2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Shared Control of Autonomous Vehicles based on Velocity Space Optimization

Javier Alonso-Mora[1,2], Pascal Gohl[1], Scott Watson[2], Roland Siegwart[1] and Paul Beardsley[2]

***Abstract*— This paper presents a method for shared control of a vehicle. The driver commands a preferred velocity which is transformed into a collision-free local motion that respects the actuator constraints and allows for smooth and safe control. Collision-free local motions are achieved with an extension of velocity obstacles that takes into account dynamic constraints and a grid-based map representation. To limit the freedom of the driver, a global guidance trajectory can be included, which specifies the areas where the vehicle is allowed to drive in each time instance. The low computational complexity of the method makes it well suited for multi-agent settings and high update rates and both a centralized and a distributed algorithm are provided that allow for real-time control of tens of vehicles. Extensive experimental results with real robotic wheelchairs at relatively high speeds in tight scenarios are presented.**

I. Introduction

Human-robot shared control has been applied in the field of tele-robot operation for tasks such as space exploration and surgery [1]. Applied to mobile robots, a method for human interaction with a forklift was presented in [2]; an approach for shared control of a formation of aerial vehicles via a haptic device in [3]; work on semi-autonomous wheelchairs includes [4], [5] and [6]; and work on shared control of self-driving cars includes [7]. There is potential to apply shared control to personal urban transporters like the Segway [8]. The ability to impose a guidance trajectory and associated schedule on a mobile vehicle is of further interest as self-driving vehicle technology begins to combine with on-demand car services [9].

The main contribution is a method for shared control of a semi-autonomous vehicle based on: (1) Real-time input from a driver, (2) guidance trajectory, which specifies the areas where the vehicle is allowed to drive in each time instance, (3) avoidance constraints with respect to other agents and a grid map, and (4) motion continuity constraints. As an extension to [10]-[11], efficient centralized and distributed algorithms based on convex optimization and non-convex search are introduced to achieve real-time local planning where an exact grid-based representation of the map is used in conjunction with the velocity obstacles framework. This allows for real-time control of tens of vehicles. Finally, the approach is tested with robotic wheelchairs moving at speeds of up to 3 m/s in close proximity and results are discussed.

II. Overview

Consider a group of n vehicles of which m are controlled, either in a distributed or centralized way. The position, velocity and acceleration of robot i are denoted by $\mathbf{p}_i \in \mathbb{R}^2$, $\mathbf{v}_i = \dot{\mathbf{p}}_i$ and $\mathbf{a}_i = \ddot{\mathbf{p}}_i$ respectively. A disk $D(\mathbf{p}, r)$ of radius r_i centered at $\mathbf{p}_i$ defines its enveloping shape. The enlarged obstacle map is given by a static grid map $\mathscr{O}_r$ of the positions $\mathbf{p}$ where a disk $D(\mathbf{p}, r)$ is in collision with a static obstacle.

The driver commands the vehicle by specifying a desired velocity. To guide the motion of the vehicle, to limit its movements, to guarantee that the vehicle has a certain performance even in the case where the driver does not specify a velocity and to control the throughput of vehicles through out an area, a guidance trajectory together with active areas (areas where the vehicle is allowed to drive) can be specified (Sec. IV). This two inputs are combined and a local trajectory is computed.

For each vehicle, the set of local trajectories is given by a controller (of sufficient continuity, $\mathscr{C}^2$ in our work) towards a straight-line reference trajectory at velocity $\mathbf{u} \in \mathbb{R}^2$ (see Fig. 1 and Sec. III). This provides a low dimensional parametrization (given by $\mathbf{u}$) of the local motions and allows for an efficient optimization in reference velocity space to achieve collision-free motions that respect the dynamic constraints of the vehicle.

In each time-step of the local planner, the motion is obtained (Sec. VI) by computing an optimal reference velocity $\mathbf{u}^*$ such that its associated trajectory is collision-free, satisfies the motion and guidance constraints and minimizes the distance to a preferred velocity $\bar{\mathbf{u}}$. The preferred velocity (Sec. V) takes into account the driver input, a guidance trajectory (Sec. IV) and the neighboring obstacles.

Throughout this paper x denotes scalars, $\mathbf{x}$ vectors, $x = ||\mathbf{x}||$ its Euclidean norm, X matrices, $\mathscr{X}$ sets and $\mathscr{X} \oplus \mathscr{Y}$ their Minkowski sum. The super index $\cdot^k$ indicates the value at time t^k, subindex $\cdot_i$ indicates agent i and $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ relative vectors. Time relative to the current time step is employed when appropriate and denoted by $\tilde{t} = t - t^k$. In practice estimated states are used, but to simplify the notation no distinction is made between estimated and real states.

III. Vehicle model and control

Given a reference trajectory at constant velocity $\mathbf{p}_{\text{ref}}(t) = \mathbf{p}^k + \mathbf{u}\tilde{t}$ for $\tilde{t} \in [0, \infty)$, with $\mathbf{p}^k$ the position of i at the current time-step k and $\mathbf{u}$ the reference velocity, consider a trajectory tracking controller $f(\mathbf{z}^k, \mathbf{u}, t) = \mathbf{p}(t)$ continuous in the initial state, $\mathbf{z}^k = [\mathbf{p}^k, \dot{\mathbf{p}}^k, \ddot{\mathbf{p}}^k, \dots]$ of the agent and converging to the reference trajectory. The set of feasible motions is given by

$$\mathscr{R}(\mathbf{z}^k) = \{\mathbf{u} \in \mathbb{R}^2, \text{ such that the trajectory } f(\mathbf{z}^k, \mathbf{u}, t) \text{ respects all dynamic constraints}\}. \quad (1)$$

[1]J. Alonso-Mora et al. are with the Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland {jalonso}@ethz.ch
[2]J. Alonso-Mora et al. are with Disney Research

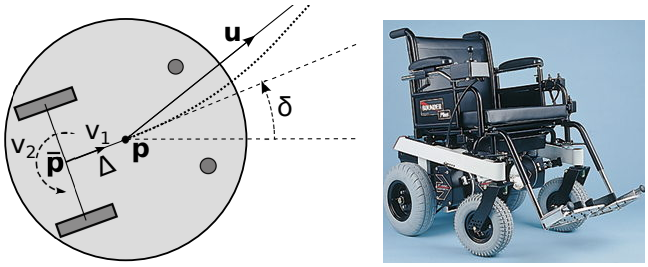

Fig. 1. *Right:* Wheelchair employed in the experiments (wheelchairs.com). *Left:* Schema of the vehicle formed by two rear driving wheels and two front castor wheels. Reference trajectory given by $\mathbf{u}$ and executed trajectory.

This set can be pre-computed by forward simulation and depends on $\dot{\mathbf{p}}^k$ and $\ddot{\mathbf{p}}^k$. A trajectory controller appropriate for the vehicle kinematics is required. We use the following.

1) Trajectory controller: Following [12], if the system is modeled as a second order integrator (continuity in acceleration) and the appropriate state feedback control low towards the reference trajectory is formulated, the trajectory $f(\mathbf{z}^k,\mathbf{u},t)$ is given (for $\tilde{t}>0$) by

$$\mathbf{p}(t)=\mathbf{p}^k+\mathbf{u}\tilde{t}+((\dot{\mathbf{p}}^k-\mathbf{u}+(\omega_1-\omega_0)F)\tilde{t}-F)e^{-\omega_0\tilde{t}}+Fe^{-\omega_1\tilde{t}},$$

with $F=(\ddot{\mathbf{p}}^k+2\omega_0(\dot{\mathbf{p}}^k-\mathbf{v}_\infty))/(\omega_0-\omega_1)^2$ and ω_0, ω_1 design parameters related to the decay of the initial velocity and acceleration.

Saturation limits are added, maximum linear and angular speed ($|v_1|\le v_{1,max}$, $|v_2|\le v_{2,max}$) and maximum linear and angular accelerations ($|\dot{v}_1|\le \dot{v}_{1,max}$, $|\dot{v}_2|\le \dot{v}_{2,max}$).

2) Wheelchair kinematics: The unicycle model is used, with $\bar{\mathbf{p}}$ the point in-between both tractor wheels, δ the orientation of the vehicle and v_1, v_2 the linear and angular speed of the platform (see Fig. 1),

$$\begin{bmatrix}\dot{\bar{\mathbf{p}}}, & \dot{\delta}\end{bmatrix}'=\begin{bmatrix}\cos\delta, & \sin\delta, & 0\end{bmatrix}'v_1+\begin{bmatrix}0, & 0, & 1\end{bmatrix}'v_2,$$

Following [13], a (fully controllable or holonomic) point $\mathbf{p}=\bar{\mathbf{p}}+[\cos\delta,\sin\delta]^T\Delta$ to the front of the vehicle axles is considered, where $\Delta>0$ defines the maneuverability of the vehicle. The following relations hold

$$\dot{\mathbf{p}}=\xi,\;\; v_1=\xi\cdot[\cos\delta,\sin\delta]^T,\;\; v_2=\xi\cdot[-\sin\delta,\cos\delta]^T/\Delta.$$

The local trajectory is applied at this point and a circle of radius r is considered as the enveloping shape of the vehicle.

IV. Guidance Trajectory

For some applications, and to have higher control over the motion of the vehicles, it can be beneficial (although not required) to create a roadmap of guidance trajectories. From a static map, the roadmap can be computed automatically via geometric or sampling-based methods [14] or be designed (this is the case in our work) via waypoints connected by straight lines. This representation does not require to account for the dynamics of the vehicle, nor to perfectly fit the map, since both are considered by the local planner.

For each waypoint $\mathbf{w}_l\in\mathbb{R}^2$ the designer specifies an arrival time T_l (or alternatively the speed over the segment), as well as an active area of radius ρ_l around it, where the vehicle is allowed to freely move. Consecutive waypoints are connected and define the guidance trajectory and the active area around it (by interpolation), an example is shown in Fig. 2. When two or more active areas (from different trajectories) intersect, the vehicle is assigned depending on its current position and specified attributes of the guidance trajectories, such as if more than one vehicle can follow that trajectory or if the vehicle must stay in the current one. We select the trajectory s that minimizes the weighted distance to the current point on the guidance trajectory $\mathbf{q}^s$ relative to the current radius of the active area ρ^s,

$$\arg\min_s(1-\sigma^s)||\mathbf{p}-\mathbf{q}^s||/\rho^s \qquad (2)$$

where $\sigma^s\in(0,1)$ is an attribute indicating the predilection to stay in one trajectory.

This framework provides an intuitive way to guide a team of vehicles through-out the scenario while giving them limited freedom in their movement via the active areas. Large radius ρ implies freedom of movement, while small radius ρ imposes accurate tracking of the guidance trajectory.

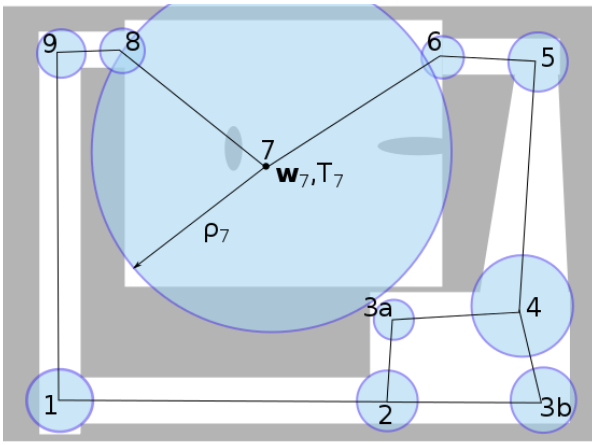

Fig. 2. Map with several rooms and corridors. Empty space in white and occupied space in grey. Nine waypoints $\mathbf{w}_i$ are created with arrival times T_i and different radii ρ_i for their associated active area (blue circle). In-between waypoints, the ideal trajectory and active area is given by interpolation. Waypoints 3a and 3b represent a bifurcation and waypoint 7 a large area where the vehicles are allowed to freely drive.

V. Shared Control

In each time step, for each vehicle i, a preferred velocity $\bar{\mathbf{u}}_i$ is computed, based on the input from the driver, the guidance trajectory and the neighboring obstacles,

$$\bar{\mathbf{u}}_i=\mu_1(\mu_2\mathbf{u}_i^{joy}+\mathbf{u}_i^{traj})+(1-\mu_1)\mathbf{u}_i^{A*}+\mathbf{u}_i^{rep}, \qquad (3)$$

where $\mu_2=1$ if the vehicle is inside the active area and $\mu_2=0$ otherwise, and $\mu_1=1$ if $\mathbf{q}$ is in line of sight of the vehicle and $\mu_1=0$ otherwise (see Fig. 3). In practice a small hysteresis is introduced to μ_1 and μ_2 to avoid oscillations. In normal operation $\mu_1=\mu_2=1$. Each velocity term is computed as follows. The input velocity $\mathbf{u}_i^{joy}$ is proportional to the joystick position (in the relative frame of the vehicle).

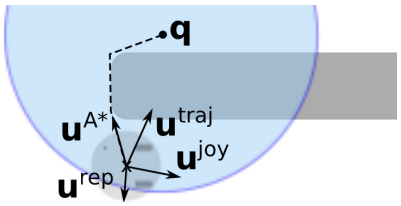

Fig. 3. Shared control: components of the preferred reference velocity for a vehicle within the active area (circle) of the current point $\mathbf{q}$ on the ideal trajectory. An enlarged obstacle is marked in grey.

The guidance velocity $\mathbf{u}_i^{traj}$ (optional term) is given by a proportional controller to the guidance trajectory

$$\mathbf{u}_i^{traj} = K_p(\mathbf{q}_i - \mathbf{p}_i) + \dot{\mathbf{q}}_i, \tag{4}$$

where $\mathbf{q}_i$ is the ideal position on the guidance trajectory at the current time. The repulsive velocity $\mathbf{u}_i^{rep}$ is computed with a potential field approach that pushes the vehicle away from both static and dynamic obstacles[1]. A linear function weighting the distance to other vehicles is used

$$\mathbf{u}_i^{rep} = V_r \frac{\bar{D} - d(\mathbf{p}_i, \mathscr{O})}{\bar{D}} \mathbf{n}_{i,\mathscr{O}} + \sum_{j \,|\, p_{ij} < D} V_r \frac{D - p_{ij}}{D - r_i - r_j} \frac{\mathbf{p}_{ij}}{p_{ij}}, \tag{5}$$

where V_r is the maximum repulsive velocity, D and $\bar{D}$ the maximum distance from which a repulsive force is added, $d(\mathbf{p}_i, \mathscr{O})$ is the distance to the closest static obstacle within the $\mathscr{O}_{r_i}$ map and $\mathbf{n}_{i,\mathscr{O}}$, the normal vector from it. In order to avoid corner issues, not only the closest point is considered, but an average over the closest points. The norm of $\mathbf{u}_i^{rep}$ is then limited to $||\mathbf{u}_i^{rep}|| \leq V_r$.

In cases where the guidance point $\mathbf{q}_i$ is not in direct line of view from the current position $\mathbf{p}_i$ a preferred velocity solely based on the previous terms could be prone to deadlocks. To avoid this, a term $\mathbf{u}_i^{A^*}$ computed frorm a global path from $\mathbf{p}_i$ to $\mathbf{q}_i$ taking into account the map $\mathscr{O}_r$ but ignoring the kinematic constraints and other vehicles is added. This term is given by the first control input of a standard A* search in the grid map [15].

VI. Local motion planning

In each control loop, given the preferred velocity $\bar{\mathbf{u}}_i$ of the ego vehicle and the current position and velocity of neighboring vehicles, a local trajectory given by an optimal reference velocity $\mathbf{u}_i^*$ is computed by solving an optimization problem in reference velocity space, formulated as a combination of a convex optimization with quadratic cost and linear and quadratic constraints and a non-convex optimization.

Two alternative formulations are presented, a *centralized* optimization where the optimal reference velocities of all vehicles $\mathbf{u}_{1:m}^* = [\mathbf{u}_1^*, ..., \mathbf{u}_m^*]$ are jointly computed. And a *distributed* optimization where each vehicle i independently solves an optimization where its optimal reference velocity $\mathbf{u}_i^*$ is computed. The position $\mathbf{p}_j$ and velocity $\mathbf{v}_j$ of neighboring agents is known.

The **optimization cost** is given by two parts, the first one, a regularizing term penalizing changes in velocity and the second one minimizing the deviation to the preferred reference velocity. For the *centralized* case

$$C(\mathbf{u}_{1:m}) := K_o ||\mathbf{u}_{1:m} - \mathbf{v}_{1:m}||^2 + ||\mathbf{u}_{1:m} - \bar{\mathbf{u}}_{1:m}||^2, \tag{6}$$

where K_o is a design constant. For the *distributed* case

$$C(\mathbf{u}_i) := K_o ||\mathbf{u}_i - \mathbf{v}_i||^2 + ||\mathbf{u}_i - \bar{\mathbf{u}}_i||^2, \tag{7}$$

The kinematic constraints of the vehicle are included following the idea of [10] where the vehicle's radius is enlarged by a value $\varepsilon > 0$ and the local trajectories are limited to those with a tracking error below ε with respect to the reference trajectory (parametrized by $\mathbf{u}$).

*Constraint 1 (**Avoidance other agents**):* For every pair of neighboring agents $i \leq m$, $j \leq n$ the constraint is given by the reference velocities such that $\|\mathbf{p}_i - \mathbf{p}_j + (\mathbf{u}_i - \mathbf{u}_j)t\| \geq r_{i+j} = r_i + \varepsilon_i + r_j + \varepsilon_j$, for all $t \in [0, \tau]$.

Following [16] this constraint is rewritten as $\mathbf{u}_i - \mathbf{u}_j \notin VO_{ij}^\tau = \bigcup_{t=0}^\tau ((D(\mathbf{p}_j, r_j + \varepsilon_j) \oplus D(\mathbf{p}_i, r_i + \varepsilon_i))/t)$, representing a truncated cone, see Fig. 4, and computed if the distance between the two agents is below a threshold ($p_{ij} < K_d$). The non-convex constraint $\mathbb{R}^2 \setminus VO_{ij}^\tau$ is approximated by three linear constraints of the form $\mathbf{n}_{ij}^l \cdot \mathbf{u}_{ij} - b_{ij}^l$

$$\begin{bmatrix} \cos(\gamma^+) \\ \sin(\gamma^+) \end{bmatrix} \mathbf{u}_{ij} \leq 0, \; -\frac{\mathbf{p}_{ij}}{p_{ij}} \cdot \mathbf{u}_{ij} \leq \frac{p_{ij} - r_{i+j}}{\tau}, \; \begin{bmatrix} \cos(\gamma^-) \\ \sin(\gamma^-) \end{bmatrix} \mathbf{u}_{ij} \leq 0, \tag{8}$$

where $\gamma^+ = \alpha + \beta$, $\gamma^- = \alpha - \beta$, $\alpha = \text{atan2}(-\mathbf{p}_{ij})$ and $\beta = \text{acos}(r_{i+j}/p_{ij})$. The first and last constraints represent avoidance to the right and to the left, and the middle one a head-on maneuver (collision-free up to $t = \tau$). The linearization of $\mathbb{R}^2 \setminus VO_{ij}^\tau$ is obtained by selecting the linear constraint with maximum constraint satisfaction for the current relative velocity ($\min_l(\mathbf{n}_{ij}^l \mathbf{v}_{ij} - b_{ij}^l)$). Other sensitive choices include linearization with respect to the preferred velocity $\bar{\mathbf{u}}_{ij}$ or fixed avoidance side (right / left). For $j \leq m$ this linear constraint is added to the *centralized* optimization. For $j > m$ (dynamic obstacles) or when distributed, it is rewritten as follows.

In the *distributed* case all agent's are considered as independent decision-making agents solving their independent optimizations. To globally maintain the constraint satisfaction and avoid collisions, an assumption on agent j's reference velocity is required [2]. Variable sharing of avoidance effort might be considered with $\Delta\mathbf{v}_i = \lambda \Delta\mathbf{v}_{ij}$ and the assumption that $\Delta\mathbf{v}_j = -(1-\lambda)\Delta\mathbf{v}_{ij}$. For collaborative agents that equally share the avoidance effort $\lambda = 0.5$. If it is considered that agent j ignores agent i and continues with its current velocity, then $\lambda = 1$ and full avoidance is performed by agent i. For $\lambda \in [0, 1]$ the constraint is given by

$$\mathbf{n}_{ij}^l \cdot \mathbf{u}_{ij} = \frac{\mathbf{n}_{ij}^l}{\lambda} \cdot (\mathbf{u}_i - (1-\lambda)\mathbf{v}_i - \lambda \mathbf{v}_j) \leq b_{ij}^l, \tag{9}$$

[1]Constraints to achieve collision-free motion are included in the local planning framework. Since they are written in velocity space, vehicles can come arbitrarily close to each other. This can be unsafe in real scenarios with uncertainties in sensing and vehicle model. The repulsive velocity is thus included to ideally maintain a minimum distance to other vehicles and walls and moves the ego-vehicle away when it is too close.

[2]The change in velocity is denoted by $\Delta\mathbf{v}_i = \mathbf{u}_i - \mathbf{v}_i$ and the relative change in velocity by $\Delta\mathbf{v}_{ij} = \Delta\mathbf{v}_i - \Delta\mathbf{v}_j$.

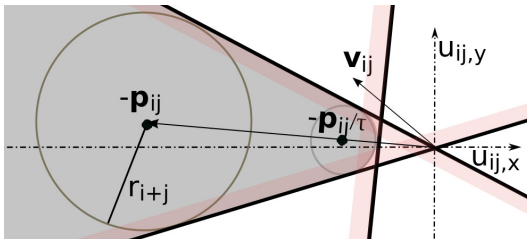

Fig. 4. In relative reference velocity space, constraint (grey) for avoidance between two vehicles, approximated by three linear constraints.

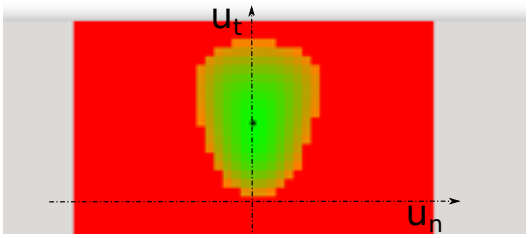

Fig. 5. In green and yellow, reachable reference velocities R_i for a representative current velocity (dot). Plot in reference velocity space aligned with the vehicle axis (tangential/normal). Forward velocities displayed.

*Constraint 2 (**Active area**):* Guarantees that the agent locally remains within the active area. For agent $i \leq m$ the constraint is given by $\|\mathbf{p}_i - \mathbf{q}_i + (\mathbf{u}_i - \dot{\mathbf{q}}_i)\tau\|^2 \leq \rho_i^2$, with $\mathbf{q}_i$ the point on the guidance trajectory for the current time, $\dot{\mathbf{q}}_i$ its speed and ρ_i the radius of its associated active area (by interpolation between the previous and next waypoint).

With the constant velocity assumption for the guidance trajectory, if the vehicle is initially within the active area, this constraint guarantees that it remains inside for time $t \leq \tau$. If the vehicle is initially outside, it guarantees that the vehicle enters within time τ. The constraint can be rewritten as

$$\mathbf{u}_i^T \mathbf{u}_i + 2\left(\frac{\mathbf{p}_i - \mathbf{q}_i}{\tau} - \dot{\mathbf{q}}_i\right)^T \mathbf{u}_i + \left\|\left(\frac{\mathbf{p}_i - \mathbf{q}_i}{\tau} - \dot{\mathbf{q}}_i\right)\right\|^2 \leq \frac{\rho_i^2}{\tau^2}.$$

This is a quadratic constraint (circle in reference velocity space), ignored if no active areas are given.

*Constraint 3 (**Avoidance static obstacles**):* For agent $i \leq m$, the constraint is given by the reference velocities such that the new positions are not in collision with the enlarged obstacle map, $\mathbf{p}_i + \mathbf{u}_i t \notin \mathscr{O}_{r_i+\varepsilon_i}$, for all $t \in [0, \tau]$.

This constraint is kept in its grid-based form (gray area in Fig. 2) to allow the vehicle full navigation capability, being able to move close to the static obstacles and go inside small openings. Given a known map $\mathscr{O}_0$, the enlarged map $\mathscr{O}_{r_i+\varepsilon_i}$ can be precomputed for several values of ε_i.

*Constraint 4 (**Dynamic restrictions**):* Each agent must remain within ε_i of its reference trajectory. For a maximum tracking error ε_i and current state $\mathbf{z}_i = [\mathbf{p}_i, \dot{\mathbf{p}}_i, \ddot{\mathbf{p}}_i]$, the set of reference velocities $\mathbf{u}_i$ that can be achieved with position error below ε_i is given by $R_i := R(\mathbf{z}_i, \varepsilon_i)$

$$R_i := \{\mathbf{u}_i \in \mathscr{R}(\mathbf{z}_i) \mid ||(\mathbf{p}_i + t\mathbf{u}_i) - f(\mathbf{z}_i, \mathbf{u}_i, t)|| \leq \varepsilon_i, \forall t > 0\}. \tag{10}$$

If the trajectory controller of Sec. III is used, the set of reachable reference velocities R_i depends on $\dot{\mathbf{p}}_i$, $\ddot{\mathbf{p}}_i$ and ε_i. A mapping γ from initial state $\dot{\mathbf{p}}$, $\ddot{\mathbf{p}}$ and reference velocity $\mathbf{u}$ to maximum tracking error is precomputed and stored in a look up table

$$\gamma(\mathbf{u}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}) = \max_{t>0} \|(\mathbf{p} + t\mathbf{u}) - f(\mathbf{z}, \mathbf{u}, t)\| \tag{11}$$

An example of this constraint is shown in Fig. 5. A bounding box given by four linear constraints of the form $H_l = \{\mathbf{u}_i \mid A_l \mathbf{u}_i \leq b_l\}$ is computed such that $R(\mathbf{z}_i, \varepsilon_i) \subset \bigcup_{l=1}^4 H_l$. These linear constraints are included in both the *centralized* and *decentralized* optimizations to reduce the search space.

VII. Algorithm and Guarantees

The optimization consists of two types of constraints, convex (linear and quadratic) and non-convex (grid-based). To efficiently find a solution, the optimization is divided in two parts, fist a convex subproblem is solved resulting in $\mathbf{u}_i^c$, followed by a search within the grid-based constraints restricted to the convex area defined by the linear and quadratic constraints. The set of convex constraints (Constraints 1, 2 and bounding box of Constraints 4) is denoted by $\mathscr{C}$. The set of non-convex constraints (Constraints 3 and 4 for agent i, with respect to a grid of the same resolution) is denoted by $\bar{\mathscr{C}}_i$. For both the *centralized* ($N = m$) and the *distributed* ($N = 1$, without loss of generality) optimizations the algorithm proceeds as follows.

Input distributed: $\mathbf{z}_1, \bar{\mathbf{u}}_1$ and $r_1 + \varepsilon_i$; $\mathbf{p}_j, \dot{\mathbf{p}}_j$ and r_j $\forall j$ neighbor of 1. Consider $\varepsilon_j = \varepsilon_1$.
Input centralized: $\mathbf{z}_i, \bar{\mathbf{u}}_i$ and $r_i + \varepsilon_i$ $\forall i \leq m$; $\mathbf{p}_j, \dot{\mathbf{p}}_j$ and r_j $\forall j > m$ neighbor of $i \leq m$.
Result: $\mathbf{u}^*_{1:N} = [\mathbf{u}^*_1, \ldots, \mathbf{u}^*_N]$
Compute constraints (Sec. VI)
$\mathbf{u}^c_{1:N} \leftarrow$ solution $2N$-dimensional convex optimization with quadratic cost (Eq. (6)-(7)) and convex constraints $\mathscr{C}$.
// Wave expansion from $\mathbf{u}^c_{1:N}$ within convex area $\mathscr{C}$.
Initialize sorted list $\mathscr{L}$ (increasing cost) with $\mathbf{u}^c_{1:N}$.
while $\mathscr{L} \neq \emptyset$ **do**
  $\mathbf{u}_{1:N} \leftarrow$ fist point in $\mathscr{L}$; $\mathscr{L} := \mathscr{L} \setminus \mathbf{u}_{1:N}$; feasible := 'true';
  for $i = 1$ **to** N **do**
    if feasibleDynamics($\mathbf{u}_i$)='false' **or** feasibleMap($\mathbf{u}_i$)='false' **then**
      $\mathscr{L} \leftarrow$ expandNeighbors($\mathscr{L}$, $\mathbf{u}_{1:N}$, i, 1);
      feasible := 'false'; **break**;
    end if
  end for
  if feasible = 'true' **then return** $\mathbf{u}^*_{1:N} = \mathbf{u}_{1:N}$ **end if**.
end while; **return** 0;

Function feasibleDynamics($\mathbf{u}_i$), checks in a precomputed grid if the tracking error is below ε_i, given the initial state of the vehicle.

if $\mathbf{u}_i \in R_i$ (See Eq. (10)) **then return** 'true'; **else** 'false';

Function feasibleMap($\mathbf{u}_i$), checks if $\mathbf{u}_i$ leads to a trajectory in collision with static obstacles given by the grid map $\mathscr{O}$. This is efficiently checked in the precomputed dilated map $\mathscr{O}_{r_i+\varepsilon_i}$ (See Constraint 3).

if segment $(\mathbf{p}_i, \mathbf{p}_i + \mathbf{u}_i\tau) \cap \mathscr{O}_{r_i+\varepsilon_i} = \emptyset$ **then return** 'true';

The function expandNeighbors adds the neighboring grid points if they are within the convex region defined by the convex constraints in $\mathscr{C}$, and they were not previously explored. This algorithm is recursive for $N > 1$, although only one level of recursion is used to reduce computational time at the expense of optimality.

Function $\mathscr{L} \leftarrow$ expandNeighbors($\mathscr{L}$, $\mathbf{u}^{in}_{1:N}$, k, rec);
for each 8-connected grid neighbor $\mathbf{u}_k$ of $\mathbf{u}^{in}_k$ **do**
  Consider $\mathbf{u}_{1:N} = [\mathbf{u}^{in}_1, \ldots, \mathbf{u}^{in}_{k-1}, \mathbf{u}_k, \mathbf{u}^{in}_{k+1}, \ldots, \mathbf{u}^{in}_N]$.
  if $\{\mathbf{u}_{1:N}$ not previously added to $\mathscr{L}\}$ **and**
  $\{\mathbf{u}_k$ not checked as unfeasible w.r.t. constr. $\tilde{\mathscr{C}}_i\}$ **then**
    if $\mathbf{u}_{1:N}$ satisfies convex constraints $\mathscr{C}$
      Add $\mathbf{u}_{1:N}$ to $\mathscr{L}$
    else
      if{rec=1} **and** {$\exists$ constr. type 1 unfeasible} **then**
        Select constr. which maximizes $\mathbf{n} \cdot \mathbf{u}_{1:N} - b > 0$.
        $j :=$ "index of second agent in the selected constraint".
        if $j \leq N$ **then** $\mathscr{L} \leftarrow$ expandNeighbors($\mathscr{L}$, $\mathbf{u}_{1:N}$, j,0)
end if.; **end if**; **end if**; **end if**; **end for**

If the optimization is unfeasible, constraints 2 are removed and the optimization is recomputed. If still unfeasible, agents $i \leq N$ decelerate on their last feasible local trajectory following the time remap

$$\gamma(t) = -t_f^2/(2\tau) + (1 + t_f/\tau)t - t^2/(2\tau), \qquad (12)$$

where t_f is the time where the last feasible local motion was obtained. This reparametrization of the trajectories guarantees that the vehicles stop within the time horizon of the local planner, unless the optimization becomes feasible in a later time-step [17].

***Remark** 1 (Computational complexity):* With a limit in the number of linear constraints for collision avoidance the complexity of the algorithm can be kept linear with the number of agents, although the *centralized* is of higher cost. If *distributed*, for each agent the optimization consists of two variables, one quadratic constraint, four linear constraints (bounding box of Constr. 5), a maximum of m linear constraints of type 1 (in practice limited to a constant K_c) and a wave expansion within the 2D grid. If *centralized*, the optimization consists of $2n$ variables, n quadratic constraints, less than $4n + \frac{n(n-1)}{2} + n(m-n)$ linear constraints (limited to $4n + nk_c$) and a joint wave expansion in n 2D grid maps.

***Remark** 2 (Safety guarantees):* If *feasible*, the local trajectories are collision-free up to time τ, with the assumption that other vehicles follow the same algorithm or maintain a constant velocity. If *unfeasible*, no collision-free solution exists that respects all the constraints. If the time horizon is longer than the required time to stop, safety is preserved if all vehicles drive their last feasible trajectory with a time reparametrization (12) to reach stop before a collision arises,

$$\dot{\gamma}(t_f + \tau) = (1 + t_f/\tau) - (t_f + \tau)/\tau = 0. \qquad (13)$$

***Remark** 3 (Unfeasibility):* The optimization can be unfeasible due to several causes:
(a) Not enough time to find the solution within the allocated time. (b) Differences between the model and the real vehicle. (c) Noise in localization and estimation of vehicles' state. (d) Due to the limited local planning horizon together with over simplification of motion capabilities by reducing them to the set of local motions of Sec. III. (e) If *distributed*, given the use of pair-wise partitions of velocity space with either the assumption of equal effort in the avoidance or constant speed, not all world constraints are taken into account for the neighboring agents. Thus a vehicle may have conflicting partitions with respect to different neighbors / static obstacles / kinematics rendering his optimization unfeasible.

***Remark** 4 (Deadlock-free guarantees):* For a single agent, deadlock-free navigation steams thanks to the u_{A^*} term that drives the vehicle towards its goal position or guidance trajectory. In the multi-agent scenario, deadlocks are still feasible, although in degenerated situations. Giving priority to those agents further in their guidance trajectory can help resolve the situation. The input from the driver can also act as a deadlock breaking input.

VIII. Experimental results

The algorithms have been extensively tested in simulation with larger groups of robots. Due to space limitations, the analysis is restricted to the case with real wheelchairs (Fig 1) with driver. Each vehicle has an enclosing radius of 1 m, limits $v_{1,max} = 3$ m/s, $v_{2,max} = 2$ rad/s, $\dot{v}_{1,max} = 2$ m/s^2, $\dot{v}_{2,max} = 20$ rad/s^2, $\dot{v}_{1,min} = -1.1$ m/s^2 and a maximum sideways acceleration of 1.5 m/s^2. Vehicles are tracked by an overhead tracking system and controlled from a central computer at 30Hz. Velocity commands from the driver are transmitted to the central computer at the same rate. Computations are performed in an i5 3GHz PC.

A. Free driving

In this section semi-autonomous driving is discussed, without a guidance trajectory and where each vehicle is driven by a human that inputs a desired velocity $\mathbf{u}^{joy}$. Over 50 hours of experiments showed that the algorithms are safe and collisions with walls and other vehicles are avoided.

Fig. 6 shows two representative examples of the distributed collision avoidance, where the driver-commanded velocity $\mathbf{u}^{joy}$ is shown with arrows in the right-side images. In the top, safe and smooth velocity between 1 and 2.5 m/s is achieved in very close proximity to the wall when the driver was commanding the vehicle towards it. The vehicle slightly slows down as it gets closer. In the bottom figure, a frontal collision is avoided where a relative velocity of about 5 m/s (18 km/h) in very close proximity (below 6 m) was handled. In this extreme case the optimization became unfeasible during 0.15 seconds, mostly due to lack of reactiveness in the low level controller and unmodeled dynamics, slowing the vehicle. This renders the algorithm feasible in subsequent time-steps.

Even if the velocity input by the driver seems to follow a bang-bang control of zero - maximum velocity (Fig. 7, bottom left) the algorithm adapts the velocity of the vehicle in the range -1 to 3 m/s to remain safe (Fig. 7, top left). The driver stops at some times. The angular velocity shows to be smooth and within the limits (Fig. 7, top right),

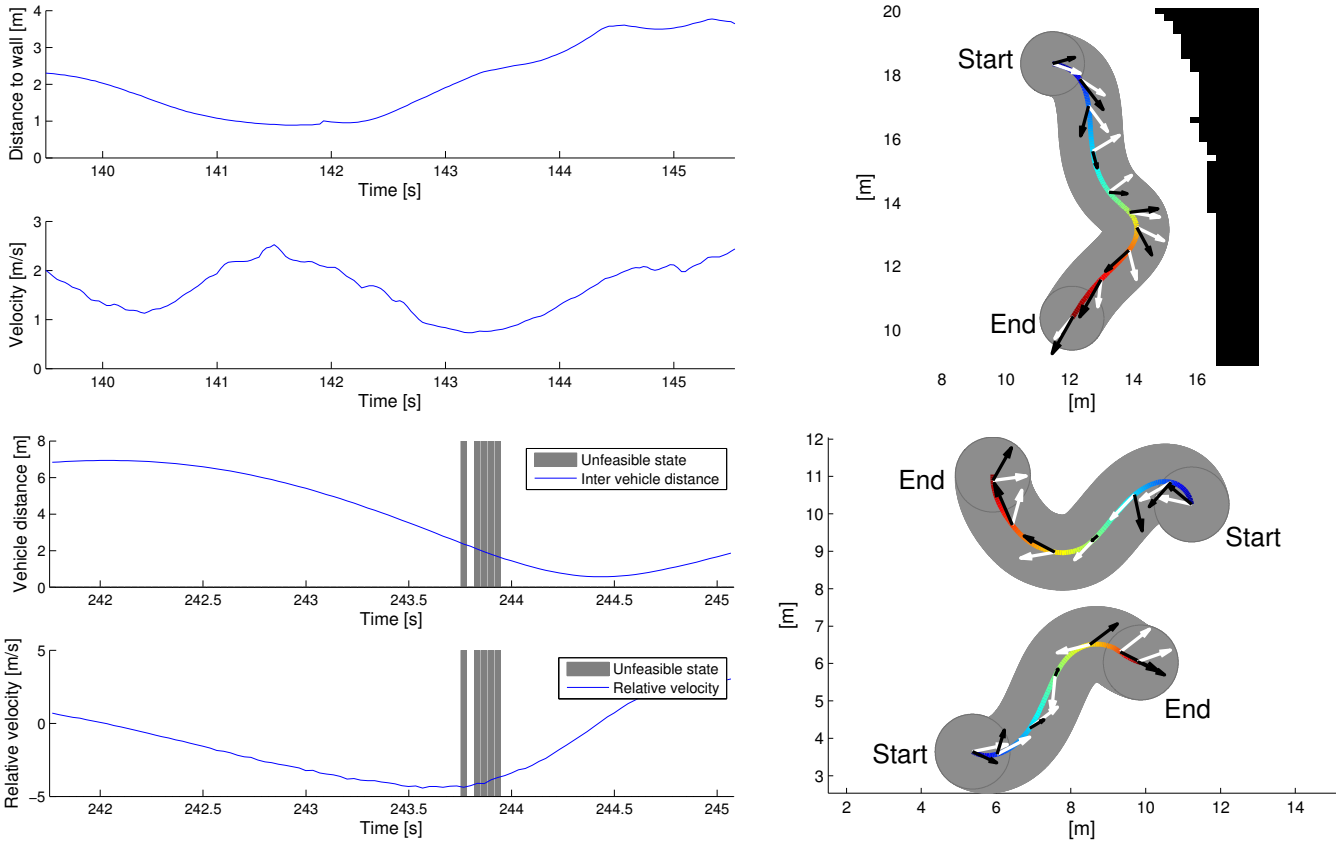

Fig. 6. *Top:* Example of robot - wall collision avoidance. *Bottom:* Example of robot-robot collision avoidance. *Left:* Distance and relative velocity (negative when towards each other). The time-steps where the optimization was unfeasible are marked in grey in the background. Zero distance indicates that two objects touch. *Right:* Vehicle position, the input velocity from the driver is displayed with white arrows and the safe vehicle command with black arrows.

while the orientation of $\mathbf{u}^{joy}$ (Fig. 7, top right, in vehicle reference frame) is mostly centered in the forward direction and presents clear peaks at 0, ± 90 and 180 degrees.

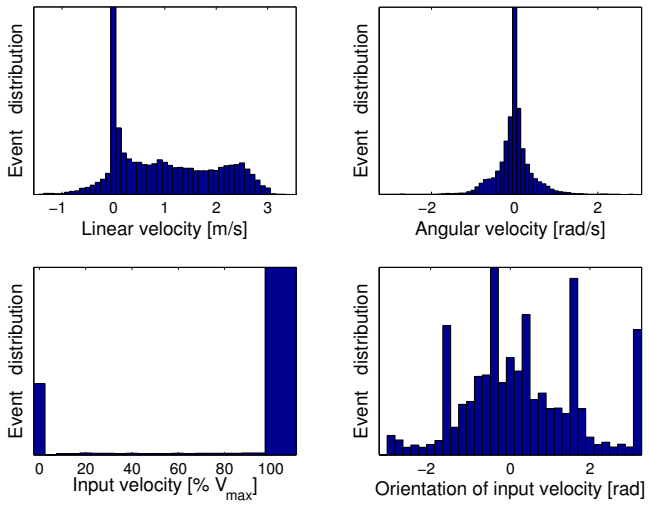

Fig. 7. Histograms over all experiments. Linear and angular velocities (top) and input velocity from the driver relative to the vehicle (bottom).

Fig. 8 shows, in log scale to emphasize the unfrequent worst cases, statistics of the computational time for the collision avoidance optimization. For the distributed algorithm (left, per vehicle) a solution is usually found in below 1.5ms. Higher times (below 6ms in all the experiments) depend on the non-convex search within the convex region (see Sec. VII). Similar computational time per agent is observed for experiments with larger number of vehicles in simulation. For the centralized algorithm (right, two vehicles) higher computational times where observed for the worst case, we believe that due to inefficient handling of high-dimensional sorted lists and other implementation inefficiencies.

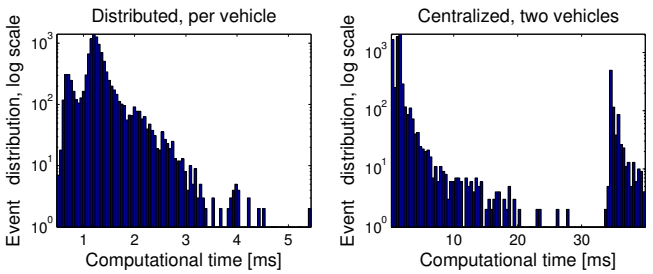

Fig. 8. Histogram in log-scale of the computational time of the collision avoidance algorithm. Distributed (left), and centralized (right). If no solution is found in 35ms, the algorithm returns unfeasible for that time step,

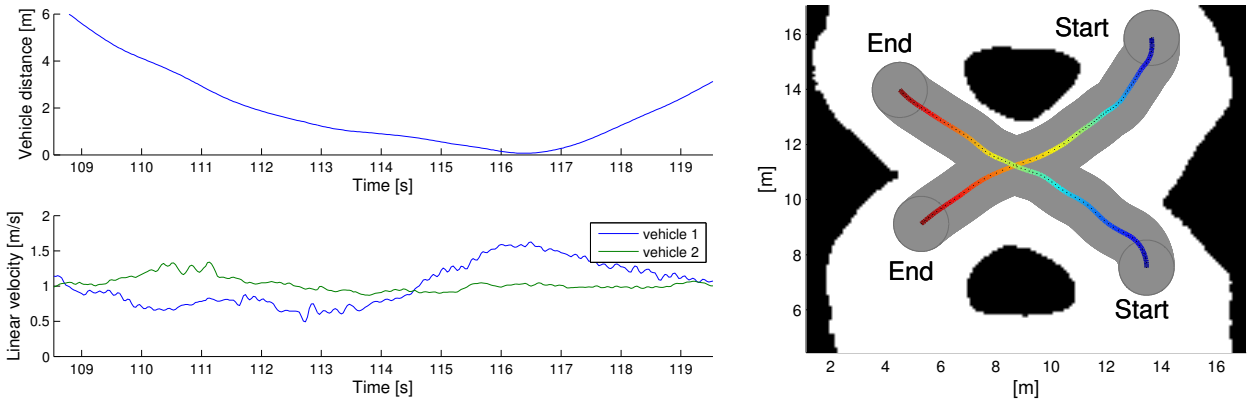

Fig. 9. Example of two vehicles without drivers following two guidance trajectories that intersect at the same time in the middle point. Vehicles distance and linear velocities (left), and position (right). Black: obstacles, grey: area covered by the vehicle, color line: path with time indicator.

B. Constrained driving with guidance trajectory

Fig. 9 shows an example of two vehicles without driver following two guidance trajectories that intersect at the same time in the middle point of an eight-shape scenario. Vehicle 2 accelerates within its active area, while vehicle 1 first decelerates to let vehicle 2 safely pass and then accelerates to catch up with its guidance trajectory. The position on the path is color coded according to time for comparison.

Fig. 10 shows statistics over several experiments of vehicle distance to the guidance trajectory relative to the active area radius. For a vehicle without driver (left) the 10% offset is due to the proportional tracking controller employed. For a vehicle with driver (right) it is observed that the driver is able to freely move within the active area but the algorithm successfully constraints the motion of the vehicle.

IX. Conclusion

A method for shared control of a vehicle has been presented where the driver commands the vehicle by specifying a preferred velocity. The specified velocity is then transformed into a local motion that respects the actuator limits and is collision-free with respect to other moving vehicles and static obstacles, given by a grid map. This allows for smooth and safe control of the vehicle. In order to limit the freedom of the driver, a global guidance trajectory can be included, which specifies the areas where the vehicle is allowed to drive in each time instance. Good performance has been observed in extensive experimental tests at speeds of up to 3 m/s and in close proximity to other vehicles and walls. Further, the low computational time of the algorithm allows for real-time control of tens of vehicles.

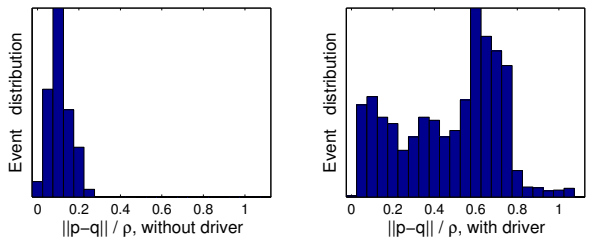

Fig. 10. Histogram of distance from vehicle to guidance trajectory $||\mathbf{p}-\mathbf{q}||$ relative to active area radius ρ. With (right) and without (left) driver.

References

[1] S. Bodenstedt, N. Padoy, and G. Hager, "Learned Partial Automation for Shared Control in Tele-Robotic Manipulation," *AAAI Fall Symposium Series*, 2012.

[2] A. Correa, M. R. Walter, L. Fletcher, J. Glass, S. Teller, and R. Davis, "Multimodal interaction with an autonomous forklift," in *Human-Robot Interaction, 5th ACM/IEEE International Conference on*, 2010.

[3] A. Franchi, C. Secchi, M. Ryll, H. H. Bulthoff, and P. R. Giordano, "Shared control: Balancing autonomy and human assistance with a group of quadrotor UAVs," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 57–68, 2012.

[4] J. Connell and P. Viola, "Cooperative control of a semi-autonomous mobile robot," *Robotics and Automation, Proceedings., IEEE International Conference on*, 1990.

[5] J. S. Nguyen, T. H. Nguyen, and H. T. Nguyen, "Semi-autonomous wheelchair system using stereoscopic cameras," pp. 5068–5071, 2009.

[6] B. M. Faria, L. Ferreira, L. P. Reis, N. Lau, M. Petry, and J. Couto, "Manual Control for Driving an Intelligent Wheelchair: A Comparative Study of Joystick Mapping Methods," *environment*, vol. 17, p. 18.

[7] A. Toffetti, E. S. Wilschut, M. H. Martens, A. Schieben, A. Rambaldini, N. Merat, and F. Flemisch, "CityMobil: Human Factor Issues Regarding Highly Automated Vehicles on eLane," *Journal of the Transportation Research Board*, Dec. 2009.

[8] H. G. Nguyen, A. J. Morrell, B. K. Mullens, A. A. Burmeister, S. Miles, C. N. Farrington, A. K. Thomas, and D. W. G. E, "Segway robotic mobility platform," in *in SPIE Mobile Robots XVII*, 2004.

[9] Techcrunch, "Google Ventures puts $258M into Uber," Aug. 2013.

[10] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2012.

[11] J. Alonso-Mora, M. Rufli, R. Siegwart, and P. Beardsley, "Collision Avoidance for Multiple Agents with Joint Utility Maximization," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[12] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal Collision Avoidance With Motion Continuity Constraints," *Robotics, IEEE Transactions on*, Mar. 2013.

[13] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot motion planning and control* (J.-P. Laumond, ed.), pp. 171–253, Springer, 1998.

[14] S. M. LaValle, "Motion Planning: The Essentials," *IEEE Robotics and Automation Society Magazine*, pp. 1–10, Apr. 2011.

[15] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the ACM*, 1985.

[16] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body Collision Avoidance," in *Int. Symp. on Robotics Research*, 2009.

[17] M. Rufli in *PhD thesis ETH Zurich*, 2012.