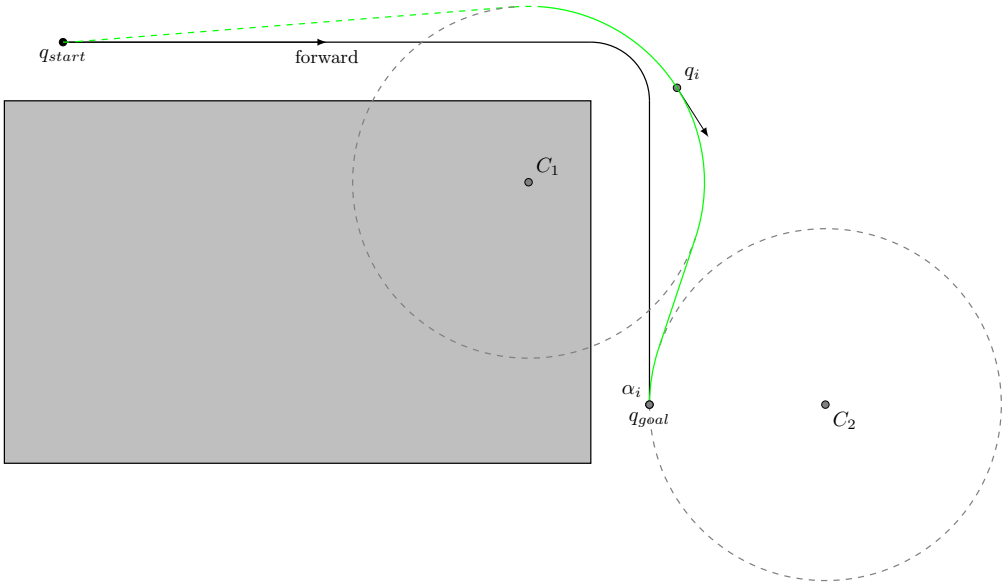


Figure 10: *The bicycle plans a path towards the Indicative Route when the attraction point α_i has moved around the corner.*

Indicative Route or the goal has been blocked temporarily. In these situations the bicycle is approaching the goal position from a different angle than expected, and computing the target orientation θ using the direction of the segment of the *Indicative Route*, as described in Section 3.3.1, will force the bicycle to take an unrealistic detour in order to reach that specific orientation. When the target orientation θ_i' becomes unfeasible we want to compute the shortest path that leads to the goal position. We will accept any orientation deriving from this new path. Since the cyclist can make either a left turn or a right turn, we will compute the shortest path for both of them. This results in two paths, each resulting in a different target orientation θ_i'. One of these paths is the shortest path and will be traversed until the goal position is reached.

The new paths and the orientations resulting from them are computed as follows. A circle with radius ρ is placed on both sides of the bicycle, tangential to the bicycle's forward vector. The centers are marked c_1 and c_2, as depicted in Figure 11. Next the tangent lines from goal position q_{goal} to circles positioned at c_1 and c_2 are computed using the Pythagorean theorem and Thales' theorem. This results in tangent points t_i and t_i' for each circle c_i perpendicular to the radius, $\{t_i, t_i'\} = q_{goal} \perp c_i$. We will assume that the goal position lies outside circles c_1 and c_2, otherwise the tangents cannot be computed.

We can compute angles α and β for both circles using the equations

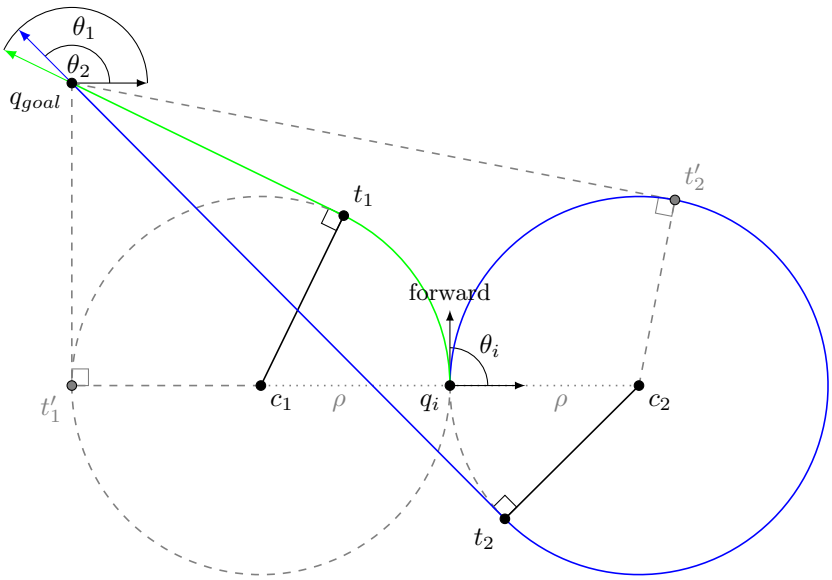


Figure 11: *New bicycle goal orientations are computed using the circle tangents. The shortest path from point q_i to point q_{goal} in this example is achieved using a left turn (marked in green). Should this path collide with an obstacle, the path using a right turn (marked in blue) is a viable alternative.*

$$\begin{aligned} \alpha_i &= \arcsin \frac{\rho}{|q_{goal} - t_i|} \\ \beta_i &= \arccos \frac{{q_{goal}}_x - {c_i}_x}{|q_{goal} - c_i|} \end{aligned} \tag{6}$$

where α_i is the angle between line segments $q_{goal}t_i$ and $q_{goal}c_i$ and, as depicted in Figure 12. β_i represents the angle between the line segment from the goal position q_{goal} to circle center c_i and the x-axis, and are computed by taking the arccos value of the normalized dot product of line segment $q_{goal}c_i$ and the x-axis.
Finally, we can compute the target orientation angles for a left and right turn using equation

$$\begin{aligned} \theta_1 &= -\beta_1 - \alpha_1 \\ \theta_2 &= -\beta_2 + \alpha_2, \end{aligned} \tag{7}$$

where θ_1 is the target orientation angle for a left turn, and θ_2 the angle for a right turn. We are using only two of the four tangent lines since two of them are pointing in a direction away from the goal, one for each circle. From the perspective of the goal position q_{goal} we are using the tangent line on the left of the circle at origin c_1, and on the right of the circle at origin c_2.

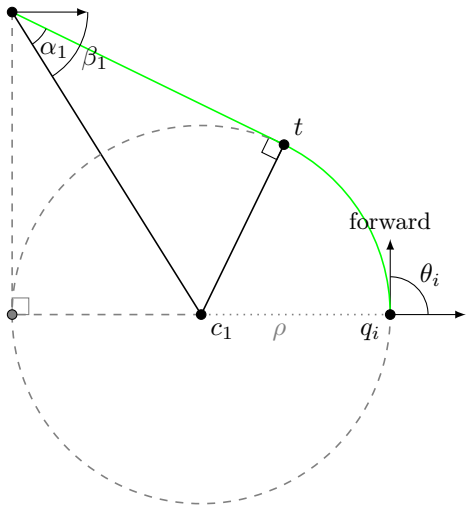


Figure 12: *The target orientation angle θ_i for both circles can be derived from angles α_i and β_i, which can be found using basic trigonometry.*

We can also find the orientation angles of both unused tangents from c_1 and c_2 using

$$\begin{aligned} \theta'_1 &= -\beta_1 + \alpha_1 \\ \theta'_2 &= -\beta_2 - \alpha_2, \end{aligned} \tag{8}$$

where θ'_1 is the orientation angle for the unused tangent point t'_1 on circle with origin c_1 and θ'_2 the orientation angle for tangent point t'_2 on circle with origin c_2. If a cyclist would be able to drive both forwards and backwards, we could achieve one of these orientations by traversing along the circle towards a tangent point and then moving backwards towards the goal position q_{goal}.

The resulting paths can be described as Dubins' *CSC* paths (*Curve, Straight, Curve*) in which the last C segment has a length $= 0$. Since this method produces two paths, we must compute the length of each path and select the shortest one as a potential candidate solution. If the path is collision-free the bicycle will store the path and will start following it. If the shortest path is not collision-free we can repeat checking for collisions for the second path, and use it when the path is collision-free. If neither of the paths are collision-free a braking force is applied to the bicycle and the bicycle will slow down, thus allowing for a wider search space. This step of the method is identical to the normal replanning method, as described in Section 3.3.1.

3.4 Planning around corners

When the bicycle is approaching a corner, the bicycle is also approaching the attraction point α_i. Since the position of the attraction point is based on the line-of-sight, the attraction point is always close to a corner, and will move away when the cyclist can see past an obstacle's corner. As a result, when a new attraction point is computed, its position might be close to the bicycle's position. In this case, the attraction point might be inside either of the turning-circles. An example of such a situation is shown in Figure 13, where the attraction point α_i lies within the right turning circle. In such cases, the attraction point can not be reached using a *CSC*-Dubins' path, but only via an unrealistic detour using a *CCC*-path, such as shown in Figure 8e. We do not want the bicycle to take unnatural detours.

There are two possible solutions for solving this problem. The first one is to slow down the bicycle such that the turning radius decreases until the attraction point α_i no longer lies inside the turning circle. This solution however is unfavorable since *(a)* it forces the bicycle to slow down excessively by braking, whereas cyclists tend to minimize braking in corners, and *(b)* because cyclists do not adhere to the usual side-preferences whilst taking corners (as explained in Subsection 3.1). Instead they try to optimize their speed by increasing the radius of the turn [AKL05].

To enable differences in side-preference in corners and to optimize for speed, we opt for a different solution: to determine an alternative, local attraction point α_l which can be reached using a *CSC* path. This will enable the bicycle to maintain higher speeds when approaching corners, and mimics the behavior of cyclists that have knowledge of the geometry of the environment around the corner. The local attraction point, denoted as α_l, lies around the corner of an obstacle, and can be computed using the following steps, as is visualized in Figure 13. First, a circle denoted as C_1 with radius ρ is placed tangential to the bicycle's position. The circle's center is projected on the *Indicative Route*. It is projected onto the same segment of the *Indicative Route* as which the attraction point is located. A new helper circle, denoted as C_{helper} with radius ρ is placed along the *Indicative Route*, tangential to the projected center of C_1. Next, the intersection points between circles C_1 and C_{helper} are computed, after which the distance between these intersection points, denoted as γ, is found. Finally, a new turning circle, denoted as C_2, is placed with further along the *Indicative Route*, with a distance γ between its center and the center of C_{helper}. The local attraction point α_l is at the tangent of circle C_2 and the *Indicative Route*. The resulting path, formed by circles C_1 and C_2, is a *CSC* Dubins path, with a straight segment of zero length. An example of such a path (marked in green) is shown in Figure 13 along with helper circle C_{helper} (marked in blue), and the distance between the intersection points of C_1 and C_{helper}, denoted as γ. An algorithmic overview of the solution is given in Algorithm 3.3.
The combined steps as described in Subsection 3.3.1 and its subsections form the basis of our path following method. So far, the method is able to follow a given path as closely as possible whilst avoiding static obstacles, thus providing an adequate solution for Research

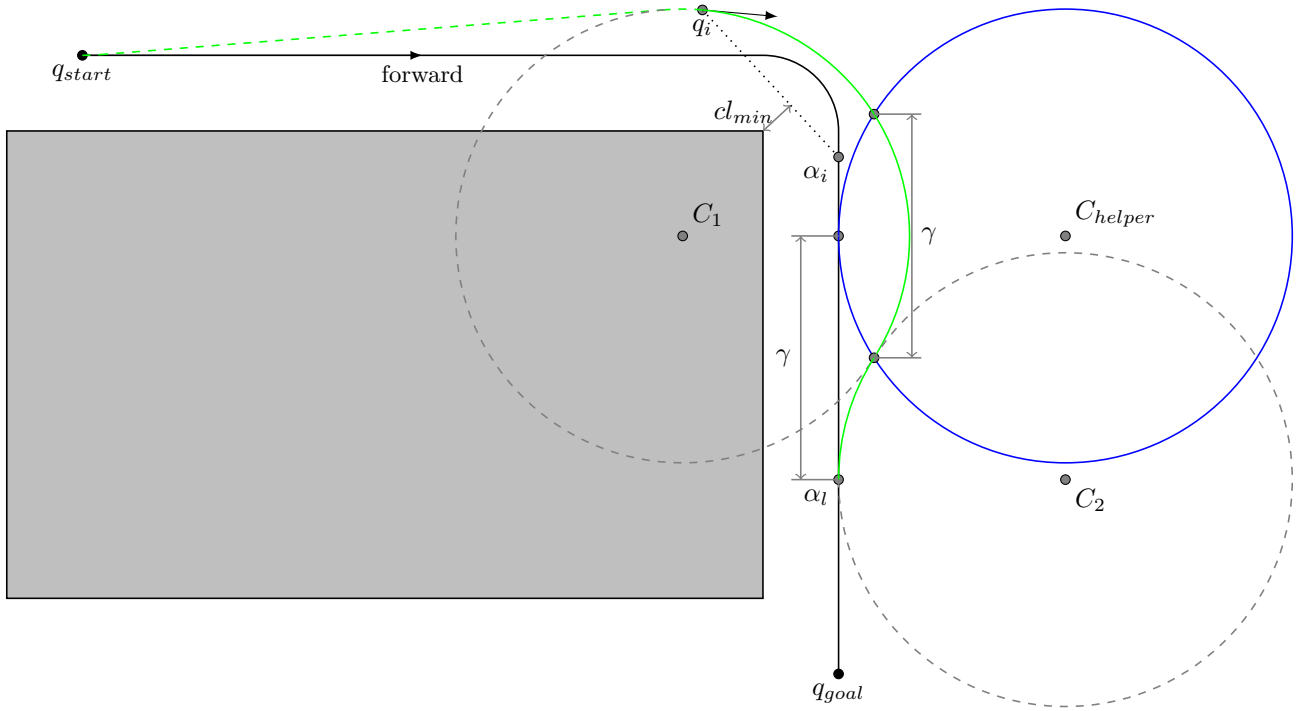


Figure 13: *When the attraction point α_i is only reachable via a detour, a local attraction point α_l further along the Indicative Route is computed.*

Algorithm 3.3: The lookaround method computes a new local attraction point, which allows the bicycle to optimize speed, and enables it to slightly deviate from the *Indicative Route* without taking an unrealistic detour.

Input: Current configuration q_i and θ_i, an *Indicative Route* A, and turning radius ρ.
Output: Local attraction point α_l, and a *CSC* Dubins path.
$C_1 \leftarrow q_i +$ createTurningCircle(θ_i, ρ)
$C_p \leftarrow$ projectOnto(A, C_1)
$C_{helper} \leftarrow$ createTurningCircleTangentialTo(A, C_p, θ_i, ρ)
$i_1, i_2 \leftarrow$ computeCircleIntersections(C_1, C_{helper})
$\gamma \leftarrow |i_1 - i_2|$
$C_2 \leftarrow$ moveCircleAlongIndicativeRouteByDistance(C_{helper}, A, γ)
$\alpha_l \leftarrow$ computeTangentPoint(C_2,A)
$CSC \leftarrow$ computePathFromTurningCircles(C_1,C_2)

question 1 and Research question 2a, as described in Subsection 1.1. For experimentation and comparison purposes we have also implemented alternative path following methods, which we will discuss in Section 4.2. However, we are still left with the problem of dealing with dynamic obstacles, as described in Research question 2b and Subsection 2.1. In the next section we will discuss how we can incorporate collision avoidance with dynamic obstacles into our model.

3.5 Collision avoidance

Dynamic obstacles, such as other cyclist, cars, and pedestrians are frequently encountered by cyclists in everyday traffic situations. We want our method to support collision avoidance between multiple cyclists as well as between cyclists and other types of road users. Methods such as ORCA [Ber+11] by Van den Berg et al. solve this problem for collision avoidance between pedestrians. ORCA has also been applied to non-holonomic robots [AM+13], and therefore we think that ORCA is a good candidate for collision avoidance between both cyclists and other types of road users. Alonso-Mora et al. present their method ORCA-NH [AM+13], which supports non-holomonic constraint for robots with a differential drive. Bicycles however cannot rotate in-place. We adapt the ORCA algorithm in such a way that its results can be used by non-holonomic vehicles such as bicycles. Using an existing algorithm, such as ORCA, also shows additional benefits. We want our method to support collision avoidance between both cyclists and pedestrians. Therefore we use this existing method to inherit the collision avoidance behavior for pedestrians. ORCA computes a velocity obstacle for each character and chooses a new velocity vector V_A^{new} for each of them such that collision-free movement is guaranteed *"for at least a fixed amount of time into the future"* [Ber+11].

However, ORCA is not tailored to non-holonomic motion but to holonomic motion. Therefore we cannot directly use the velocity vector V_A^{new}. We must satisfy the non-holonomic constraints of the bicycle as well as the relationship between speed s and steering angle ϕ_{max}, as given by Equation 4. To achieve this, first we compute the angle between bicycle forward vector and velocity vector V_A^{new}. The bicycle's steering angle ϕ is set to this angle as long as it does not exceed ϕ_{max}. If ϕ is greater than ϕ_{max} the bicycle's steering angle is set to ϕ_{max}. To account for required changes in speed, the length of vector V_A^{new} is compared to the current speed s. If the difference is positive the cyclist must accelerate, and the desired acceleration force is applied to the pedals. Otherwise the cyclist must brake, and a braking force is applied. The speed resulting from the ORCA-solution is not dependent on the cyclist's preferred speed. This property is inherent to the ORCA method [Ber+11] and allows multiple cyclists to form a flow. An example of a solution for two bicycles moving in opposite directions is shown in Figure 14.

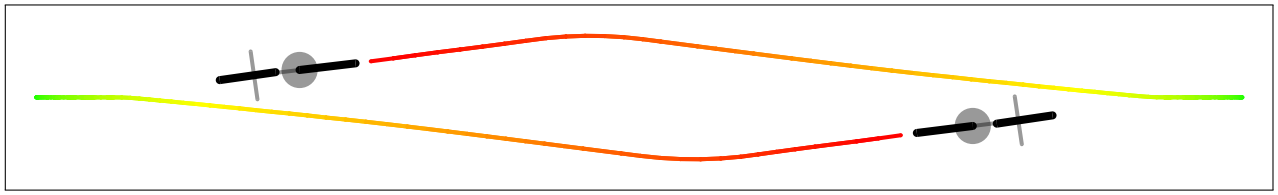
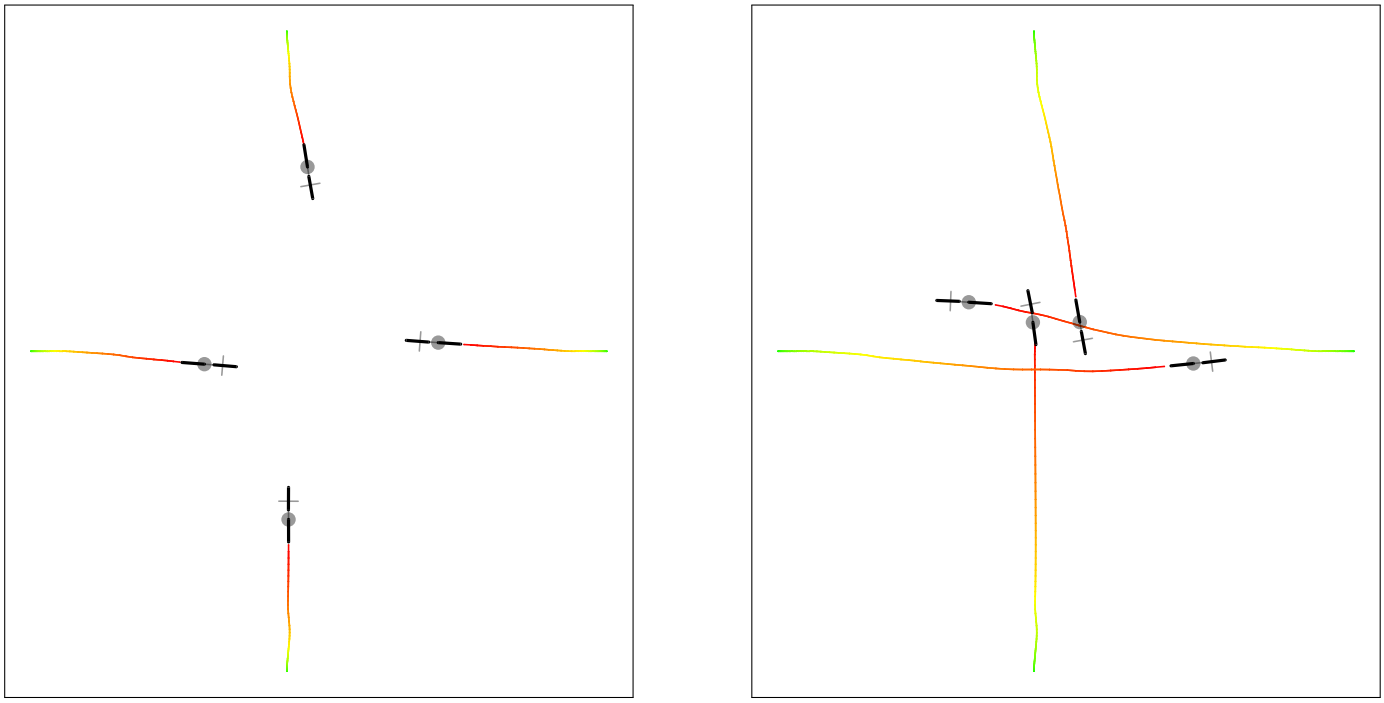
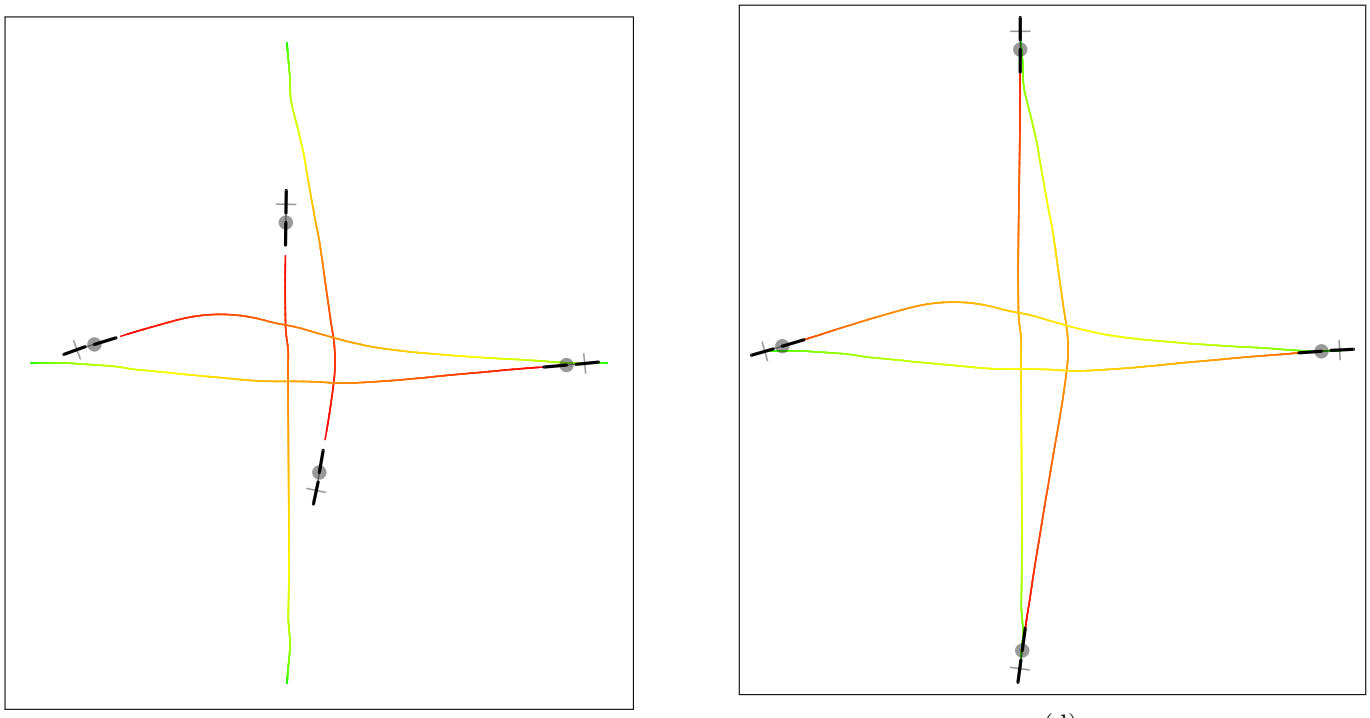


Figure 14: *Two cyclist have successfully avoided a collision. The color of the traversed curve indicates the time.*

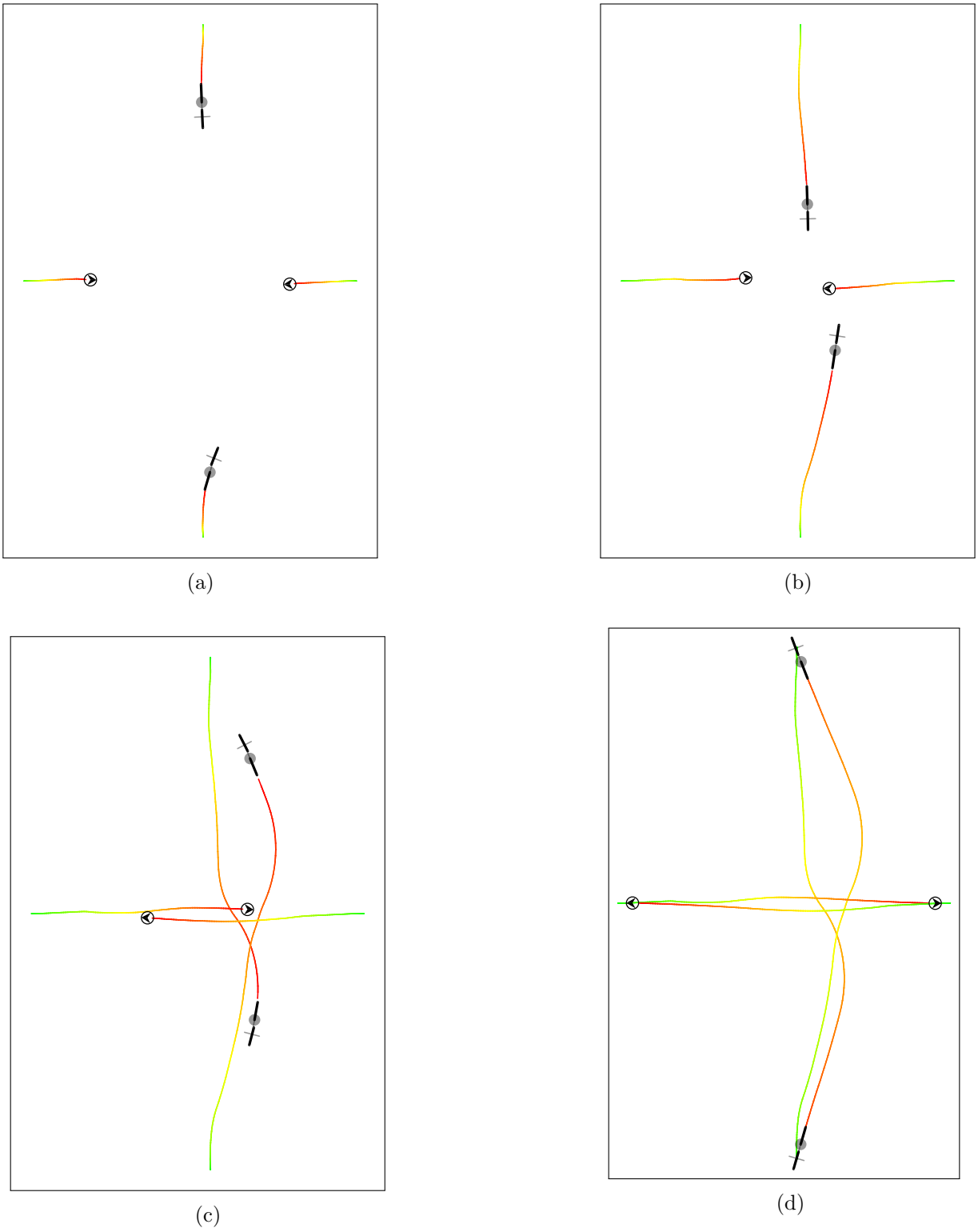


(a) (b)



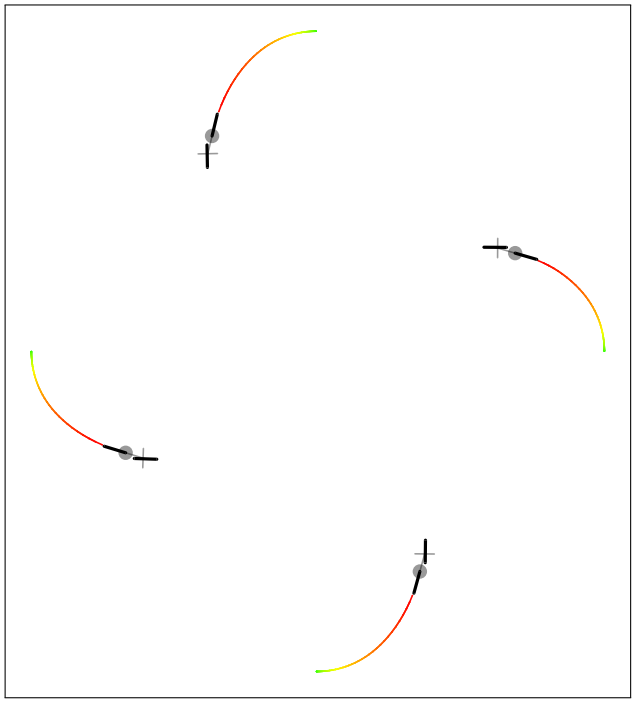
(c) (d)

Figure 15: *Four cyclist have successfully changed positions and avoided collisions.*

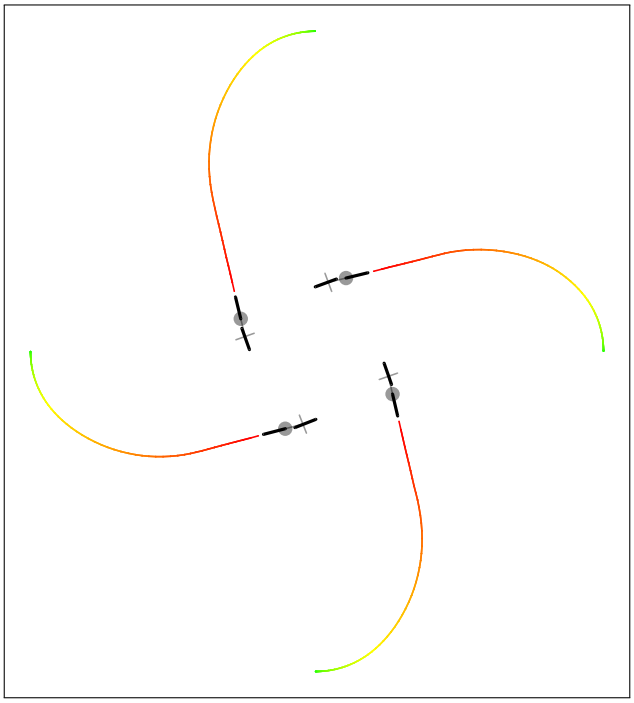


(a) (b) (c) (d)

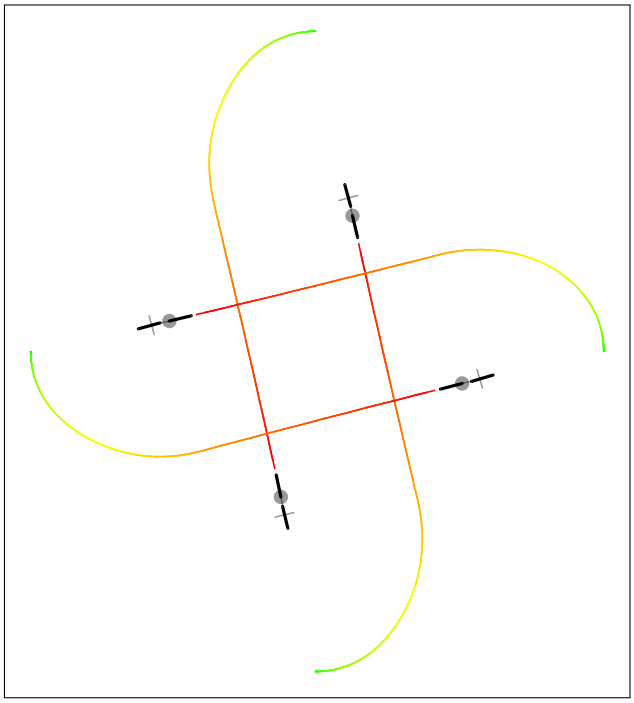
Figure 16: *Two cyclist have successfully changed positions and avoided collisions with two pedestrians crossing their path.*



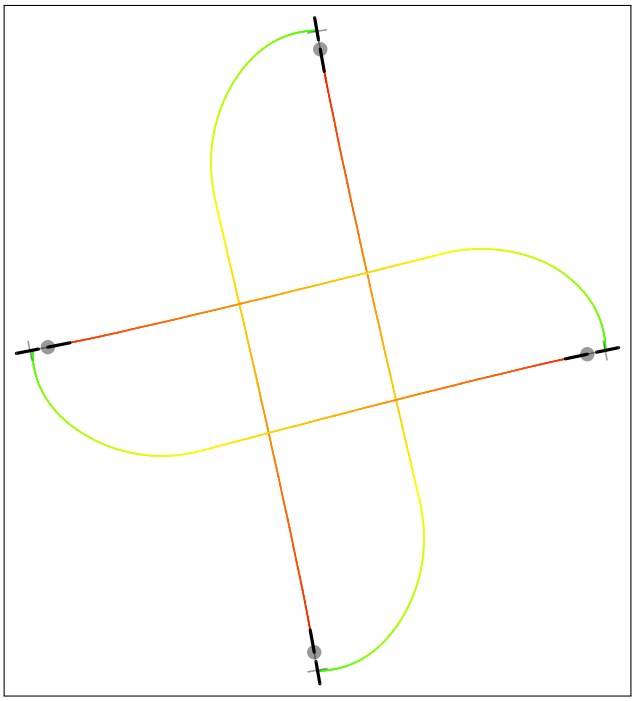
(a)



(b)



(c)



(d)

Figure 17: *Four cyclist start in a different orientation, and have successfully changed positions and avoided collisions.*

If there are no other cyclists or pedestrians within close proximity of the current bicycle, we do not have to consider collision avoidance. In this case the velocity vector V_A^{new} computed by ORCA becomes a null vector and we can ignore it. This approach enables an easy integration of ORCA into our method while honoring the non-holonomic constraints, but also allows other methods to be used when desired.

4 Implementation

As explained in Subsection 3.1, our method operates on a given global path. Therefore we will now briefly explain the method chosen for computing global paths. We want the implementation of our method to support the side-preferences typical to cyclists, as described in Subsection 3.1. As mentioned in Subsection 3.1 and Subsection 3.3 we have selected the MIRAN method [JCG13] for creating the global paths, on which our method will operate. Since an implementation of the MIRAN method is provided for in the *Crowd simulation framework of Utrecht University* [GT13], we have implementated our methods using this framework. Another positive aspect of using this framework is its implementation of the ECM navigation mesh [TCIG11]. The ECM *Explicit Corridor Map* enables us to check whether the paths created by our method are collision-free. The ECM navigation mesh can check whether a line-segment is collision-free in linear time in the number of obstacle vertices. Our method is based on Dubins curves [Dub57], as explained in Subsection 3.3.1. Walker [Wal08] provides an efficient implementation for computing the complete set of Dubins Curves. The code was published in conjunction with Walker's Phd thesis [Wal11]. The complete set of all 6 Dubins paths is computed in constant time.

4.1 Replanning criteria

In Section 3 we have explained how our method follows a given global path, known as the *Indicative Route*, using two repeating steps. These steps are *replan* and *follow*, as shown in Section 3. In the *replan* step, a local path is planned towards an attraction point α_i. In the *follow* step, the local path is followed by the bicycle using Dubins' car model [Dub57] [LaV06, p. 595-599] for dealing with the bicycle's non-holonomic constraints. The non-holonomic model and the Dubins curves are defined in continuous form, and must be integrated. For feasibility of implementation, we discretize the non-holonomic model by choosing a Δt value such that Equation 2 can be easily computed. We have set $\Delta t = \frac{1}{30}$ s, such that real-time simulations can be run smoothly at 30 frames/second. The value of 30 is also chosen because of the framerate of video footage which we have used to validate our model, which we will explain in Subsection 5.1. The *replan* step, as shown in Algorithm 3.2, discretizes the local path using the same Δt value. The *replan* step itself however, could in theory be executed during every time step. However, the local path does not change much after a single time step since the bicycle will not have moved a lot. Therefore we will not perform replanning in each time step, but only in some cases, which we will now explain. The main reason for replanning is when the distance to the

attraction point α_i increases. Aside from this main reason there are several special cases for which the replanning step is invoked, which are listed below.

4.1.1 Initial local planning

When no local path has been computed previously the *replan* method is invoked. This occurs when the bicycle is initially created and receives a global path.

4.1.2 Local planning window

Since the method uses as planning window, as described in Subsection 3.3, we must *replan* when the window moves. The window is defined by the attraction point α_i, as explained in the section's introduction. To simulate the Optokinetic Nystagmus and to avoid constant replanning, the *replan* step is only invoked when the attraction point α_i has moved. We set the distance threshold $d \leq 0.5$ m for the maximum distance between α_i and the end point of the existing local path.

4.1.3 Local path is old

When the local planner does not find a collision-free path towards the new attraction point α_i, a flag is set. The flag indicates that the local path computed previously is old, and that a new solution must be found as soon as possible. When the flag is set, a braking force is applied, thus slowing the bicycle down, as described in Subsection 3.3.1. Additionally, the *replan* step will be invoked during each successive iteration until a new solution is found.

4.1.4 Goal position is found

When the goal position becomes visible from the bicycle's position, three re-planning criteria apply, which are listed below.

(a) Local path does not reach goal position
When the attraction point α_i reaches the goal position, the distance d from the goal position to the end point of the existing local path might be less than 0.5 m. As explained in Subsection 4.1.2, re-planning is performed when $d > 0.5$ m. However, since the local path does not lead to the goal position a new *replan* step is performed.

(b) Goal position is blocked
When the bicycle is approaching the goal position, the goal position itself might be (temporarily) occupied by obstacles such as other cyclists or pedestrians. The goal position is blocked if a circle with radius r identical to the radius of the bicycle at the goal position contains other characters. In this case a timeout is activated, after which the bicycle slowly stops. Additionally, the *replan* step will be invoked during each successive iteration.

(c) Goal position has cleared
When the goal position was blocked previously by other cyclists or characters, which have left the goal position, the above timeout is cleared. Additionally a new local path towards the goal position is planned.

4.2 Alternatives for local planning

So far, we have developed a fairly non-trivial model to solve the problem of following a given path. Our model is guaranteed to satisfy the non-holonomic constraints and to avoid collisions with obstacles. However, we will compare it against other methods to determine how well it stands out. We have selected two alternative methods that can both deal with non-holonomic constraints, or can be easily modified to do so.

4.2.1 Pure Pursuit

The first method selected is Pure Pursuit [Wal+85] [Cou92]. We have selected this very simple method in order to check whether our method is not overly complex, and whether our results could also be achieved in much simpler ways. The Pure Pursuit method is a following algorithm that follows a moving point using an agent. In the holonomic version of this method, the orientation of the agent is always kept into the direction of the point. In our version, instead of instantaneously adjusting the bicycle's orientation, we can only control the orientation by steering. Therefore, we control the steering angle such that it steers towards the moving point, which is in our case the *attraction point* α_i. This is achieved by computing the angle between the bicycle's forward direction and the direction towards the attraction point. The bicycle's steering angle ϕ is set to this angle. Since not all steering angles are considered realistic for normal driving, the maximum steering angle is limited. The maximum angle is affected by the bicycle's speed, according to the model by Cain and Perkins [CP10], as shown in Figure 5.

4.2.2 RRT

We have chosen Pure Pursuit as an alternative method to verify if the path following problem can also be solved by a simple algorithm. However, Pure Pursuit might not be able to solve complex situations, in which case we cannot perform a fair comparison. Therefore, we have also selected the RRT *(rapidly exploring random tree)* method [KL00], whose principle is well known for its ability to solve complex planning problems [Kav+96] [LK01] [FDF02] [LaV06] [Kuw+08] [SWT09] [Per+12] [PA14]. RRTs and PRMs *(probabilistic roadmaps)* [Kav+96] can be constructed using various sampling heuristics, such as Medial axis based sampling [HK00] and Reachability-Guided sampling [SWT09]. Likewise, we have also chosen for a biased sampling heuristic. For each sampling iteration the steering angle ϕ and a time duration Δt are sampled. The time duration is sampled in the domain $[1, 5]$, such that the bicycle has enough time to reach the target configuration within its planning window. The steering angle is sampled in a dynamic range, which depends on the distance between the bicycle's current position and the attraction point

Distance to attraction point α_i	**Angle towards α_i**	**Steering angle ϕ sampling domain**
> 25 m	$[-1, 1]^\circ$	$[-1, 1]^\circ$
> 20 m	$[-2, 2]^\circ$	$[-2, 2]^\circ$
> 15 m	$[-3, 3]^\circ$	$[-3, 3]^\circ$
> 10 m	$[-4, 4]^\circ$	$[-4, 4]^\circ$
> 7 m	$[-7, 7]^\circ$	$[-7, 7]^\circ$
> 4 m	$[-11, 11]^\circ$	$[-11, 11]^\circ$
≤ 4 m	$[-\infty, \infty]^\circ$	$[-20, 20]^\circ$

Table 2: *The RRT implementation uses a dynamic sampling range for the bicycle's steering angle, based on the attraction point α_i.*

α_i, as well as the angle between the bicycle's orientation and the direction towards the attraction point α_i. The range of steering angles is shown in Table 2. After a steering angle ϕ has been sampled the corresponding maximum speed is computed according to Cain and Perkins's model [CP10], as shown in Figure 5. This biased sampling strategy ensures that when the attraction point α_i is far away from the bicycle, the maximum steering angle is limited, thus avoiding unnecessary detours and enabling higher speeds. When the bicycle is approaching the attraction point α_i, e.g. when approaching a corner, the larger sampling range allows the RRT method to reach a wider set of configurations.

4.3 Multiple cyclists and other road users

Collision avoidance between cyclists is implemented using the ORCA method, as described in Subsection 3.5. The ORCA method approximates each character using disks. Karamouzas et al. [KGO09] define the radius of a walking character at $r = 0.25$ m. We use the same value for the size of the character riding on the bicycle, but add an additional radius of $r_b = 0.6$ to account for the size of the bicycle itself. This results in a disc with a diameter of 1.7 m. This is based on the wheelbase of $L = 1$m and a radius of $r_i = 0.35$ m for both wheels [AKL05], as described in Subsection 1.3. The ORCA method itself takes the velocity of the discs in relation to their radius into account. Since the ORCA method supports collision avoidance between discs of different sizes, this combination enables cyclists and pedestrians to avoid each other. A disadvantage of this approach is however that traffic regulations are not incorporated. Therefore a cyclist might for example overtake another cyclist either on the left or the right side. This depends on the surrounding geometry, e.g. when there is more room on the right side of the cyclist being overtaken. Additionally, bicycles approaching each other from opposing directions might not always keep right, but cross each other on the wrong side of the road instead.

5 Validation

To validate the results of our algorithm, we want to compare the results of our algorithm to data of real cyclists, and check whether they are statistically significantly different. To perform such a comparison we have captured data of cyclists from video footage. We have captured data of several basic scenarios, such as: taking a right turn, taking a left turn, or following a straight section.

5.1 Data collection

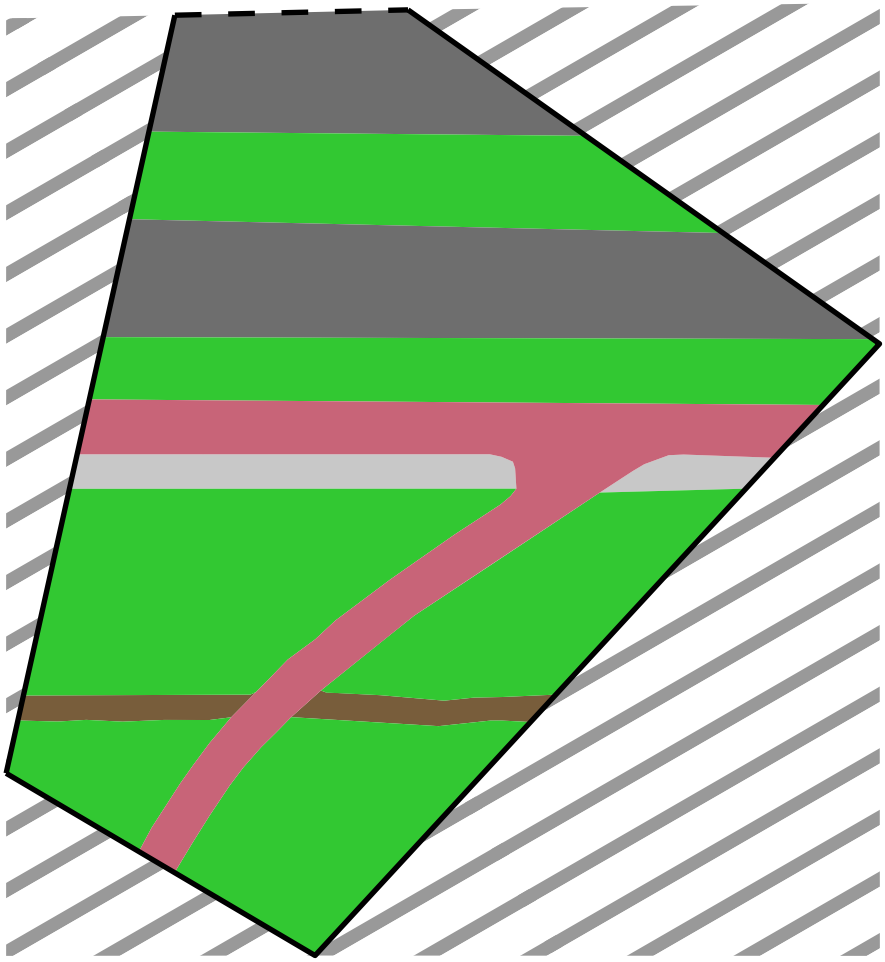
We have extracted bicycle trajectories from video footage shot at two locations on the campus of Utrecht University. Each bicycle's trajectory is extracted by manually tracking the position of the bicycle's rear wheel over multiple frames. The point at which the rear wheel contacts the ground is converted to a 3D point on the ground plane using 2D to 3D reconstruction also known as back projection (referred to as "back projection"). Each 3D point is stored along with a time stamp of the current frame, which enables us to compute the time it took the cyclist to traverse its trajectory and thus its average speed. For back projection to work, the cameras must be calibrated, i.e. the location and orientation of each camera with respect to the ground must be computed, as well as the camera lens properties such as focal point and distortion. Performing the camera calibration is done by taking measurements of identifiable road segments within the visible camera frustum. An identifiable segment means that the segment can be accurately identified both in the video footage as well as in the real world, e.g. using lamp posts, road markings, or paving stones. An overview of the segments used for the camera calibration is show in Figure 19. The camera calibration is performed using the QtCalib program [ZL12], which computes the camera intrinsics and extrinsics based on the road segments as shown in Figure 19 and specifying the dimensions of each segment. The calibration is performed using Tsai's algorithm [Tsa87]. The bicycle trajectories are extracted using CVTracker, a program created in an earlier work [RG14], which uses OpenCV [BK08] to perform the back projection using the camera configuration computed earlier. An overview of the characteristics of each filming location is shown in Table 3. From each environment 30 trajectories were extracted. Table 4 lists the types of trajectories extracted from each filming location. Using this method of back projection for extracting data from the video footage also enables us to model the road and terrain layout as well as obstacles. Since the same camera configuration is used for creating these models and extracting the bicycle trajectories, they are both in the same coordinate frame. This allows us to run our method on exactly the same environment as from which the data are captured.

Location	**Top**	**Bottom**	**Left**	**Right**
BBG	61.83 m	26.90 m	69.56 m	62.43 m
LANG	184.09 m	23.21 m	90.31 m	188.86 m

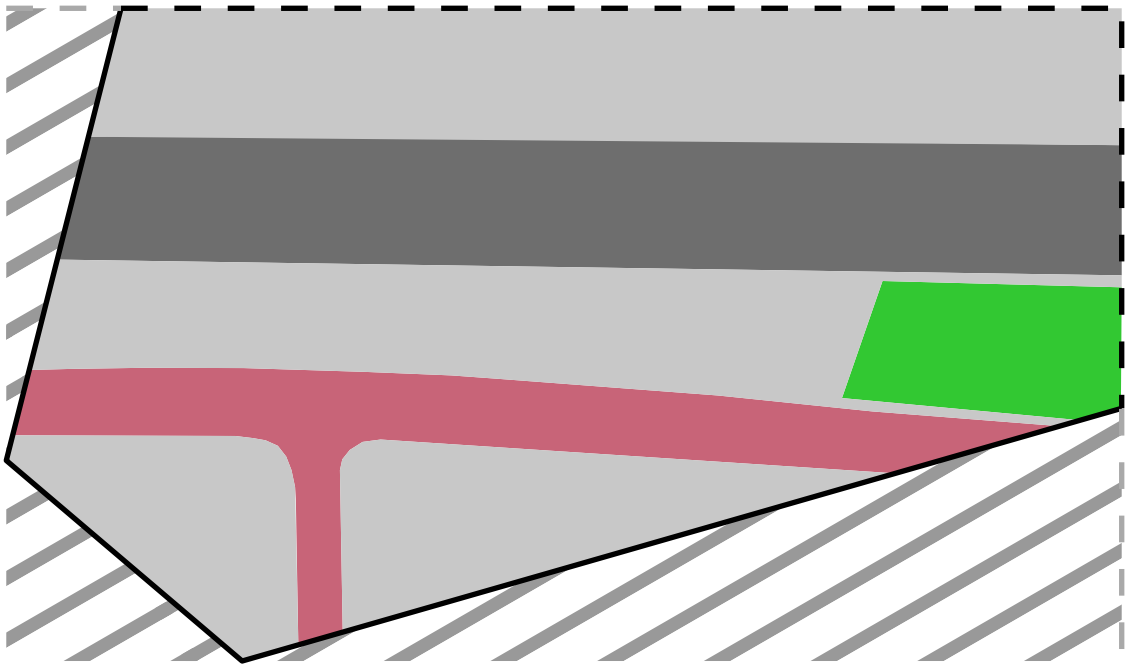
Table 3: *An overview of the dimensions of the visible area for each video footage location*

Location	**Left turn**	**Right turn**	**Straight on**	**Total**
BBG	3	21	6	30
LANG	13	7	10	30
Total	16	28	16	60

Table 4: *An overview of the trajectories in the data set. The type of turn indicates how many cyclists took such a turn in each environment. Cyclist that only took the main road are indicated as 'Straight on', whereas cyclists taking a left or right turn following the branch are marked as 'Left turn' or 'Right turn' accordingly.*

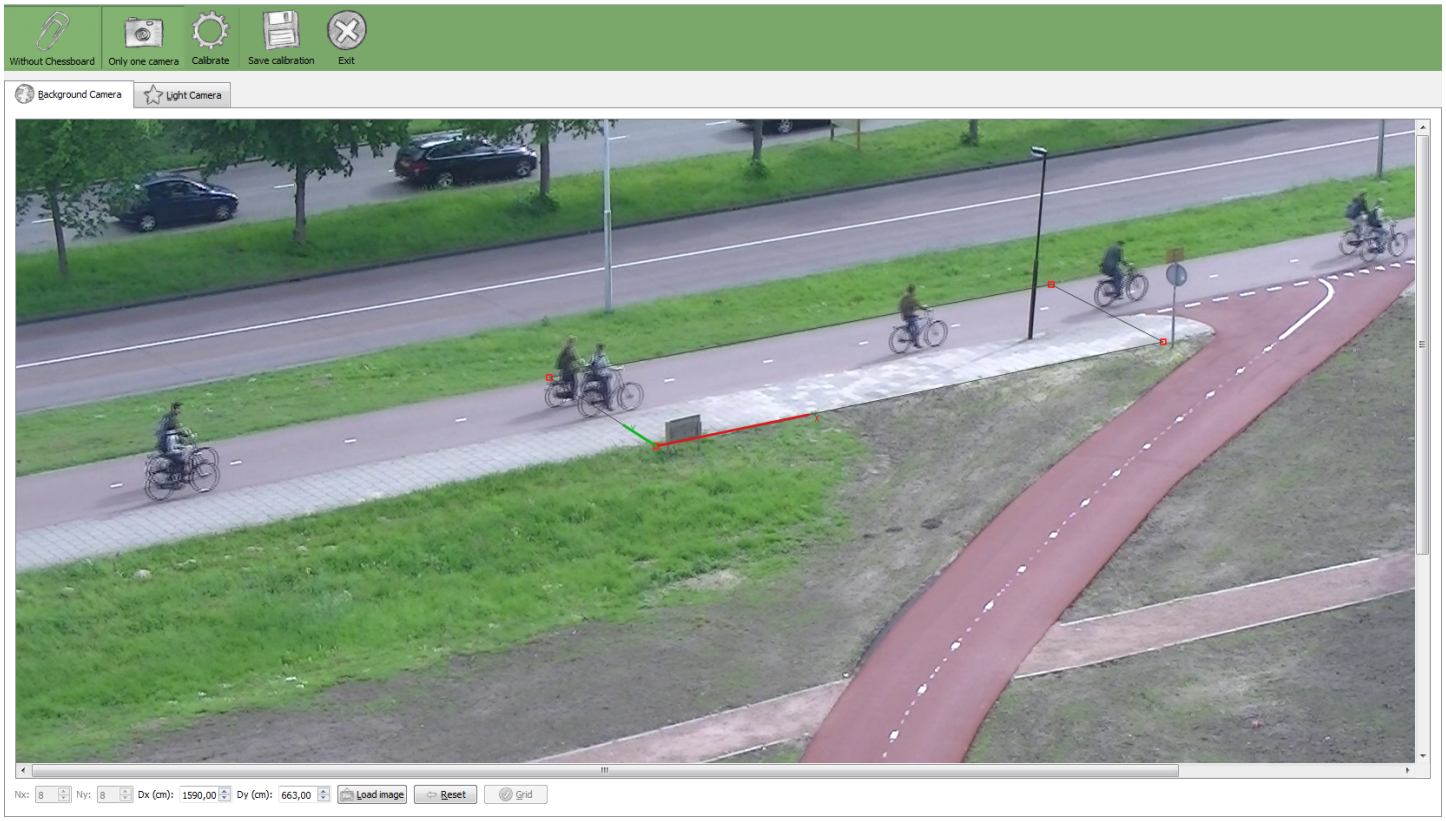


(a) *Location: BBG*

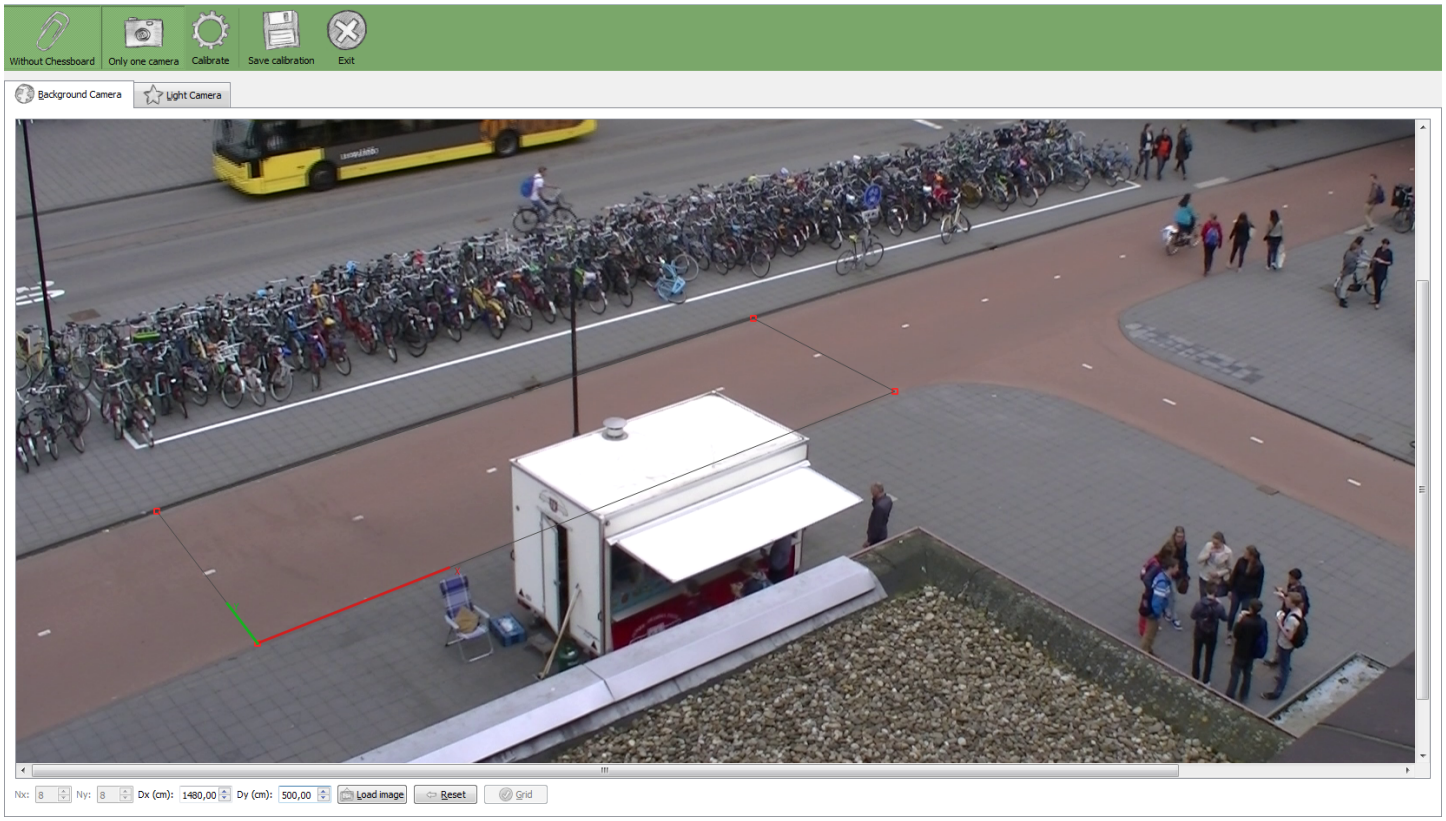


(b) *Location: LANG*

Figure 18: *An overview of the camera view frustums and terrain types for each locations, as seen from above. The frustums are marked in black, grass is marked in green, bicycle lanes in pink, sidewalks in light gray, main roads in dark gray, and dirt tracks in brown. The view frustum is cut off at the dashed line.*



(a) *Ground segment for environment BBG (*$15.90 \times 6.63m$*).*



(b) *Ground segment for environment LANG (*$14.80 \times 5.00m$*).*

Figure 19: *An overview of the identifiable segments on the ground plane as used for camera calibration. The segments are identified by marking 4 points forming a rectangle on the ground plane. The 4 points for each segment and the x-axis are marked in red. The y axis is marked in green.*

(a) *Snapshot for environment BBG.*

(b) *Snapshot for environment LANG.*

Figure 20: *An overview of snapshots from the video footage for both filming locations.*

5.2 Trajectory data precision

Extracting the bicycle trajectories from the video footage was done using the method detailed in Subsection 5.1. Since this comprises the task of manually tracking the bicycle's rear wheel over several video frames, it is prone to human error. As a result, the extracted data might contain inaccurate samples. Therefore we will now elaborate on the precision of the data extracted.

The inaccuracies in our data can have several causes. First, the camera calibration affects the precision of the back projection process. If the camera is calibrated incorrectly, back projection of 2D pixels from a video frame results in 3D points being shifted or warped. The camera calibration will be inaccurate e.g. when the road segment, as put into the QtCalib tool [ZL12] is not rectangular. Additionally, if the dimensions for the road segment are put in incorrectly, the scale of the 3D points will be warped. To counteract the effects of incorrect camera calibration, we have measured an as large as possible road segment within each camera's view frustum and made sure to use rectangular road segments. Video footage from a third filming location was discarded since there was no suitable rectangular road segment visible within the camera's view frustum to perform the camera calibration.

The second cause for inaccuracies lies in the precision of the user input during process of tracking a bicycle's rear wheel position on the ground. Since the video footage has a limited resolution, the position of the rear wheel can only be identified in pixel units. However, the position identified can be off by a few pixels because the user has to manually determine the center of the contact point of the wheel and mark the contact point on the ground, and might accidentally mark a pixel that is not exactly in the center of the contact point. As a result, each sampled point can have a drift of several pixels. This drift also translates to a drift of the 3D world coordinates, and depends on the position in the video frame. The video contains more pixels for objects near the camera, and fewer pixels for objects far from the camera. Therefore, Table 5 shows an overview of the dimensions per pixel with respect to the 2D position in the video frame for each video location.

For example, we can observe in Table 5b that the top right corner of the video footage from the LANG environment has a dimension per pixel of 1.53 m, which is more than an order of magnitude greater than the dimension per pixel at the other positions in the video. These differences are caused by the difference in projection and tilt angle of the camera, which is tilted almost horizontally for the LANG environment. However, the view in the top right corner is obstructed by a building, and thus there is no data from this section of the video. To minimize the effects of inaccuracies in the sampled rear wheel positions, we take a large amount of samples, at least one sample in every three frames. Additionally, we stop taking samples once a bicycle moves far away from the camera, e.g. towards the top right position in the LANG environment. Since our method is focused on taking corners, each camera's view frustum is also oriented towards

Position in video	mpp X	mpp Y
Top left	0.03 m	0.15 m
Top right	0.03 m	0.13 m
Bottom left	0.01 m	0.03 m
Bottom right	0.01 m	0.03 m
Road intersection	0.03 m	0.10 m

(a) *Location: BBG*

Position in video	mpp X	mpp Y
Top left	0.05 m	0.42 m
Top right	0.19 m	1.53 m
Bottom left	0.01 m	0.02 m
Bottom right	0.01 m	0.02 m
Road intersection	0.02 m	0.06 m

(b) *Location: LANG*

Table 5: *An overview of the dimensions per pixel (in meters) for each video footage location.*

the corner or intersection at each filming location. As Table 5 shows, the precision of the video footage at the position of these corners, when assuming a maximum drift of 5 pixels, is well within 0.5 m and far more precise than the precision of 2.5 m obtained using GPS sensors, as explained in Subsection 2.6. Given the scale for which our method is intended, as explained in Section 3, these data have enough accuracy to validate the bicycle routes simulated by our method and perform statistical analysis.

5.3 Analysis

To analyze the results of our method we want to compare it to the data extracted from the video footage, as described in Subsection 5.1. Therefore, we take the trajectories from the extracted data, and let our method simulate the same trajectory. This is done by taking the first and last sample of each trajectory and set them as start position q_{start} and goal position q_{goal}. Furthermore, the average speed is computed for the data trajectory, and set as the preferred speed for our method's bicycle. Additionally, the initial speed and orientation of the method's bicycle are set to the speed and orientation in which the bicycle in the video footage enters the video frame, i.e. the speed computed and orientation from the first two samples in the trajectory data. Since there are 60 trajectories in the data set, this results in an additional 60 trajectories, computed by our method, which we can now compare to the corresponding trajectories in the video footage.

We now have 60 pairs of trajectories, in which each trajectory is approximated as a set of points on the trajectory curve. Each point is sampled using a timestep, as described in Subsection 4.1. For simplicity, we have used a Δt sampling interval of

$\Delta t = \frac{1}{30}$ for implementing an approximation of our method, resulting in 30 sampled points per second. Since the trajectories from the video footage have fewer points, as explained in Subsection 5.2, we project the point sampled during each timestep onto the trajectory from the video footage. This results in a pair of sets of points, one from our method, and one sampled from the trajectory from video footage. Comparing these pairs reduces to the general problem of *Shape matching*, for which several approaches exist [VH01] [Vel01]. Shapes and patterns can be compared using many dissimilarity measures [VH01] [DD09]. Veltkamp presents a set of dissimilarity measures suitable for comparing curves, each focused on different characteristics of curves. Van Kreveld [Kre15] recommends a selection of these dissimilarity measures suitable to our problem.

Ideally, the two trajectories are located close to each other. Therefore, our first set of dissimilarity measures focuses on geometric distances between the two trajectories. We start using Fréchet's distance measure [AG95], which represents the maximum L_2 distance between matching points on both curves. Usually the Hausdorff distance is used for finding the maximum distance, however it does not consider the temporal aspect of the curves, i.e. each point on a curve has a time stamp associated and thus corresponds to a point on the other curve. Next, we use Minkowski's L_2-distance [XW08] to find the sum of distances between the two sets of points. This provides us with an ϵ approximation of the area between the two curves. However, since not all curves in our data set are of equal length, the Minkowski distance increases along with the length of the curve, and thus favors curves of shorter length. Therefore we also use the normalized Minkowski distance [MGL08], which normalizes the Minkowski distance over the number of points. Next, we want to compare differences other than geometric distance. For example, if two cyclists follow a straight line, one might exactly follow the line whereas the other might meander on the line. The geometric distance between both curves will be small, however the second cyclist makes more turns and traverses a longer trajectory. Therefore we also use a *Turning fuction distance* [Vel01] which measures the difference in total rotation between both curves. Finally, we also compare the length of both curves. Ideally, we would want to compare the speed of the bicycles and the traversal time of the trajectories. However, the method's preferred speed is derived from the average speed of the data, whereas the actual cyclist might have a different preferred speed. No model for a cyclist's preferred speed was found.

Now that the dissimilarity measures have been computed, we can use the resulting values. According to Veltkamp and Hagedoorn [VH01], this is considered a *decision problem* in which we must decide whether the dissimilarities are smaller than a given threshold. Unfortunately there are no known threshold values for comparing curves of cyclists. Therefore, we will compare our method against the alternative methods Pure Pursuit and RRT (as described in Subsection 4.2). For each alternative method we simulate the trajectories from the data set, and compute the same dissimilarity measures as mentioned before. This results in three sets of dissimilarity measures, one for our method (referred to as *Dubins* for identification purposes), one for the alternative method Pure Pursuit (referred to as Pure Pursuit), and a set for the alternative method RRT

(referred to as *RRT*). Next we can perform a statistical analysis comparing these three sets.

5.3.1 Testing reliability

Since we are trying to decide on a single problem, whether the curves are (dis)similar or not, ideally we should incorporate all the dissimilarity measures used for each pair of curves into a single abstract construct, as is commonly done in statistics. This can be done by normalizing all the dissimilarities and taking a sum. However, this can only be done if all the dissimilarities have sufficient internal consistency, i.e. they correlate with each other, and actually measure the same construct. This is tested by computing Cronbach's α [Cro51]. The internal consistency of the dissimilarity measures for all methods is low (Cronbach's $\alpha = -.005$ for *Dubins*; $\alpha = .095$ for *Pursuit*; $\alpha = .052$ for *RRT*; $N = 5$). A commonly accepted threshold for Cronbach's alpha is $\alpha > .7$ [NBB67] for the data to be considered consistent and reliable. Since our alpha values are below this threshold we must consider our data inconsistent and, therefore, cannot combine them into a single abstract construct. The α values can be improved for all sets by removing some distance measures from the set (Cronbach's $\alpha = .941$ for *Dubins*; $\alpha = .925$ for *Pursuit*; $\alpha = .912$ for *RRT*; $N = 2$), however we have to remove all but the Fréchet distance and the normalized Minkowski distance. We do not consider this to be a feasible option since it leaves us with a severely diminished set of data. Therefore, the next step is to test all the dissimilarity measures individually, which we will explain in the next section.

5.3.2 Differences between methods

We will compare the three different methods using a factorial ANOVA test, with a repeated-measures design, as described in *Discovering statistics using SPSS* [Fie09, p. 482-502]. The three methods *Dubins*, *Pursuit*, and *RRT* are considered as "conditions". Each "condition" is applied to the each of the trajectories, thus affecting the resulting dissimilarity measures. Prior to performing the ANOVA test, we will check the sphericity of each data set. Sphericity entails whether the variances between each group are roughly equal, and thus whether the samples we took from the video footage, as described in Subsection 5.1, are likely to represent the behavior of all cyclists. The sphericity is tested using Mauchly's Test of Sphericity [Fie09, p. 460]. Mauchly's test indicated that the assumption of sphericity had been violated for the main effects of the type of method on all dissimilarity measures (Fréchet $\chi(2) = 12.24$, $p < .05$; Minkowski $\chi(2) = 9.90$, $p < .05$; normalized Minkowski $\chi(2) = 12.16$, $p < .05$; DegreesTurned $\chi(2) = 72.78$, $p < .001$; LengthDifference $\chi(2) = 6.19$, $p < .05$). Ideally, all p values for Mauchly's test should be higher than .05 for the data to have sufficient sphericity. In our case this does not apply and, therefore, it is required to adjust the degrees of freedom in order to avoid Type I errors. The degrees of freedom were corrected using Greenhouse-Geisser estimates of sphericity [Fie09, p. 461] ($\varepsilon = .83$ for Fréchet; $\varepsilon = .85$ for Minkowski; $\varepsilon = .83$ for normalized Minkowski; $\varepsilon = .57$ for DegreesTurned; and $\varepsilon = .90$ for LengthDifference).

There was a significant main effect for the type of method, $p < .001$. Additionally, a significant interaction effect between the type of method and the environment, $p < .001$, and an interaction effect between the type of method and the type of turn, $p < .001$, as well as a combined interaction effect between both the type of method and the type of turn, $p < .001$. This means that

(a) the curves are different for all methods;

(b) each method performs different in each environment;

(c) each method performs different per type of turn;

(d) each method takes different types of turns differently, based on the environment.

We will now discuss the differences between the methods for each individual dissimilarity measure. The confidence intervals have been adjusted using a Bonferroni correction [Fie09, p. 373] to account for the fact that multiple tests have been performed simultaneously. This is required to avoid Type I errors. Table 7 shows the averages of each dissimilarity measure for each method. Table 6 shows an overview of the significance of these differences. Since all differences but the Fréchet distance are significant, it means that the values in Table 7 have an intuitive meaning. A method performs better than others if its score for a dissimilarity measure is closer to 0. For example, the Minkowski distance for the *Pure Pursuit* method is 23.623, and 26.318 for our method (*Dubins*), which means that the total deviation from the bicycle trajectories captured from video is larger for our method than for *Pure Pursuit*. Unfortunately, this is the case for most of the dissimilarity measures, suggesting that our method is not performing well. In all cases, *RRT* scores the best, whereas our method performs worst in all but two measures, i.e. DegreesTurned and LengthDifference. These differences can be observed in Figure 21 also. From the turning function distance DegreesTurned, we can observe that all methods use less steering on average than real cyclists, resulting in a trajectory with less total rotation. Likewise, all methods produce shorter trajectories on average than real cyclists. Combining these observations with the positive values for the Fréchet and both Minkowski distances, we can conclude that all methods cut corners in comparison to real cyclists. This indeed should result in shorter trajectory lengths, less total rotation, and deviations from the captured data.

When we consider the differences for each environment, as shown in Figure 22, they are not completely consistent with the total averages. In general, the scores for the LANG environment appear to be lower than for the BBG environment, except for Minkowski-Normalized and LengthDifference. The difference between these scores is most likely caused by the geometry of the road in each environment. The LANG environment has T-style junction with a $\approx 90°$ turn, and the BBG environment a Y-style branch with a turn of $\approx 35°$, as shown in Figure 18. Ideally, more data from environments with a similar style of turn should be captured. This would enable us to check whether the

Dissimilarity measure	**Result**	**Significance**
Fréchet	*Dubins* and *Pursuit* do not differ significantly	$p > .05$
	Dubins is significantly different from *RRT*	$p < .001$
Minkowski	*Dubins* is significantly different from *Pursuit* and *RRT*	$p < .05$
MinkowskiNormalized	*Dubins* is significantly different from *Pursuit* and *RRT*	$p < .05$
DegreesTurned	*Dubins* is significantly different from *Pursuit* and *RRT*	$p < .05$
LengthDifference	*Dubins* is significantly different from *Pursuit* and *RRT*	$p < .001$

Table 6: *An overview of the statistical results for each dissimilarity measure.*

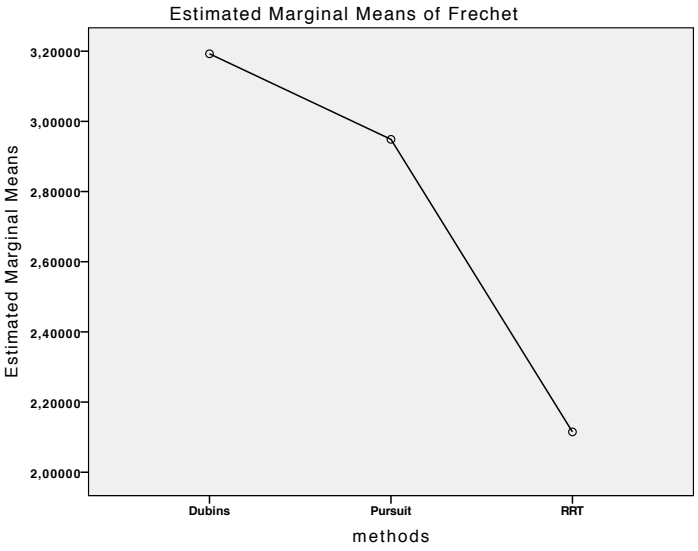
methods perform consistently in different environments with a similar layout. Similar effects are revealed when analyzing the differences for each type of turn, as shown in Figure 23. For going straight, i.e. a trajectory without a turn, each method scores the best for most dissimilarity measures. Right turns have a slightly worse score in general, and left turns score the poorest. This is probably because no shortcuts can be taken during right turns, while during left turns the other side of the road can be used as well. We did not perform a statistical analysis of the running times of our method. An overview of the performance of our method, as well as the alternative methods, is shown in Table 8.

Dissimilarity measure	**Method**	**Mean**	**Unit**
Fréchet	*Dubins*	3.193	m
	Pursuit	2.949	m
	RRT	2.115	m
Minkowski	*Dubins*	26.318	ϵm^2
	Pursuit	23.623	ϵm^2
	RRT	16.981	ϵm^2
MinkowskiNormalized	*Dubins*	1.986	$\epsilon\text{m}^2/n$
	Pursuit	1.800	$\epsilon\text{m}^2/n$
	RRT	1.335	$\epsilon\text{m}^2/n$
DegreesTurned	*Dubins*	-98.868	$^\circ$
	Pursuit	-105.596	$^\circ$
	RRT	-43.463	$^\circ$
LengthDifference	*Dubins*	$-.672$	m
	Pursuit	$-.964$	m
	RRT	$-.144$	m

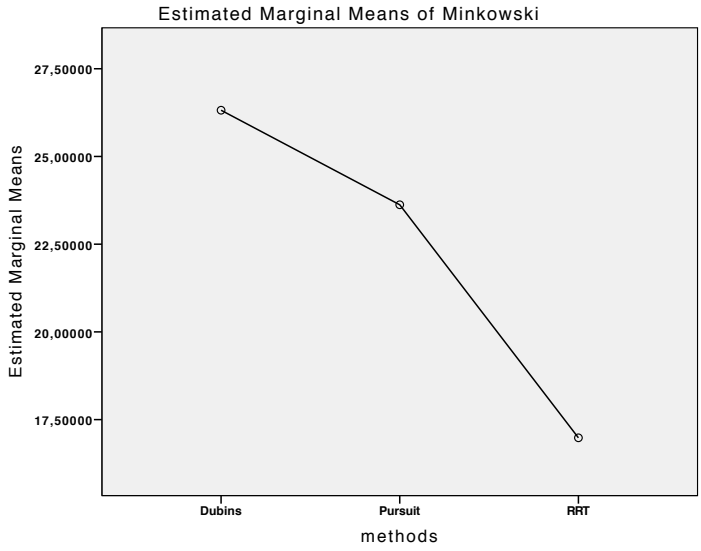
Table 7: *An overview of the estimated means over all trajectories for each dissimilarity measure. 'ϵ' denotes an approximate and 'n' is the number of points in a trajectory, as used for normalization. For example, the value of* -98.868*, for 'DegreesTurned' for the Dubins method, indicates that the trajectories from the video footage have* 98.868° *of rotation more on average compared to our method.*

Scenario	**Simulated time**	**Dubins**	**Pursuit**	**RRT**
1 cyclist	≈ 30 s	0.73	0.31	0.74
2 cyclists	≈ 5 s	0.58	0.57	0.95
4 cyclists	≈ 12.5 s	1.14	1.12	5.29

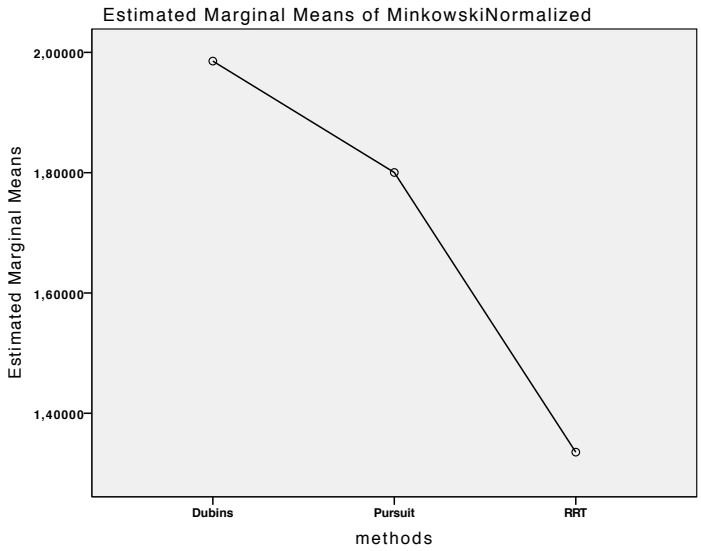
Table 8: *An overview of the average running time (in ms) per simulated time step* ($\Delta t = \frac{1}{30}$)*. The simulations were run on an Intel Core 2 Duo 2.13GHz, running Windows 7 64 bit.*



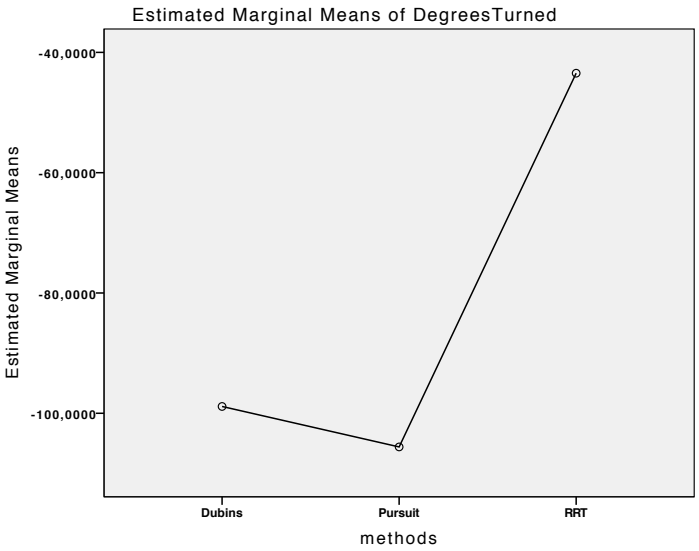
(a)



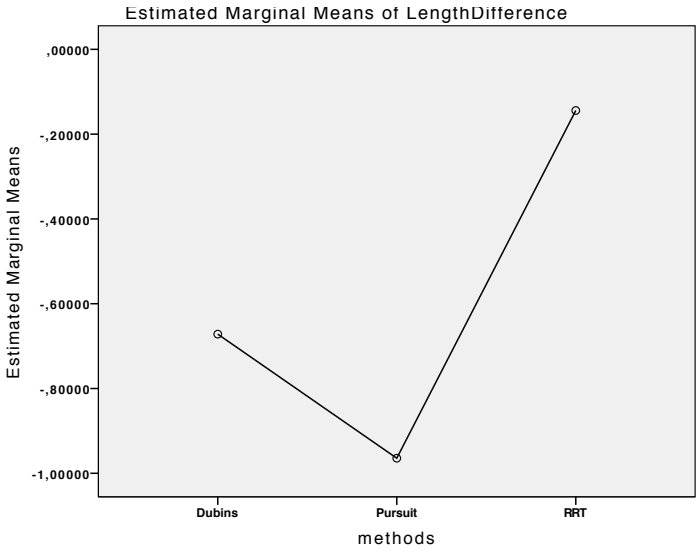
(b)



(c)

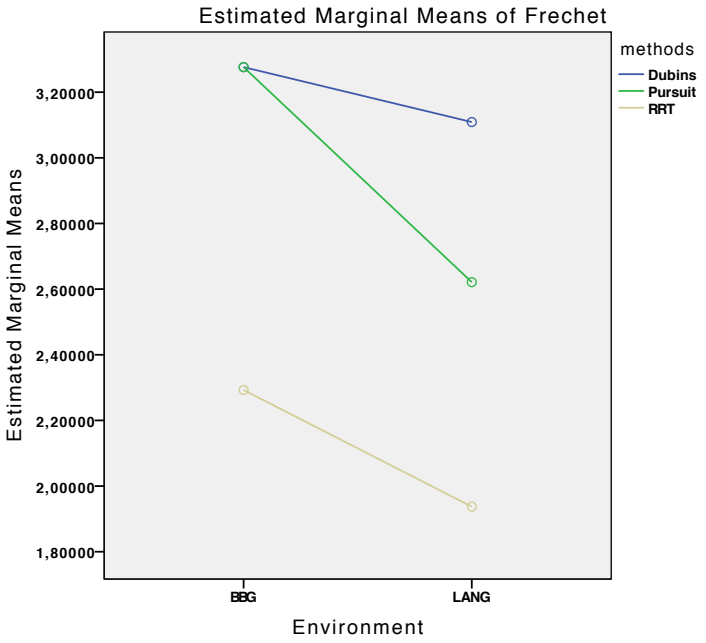


(d)

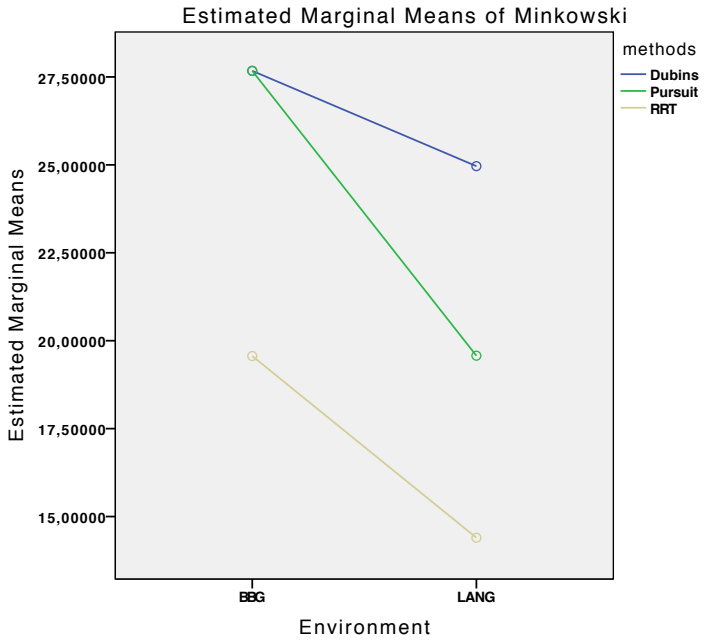


(e)

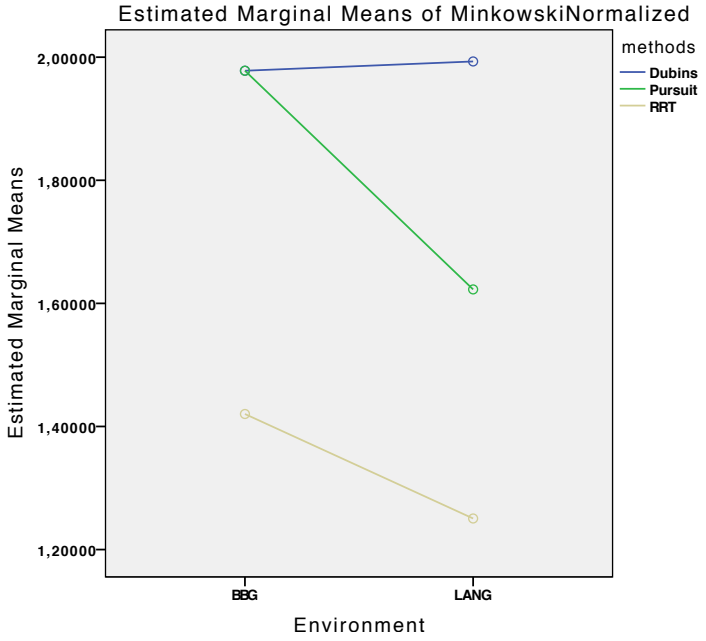
Figure 21: *Plots of the average dissimilarity per type of measure.*



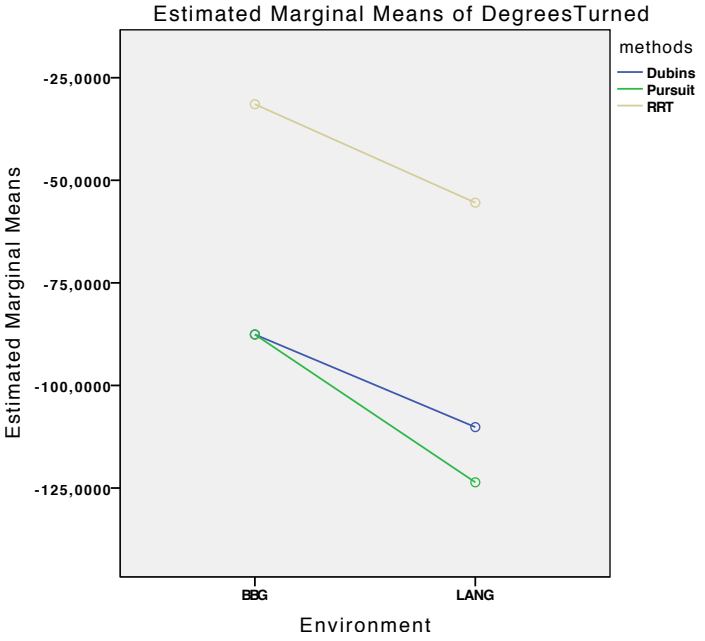
(a)



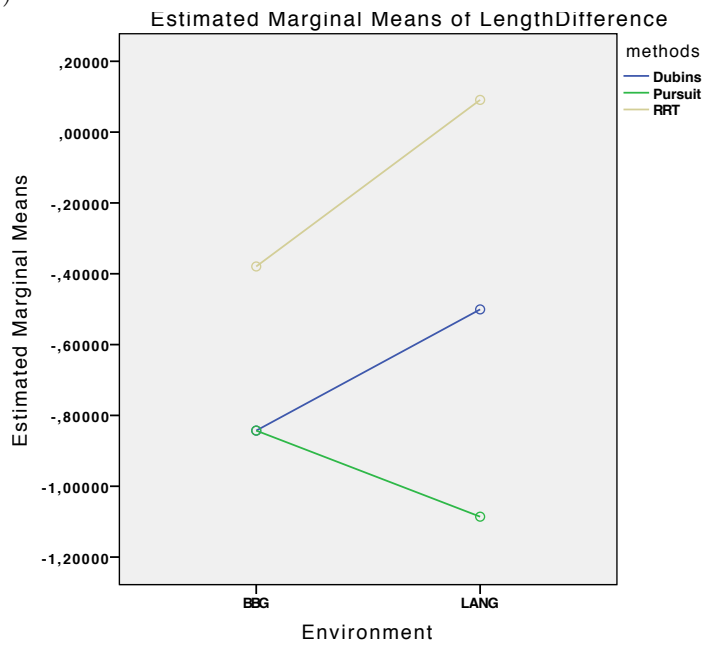
(b)



(c)



(d)



(e)

Figure 22: *Plots of the average dissimilarity for all measures per environment.*

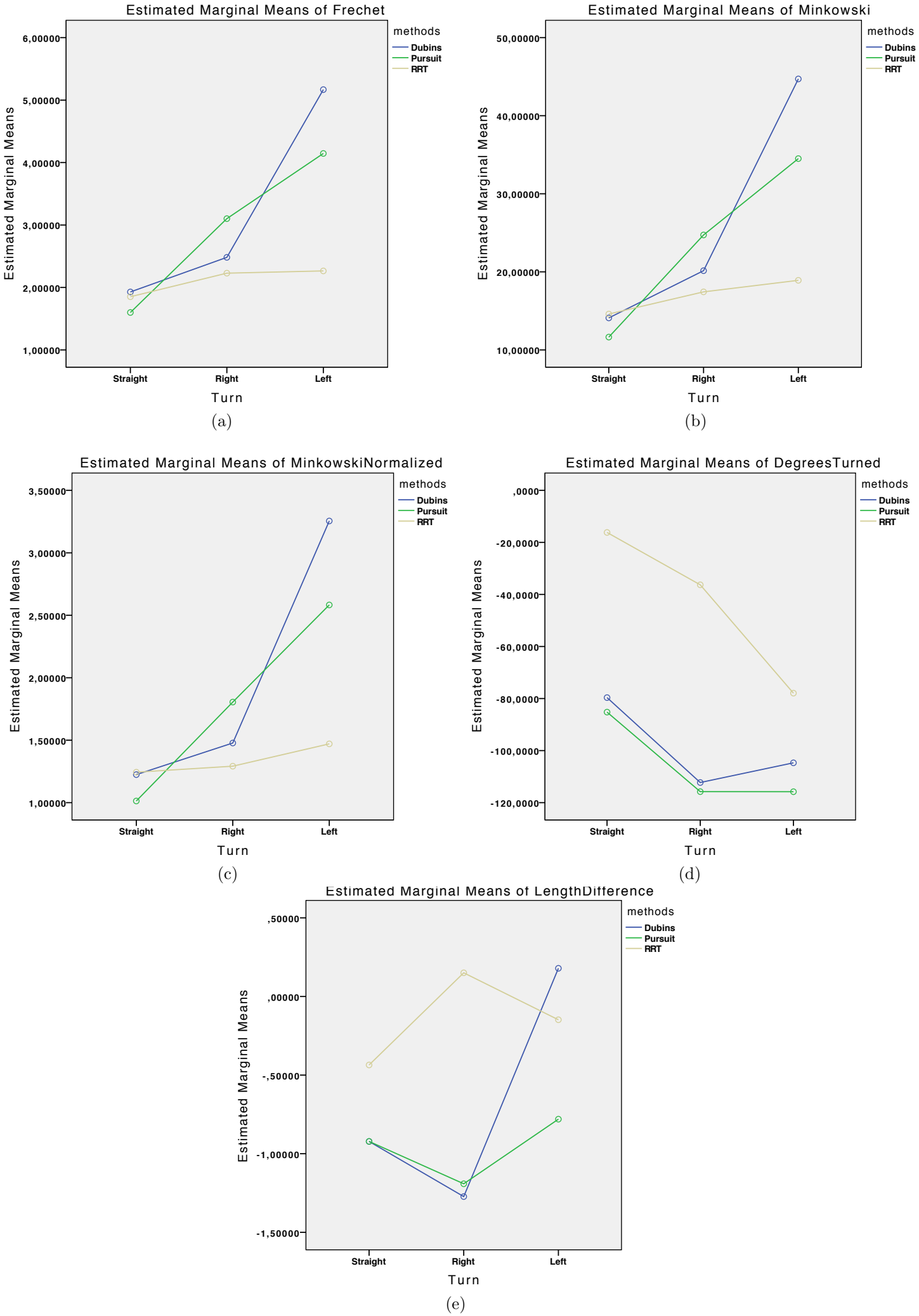


Figure 23: *Plots of the average dissimilarity for all measures per turn.*

6 Conclusion

We have presented a path following method for bicycles which operates on a given global path, described as a discrete set of points. The method uses a local planning window, as described in Subsection 3.3, which moves over the global path as the bicycle moves along. The local planning window simulates the effect of Nystagmus, i.e. it chooses temporary focus points, and plans a local path towards such a point. Dubins paths [Dub57] are used to determine the feasible local path, which is then traversed until a new focus point appears. The turn-radius for the Dubins paths is derived from the bicycle's speed [CP10], which enables the bicycle to take corners whilst minimizing braking. The path following method supports collision avoidance for dynamic obstacles, such as other bicycles, by using the ORCA method, as described in Subsection 3.5. To validate our method, we have created a data set containing bicycle trajectories extracted from video footage shot at two locations. The trajectories in the data set are re-simulated using our method and two alternative methods, as described in Subsection 5.3. The resulting new trajectories are compared to the data set using several dissimilarity measures. A statistical analysis of the dissimilarity measures reveals that our method is performing inadequately in comparison to the alternative methods *RRT* and *Pure Pursuit*. *RRT* scores the best in most of the situations. Given that *RRT* is an inefficient method, this result appears as quite surprising. A probable cause for this is that the trajectories computed by our method are too efficient in comparison to real cyclists. Since *RRT* produces very jagged paths by nature, its results are more close to the data set. In Subsection 2.1 and Section 3 we have argued that the models of bicycle dynamics should not be used for our method, since they are inappropriately complex [Mei+07]. We think that by doing so, we have eliminated the natural meandering behavior as used for keeping the bicycle in balance. Therefore, we think future work should incorporate this effect, whether by using a dynamics model or using a noise function. Despite the fact that *RRT* scored the best in our test, we do not think it is a better method since it hardly runs in real-time, whereas the Dubins paths in our method can be computed in constant time.

6.1 Future work

In our research questions, as described in Subsection 1.1, we described how a possible model should be able to simulate realistic everyday traffic scenarios. This includes collision-avoidance behavior. Since data for collision-avoidance for cyclists is hardly obtained, we did not perform statistical analysis of this part of our method. Therefore, we believe that our model still has potential in this area. Veltkamp [Vel01] states that given the *Shape Matching* problem, as described in Subsection 5.3, we can find the transformation that minimizes the dissimilarities between our method and the data set. In future work, we can try to achieve this by employing optimization techniques such as evolutionary algorithms or simulated annealing to find optimal values for the look-ahead distance σ, which controls the size of the planning window, and the re-planning distance d, which controls the Nystagmus effect. Finally, since our method intends to achieve realistic appearing results, we believe that the realism should not only be determined

using analytic comparison. Instead, the method should also be subjected to a user study in order to visually verify its results. In a user study the method can also be directly compared against the video footage data by scoring all the trajectories. In the validation of our work we did not multiple cyclists. In future work, the behavior of cyclists considering collision avoidance should be investigated. Additionally the effects of traffic regulations and social interactions, such as driving side by side, could be addressed.

List of Symbols

Symbol	Unit	Value	Description
A	-	-	The global path known as the Indicative Route
α_i	-	-	Attraction point location (x,y)
c	m	0	Trail of the bicycle's front fork
d	m	0.5	Re-planning distance
Δt	s	$\frac{1}{30}$	Time step used to discretize the non-holonomic model
L	m	1	Wheelbase of the bicycle
λ	°	90°	The tilt angle of the steering shaft (90° is vertical)
ϕ	°	[−20,20]	The bicycle's steering angle
q	-	-	Configuration of the bicycle, consists of x, y, θ, and ϕ
ρ	m	[2.75,16.65]	Turning radius of the bicycle
s	m/s	[0,10]	The bicycle's speed
σ	m	15	Maximum look-ahead distance
θ	°	[0,360]	The bicycle's orientation angle
x,y	m	-	Coordinates of the bicycle

Acknowledgements

First of all, I would like to thank both of my supervisors Roland Geraerts and Frank van der Stappen for their great support, comments, countless hours of reading, and many meetings. Additionally, I offer my gratitude to Peter de Waal for his aid in setting up the statistical analysis, as well as to Wouter van Toll for his help in implementing the method in the ECM framework, as well as to Jurian de Groot for his collaboration and many discussions. I also offer my thanks to Gert Simonse and Daniel de Kruif for providing the video recording equipment used to capture bicycle footage.

Finally, I would like to thank my fiancé and my parents for supporting me throughout the project.

References

[ABM04] F. Abdessemed, K. Benmahammed, and E. Monacelli. "A fuzzy-based reactive controller for a non-holonomic mobile robot". In: *Robotics and autonomous Systems* 47.1 (2004), pp. 31–46 (cit. on p. 9).

[Aga+02] P. K. Agarwal, T. Biedl, S. Lazard, S. Robbins, S. Suri, and S. Whitesides. "Curvature-constrained shortest paths in a convex polygon". In: *SIAM Journal on Computing* 31.6 (2002), pp. 1814–1851 (cit. on p. 8).

[AM+13] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. *Optimal reciprocal collision avoidance for multiple non-holonomic robots*. Springer, 2013 (cit. on pp. 10, 12, 29).

[AG95] H. Alt and M. Godau. "Computing the Fréchet distance between two polygonal curves". In: *International Journal of Computational Geometry & Applications* 5.01n02 (1995), pp. 75–91 (cit. on p. 44).

[AGO05] R. Alterovitz, K. Goldberg, and A. Okamura. "Planning for steerable bevel-tip needle insertion through 2D soft tissue with obstacles". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 1640–1645 (cit. on p. 8).

[Ast96] A. Astolfi. "Discontinuous control of nonholonomic systems". In: *Systems & control letters* 27.1 (1996), pp. 37–45 (cit. on p. 6).

[AKL05] K. J. Astrom, R. E. Klein, and A. Lennartsson. "Bicycle dynamics and control: adapted bicycles for education and research". In: *Control Systems, IEEE* 25.4 (2005), pp. 26–47 (cit. on pp. 3–6, 11, 13, 16, 27, 36).

[AHHB97] L. Aultman-Hall, F. L. Hall, and B. B. Baetz. "Analysis of bicycle commuter routes using geographic information systems: implications for bicycle planning". In: *Transportation Research Record: Journal of the Transportation Research Board* 1578.1 (1997), pp. 102–110 (cit. on p. 10).

[Bal+00] R. Ballantine, J. Batchelor, D. Eccles, and P. Williams. *Richard's 21st Century Bicycle Book*. Pan Books, 2000. ISBN: 0 330 37717 5 (cit. on p. 4).

[Bar93] G. Barnes. "Visual-vestibular interaction in the control of head and eye movement: the role of visual feedback and predictive mechanisms". In: *Progress in neurobiology* 41.4 (1993), pp. 435–472 (cit. on p. 18).

[BK05] S. Bereg and D. Kirkpatrick. "Curvature-bounded traversals of narrow corridors". In: *Proceedings of the twenty-first annual symposium on Computational geometry*. ACM. 2005, pp. 278–287 (cit. on p. 8).

[Ber+11] J. Berg, van den, S. J. Guy, M. Lin, and D. Manocha. "Reciprocal n-body collision avoidance". In: *Robotics research*. Springer, 2011, pp. 3–19 (cit. on pp. 9, 10, 12, 13, 29).

[Bic13] D. Bickford. "Path Following and Stabilization of an Autonomous Bicycle". In: (2013) (cit. on p. 7).

[Bla12] C. Blankers. "Fietsfiles: Een onderzoek naar de snelheid en intensiteit op fietspaden". Online: access September 24, 2014. 2012. URL: http://www.fietsberaad.nl/?repository=Fietsfiles (cit. on pp. 11, 13).

[BK08] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly, 2008 (cit. on p. 37).

[CP10] S. M. Cain and N. C. Perkins. "Comparison of a bicycle steady-state turning model to experimental data". In: *Bicycle and Motorcycle Dynamics 2010*. Symposium on the Dynamics and Control of Single Track Vehicles. Delft, Netherlands, 2010 (cit. on pp. 20, 35, 36, 52).

[Cou92] R. C. Coulter. *Implementation of the pure pursuit path tracking algorithm*. Tech. rep. DTIC Document, 1992 (cit. on p. 35).

[Cro51] L. J. Cronbach. "Coefficient alpha and the internal structure of tests". In: *psychometrika* 16.3 (1951), pp. 297–334 (cit. on p. 45).

[CRO06] CROW. *Ontwerpwijzer fietsverkeer*. 1st ed. Translated title: Design guide for bicycle traffic. Ede, Netherlands: CROW, Apr. 2006. ISBN: 90 6628 469 2. URL: http://www.crow.nl/publicaties/ontwerpwijzer-fietsverkeer (cit. on pp. 10, 13).

[DG08] C. F. Daganzo and N. Geroliminis. "An analytical approximation for the macroscopic fundamental diagram of urban traffic". In: *Transportation Research Part B: Methodological* 42.9 (2008), pp. 771–781 (cit. on p. 10).

[DD09] M. M. Deza and E. Deza. *Encyclopedia of distances*. Springer, 2009 (cit. on p. 44).

[DW14] M. Dozza and J. Werneke. "Introducing naturalistic cycling data: What factors influence bicyclists safety in the real world?" In: *Transportation research part F: traffic psychology and behaviour* 24 (2014), pp. 83–91 (cit. on p. 11).

[Dub57] L. E. Dubins. "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents". In: *American Journal of mathematics* (1957), pp. 497–516 (cit. on pp. 7, 12, 16, 17, 20, 23, 33, 52).

[Faj00] J. Fajans. "Steering in bicycles and motorcycles". In: *American Journal of Physics* 68.7 (2000), pp. 654–659 (cit. on p. 7).

[Fie09] A. Field. *Discovering statistics using SPSS*. Sage publications, 2009 (cit. on pp. 45, 46).

[FW81] R. J. Forrestall and D. G. Wilson. *Recumbent bicycle*. US Patent 4,283,070. Aug. 1981 (cit. on p. 4).

[FSR90] G Franke, W Suhr, and F Rieß. "An advanced model of bicycle dynamics". In: *European Journal of Physics* 11.2 (1990), p. 116 (cit. on p. 7).

[FDF02] E. Frazzoli, M. A. Dahleh, and E. Feron. "Real-time motion planning for agile autonomous vehicles". In: *Journal of Guidance, Control, and Dynamics* 25.1 (2002), pp. 116–129 (cit. on pp. 9, 35).

[GT13] R. Geraerts and W. Toll, van. *Crowd simulation framework of Utrecht University*. [Online; accessed 13-March-2014]. 2013. URL: http://www.staff.science.uu.nl/~gerae101/motion_planning/ecm/ (cit. on p. 33).

[GD08] N. Geroliminis and C. F. Daganzo. "Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings". In: *Transportation Research Part B: Methodological* 42.9 (2008), pp. 759–770 (cit. on p. 10).

[GM95] N. H. Getz and J. E. Marsden. "Control for an autonomous bicycle". In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*. Vol. 2. IEEE. 1995, pp. 1397–1402 (cit. on pp. 7, 12).

[HWM10] E. Heinen, B. van Wee, and K. Maat. "Commuting by bicycle: an overview of the literature". In: *Transport reviews* 30.1 (2010), pp. 59–96 (cit. on p. 10).

[HM95] D. Helbing and P. Molnar. "Social force model for pedestrian dynamics". In: *Physical review E* 51.5 (1995), p. 4282 (cit. on p. 9).

[HK00] C. Holleman and L. E. Kavraki. "A framework for using the workspace medial axis in PRM planners". In: *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 2. IEEE. 2000, pp. 1408–1413 (cit. on p. 35).

[Hon+68] V Honrubia, W. Downey, D. Mitchell, and P. Ward. "Experimental studies on optokinetic nystagmus II. Normal humans". In: *Acta oto-laryngologica* 65.1-6 (1968), pp. 441–448 (cit. on p. 18).

[HST00] T. Hyodo, N. Suzuki, and K. Takahashi. "Modeling of bicycle route and destination choice behavior for bicycle road network plan". In: *Transportation Research Record: Journal of the Transportation Research Board* 1705.1 (2000), pp. 70–76 (cit. on p. 10).

[JCG13] N. Jaklin, A. Cook, and R. Geraerts. "Real-time path planning in heterogeneous environments". In: *Computer Animation and Virtual Worlds* 24.3-4 (2013), pp. 285–295 (cit. on pp. 8, 13, 18, 20, 33).

[KGO09] I. Karamouzas, R. Geraerts, and M. Overmars. "Indicative routes for path planning and crowd simulation". In: *Proceedings of the 4th International Conference on Foundations of Digital Games*. ACM. 2009, pp. 113–120 (cit. on pp. 12, 17, 36).

[Kav+96] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *Robotics and Automation, IEEE Transactions on* 12.4 (1996), pp. 566–580 (cit. on p. 35).

[Kie+14] H Kiewiet, V. Bulsink, D van de Belt, and H. Koopman. "A Novel Experimental Setup to Apply Controlled Disturbances to Bicycle Dynamics in a Safe Environment". In: *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers. 2014, V003T01A034–V003T01A034 (cit. on p. 11).

[KSM09] J. Kooijman, A. Schwab, and J. K. Moore. "Some observations on human control of a bicycle". In: *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers. 2009, pp. 2021–2028 (cit. on p. 7).

[Kre15] M. J. van Kreveld. Personal communication. July 2015 (cit. on p. 44).

[KL00] J. J. Kuffner and S. M. LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 2. IEEE. 2000, pp. 995–1001 (cit. on pp. 8, 35).

[Kuw+08] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How. "Motion planning for urban driving using RRT". In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE. 2008, pp. 1681–1686 (cit. on pp. 9, 35).

[LSL98] J.-P. Laumond, S Sekhavat, and F Lamiraux. *Guidelines in nonholonomic motion planning for mobile robots*. Springer, 1998 (cit. on pp. 16, 17, 21).

[LaV06] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006 (cit. on pp. 8, 16, 17, 33, 35).

[LK01] S. M. LaValle and J. J. Kuffner. "Randomized kinodynamic planning". In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400 (cit. on pp. 8, 35).

[LRW98] S. Lazard, J. Reif, and H. Wang. "The complexity of the two dimensional curvatureconstrained shortest-path problem". In: *Proceedings of the Third International Workshop on the Algorithmic Foundations of Robotics,(Houston, Texas, USA)*. 1998, pp. 49–57 (cit. on pp. 8, 13).

[LHL06] S. R. Lindemann, I. I. Hussein, and S. M. LaValle. "Real time feedback control for nonholonomic mobile robots with obstacles". In: *Decision and Control, 2006 45th IEEE Conference on*. IEEE. 2006, pp. 2406–2411 (cit. on p. 9).

[May90] A. D. May. *Traffic flow fundamentals*. Prentice Hall, 1990 (cit. on p. 10).

[MK74] D. T. McRuer and E. S. Krendel. *Mathematical models of human pilot behavior*. Tech. rep. DTIC Document, 1974 (cit. on p. 6).

[Mei+07] J. P. Meijaard, J. M. Papadopoulos, A. Ruina, and A. L. Schwab. "Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* 463.2084 (2007), pp. 1955–1982 (cit. on pp. 3, 6, 12, 16, 52).

[MGL08] J. M. Merigó and A. M. Gil-Lafuente. "Using the OWA operator in the Minkowski distance". In: *International Journal of Computer Science* 3.3 (2008), pp. 149–157 (cit. on p. 44).

[Moo12] J. K. Moore. "Human control of a bicycle". [Online; accessed 22-September-2014]. PhD thesis. University of California Davis, 2012. URL: http://moorepants.github.io/dissertation/ (cit. on pp. 6, 7).

[MOW02] MOW. *Vademecum fietsvoorzieningen*. Translated title: Handbook for cycling facilities. Vlaanderen, Belgie: Departement Mobiliteit en Openbare Werken, 2002. URL: http://www.mobielvlaanderen.be/vademecums/vademecumfiets01.php (cit. on p. 13).

[Nar+09] R. Narain, A. Golas, S. Curtis, and M. C. Lin. "Aggregate dynamics for dense crowd simulation". In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 5. ACM. 2009, p. 122 (cit. on p. 3).

[NS04] H.-W. Nienhuys and A. F. Stappen, van der. "A computational technique for interactive needle insertions in 3D nonlinear material". In: *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 2. IEEE. 2004, pp. 2061–2067 (cit. on p. 7).

[NBB67] J. C. Nunnally, I. H. Bernstein, and J. M. t. Berge. *Psychometric theory*. Vol. 226. McGraw-Hill New York, 1967 (cit. on p. 45).

[Oud11] J. Van den Ouden. "Inventory of bicycle motion for the design of a bicycle simulator". MA thesis. TU Delft, Delft University of Technology, 2011 (cit. on p. 11).

[PA14] L. Palmieri and K. O. Arras. "A novel RRT extend function for efficient and smooth mobile robot motion planning". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 205–211 (cit. on pp. 8, 35).

[PB89] J. Papendrecht and H. Botma. *Onderzoek fietsverkeer: basiskenmerken verkeersstroom op éénrichtingsfietspad*. Translated title: Bicycle traffic: basic characteristics of traffic flow on a unidirectional bicycle lane. Technische Universiteit Delft, Faculteit der Civiele Techniek, Vakgroep Verkeer, 1989 (cit. on pp. 10, 13).

[Per+12] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez. "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 2537–2542 (cit. on pp. 8, 35).

[RM71] R. D. Roland and D. E. Massing. *A digital computer simulation of bicycle dynamics*. Cornell Aeronautical Laboratory, 1971 (cit. on p. 7).

[RG14] S. Rosman and R. J. Geraerts. *Realism of Character Representations in Crowd Simulation: A user study on different representations of simulated moving crowds*. July 2014 (cit. on p. 37).

[Sha08] R. S. Sharp. "On the stability and control of the bicycle". In: *Applied mechanics reviews* 61.6 (2008), p. 060803 (cit. on p. 7).

[SWT09] A. Shkolnik, M. Walter, and R. Tedrake. "Reachability-guided sampling for planning under differential constraints". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 2859–2865 (cit. on pp. 8, 35).

[TLK01] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos. "Nonholonomic stabilization with collision avoidance for mobile robots." In: *IROS*. 2001, pp. 1220–1225 (cit. on p. 9).

[TLK03] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos. "Nonholonomic navigation and control of cooperating mobile manipulators". In: *Robotics and Automation, IEEE Transactions on* 19.1 (2003), pp. 53–64 (cit. on p. 9).

[TCIG11] W. Toll, van, A. F. Cook IV, and R. Geraerts. "Navigation meshes for realistic multi-layered environments". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 3526–3532 (cit. on p. 33).

[Tsa87] R. Tsai. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses". In: *Robotics and Automation, IEEE journaltitle of* 3.4 (1987), pp. 323–344 (cit. on p. 37).

[Van+13] P. Vansteenkiste, G. Cardon, E. DHondt, R. Philippaerts, and M. Lenoir. "The visual control of bicycle steering: The effects of speed and path width". In: *Accident Analysis & Prevention* 51 (2013), pp. 222–227 (cit. on pp. 11, 18).

[VH01] R Veltkamp and M. Hagedoorn. "State of the art in shape matching". In: *Principles of visual information retrieval* (2001), p. 87 (cit. on p. 44).

[Vel01] R. C. Veltkamp. "Shape matching: Similarity measures and algorithms". In: *Shape Modeling and Applications, SMI 2001 International Conference on*. IEEE. 2001, pp. 188–197 (cit. on pp. 44, 52).

[Wal08] A. Walker. *Dubins-Curves: an open implementation of shortest paths for the forward only car*. 2008–. URL: https://github.com/AndrewWalker/Dubins-Curves (cit. on p. 33).

[Wal11] A. B. Walker. "Hard Real-Time Motion Planning for Autonomous Vehicles". PhD thesis. Swinburne University, 2011 (cit. on p. 33).

[Wal+85] R. Wallace, A. Stentz, C. E. Thorpe, H. Maravec, W. Whittaker, and T. Kanade. “First Results in Robot Road-Following.” In: *IJCAI*. Citeseer. 1985, pp. 1089–1095 (cit. on p. 35).

[Web+06] R. J. Webster, J. S. Kim, N. J. Cowan, G. S. Chirikjian, and A. M. Okamura. “Nonholonomic modeling of needle steering”. In: *The International Journal of Robotics Research* 25.5-6 (2006), pp. 509–525 (cit. on p. 8).

[Whi99] F. J. Whipple. “The stability of the motion of a bicycle”. In: *Quarterly Journal of Pure and Applied Mathematics* 30.120 (1899), pp. 312–321 (cit. on pp. 3, 6, 16).

[XW08] R. Xu and D. Wunsch. *Clustering*. Vol. 10. John Wiley & Sons, 2008 (cit. on p. 44).

[Xu+12] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha. “A real-time motion planner with trajectory optimization for autonomous vehicles”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 2061–2067 (cit. on p. 9).

[Yav98] Y Yavin. “Navigation and control of the motion of a riderless bicycle”. In: *Computer methods in applied mechanics and engineering* 160.1 (1998), pp. 193–202 (cit. on p. 6).

[ZL12] A. Zang and D. Lucio. *QtCalib Tsai Calibration Tool*. [Online; accessed 15-July-2015]. Aug. 2012. URL: http://w3.impa.br/~zang/qtcalib/ (cit. on pp. 37, 42).