AutoRVO: Local Navigation with Dynamic Constraints in Dense Heterogeneous Traffic

Yuexin Ma[1], **Dinesh Manocha** [2], **Wenping Wang**[1]
[1] The University of Hong Kong, [2] University of North Carolina at Chapel Hill
yxma@cs.hku.hk, dm@cs.unc.edu, wenping@cs.hku.hk

Abstract

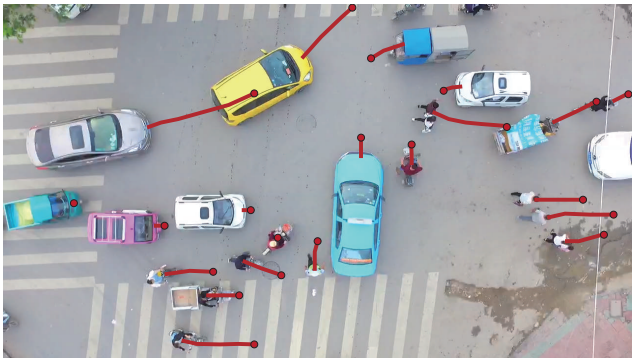
We present a novel algorithm for computing collision-free navigation for heterogeneous road-agents such as cars, tricycles, bicycles, and pedestrians in dense traffic. Our approach currently assumes the positions, shapes, and velocities of all vehicles and pedestrians are known and computes smooth trajectories for each agent by taking into account the dynamic constraints. We describe an efficient optimization-based algorithm for each road-agent based on reciprocal velocity obstacles that takes into account kinematic and dynamic constraints. Our algorithm uses tight fitting shape representations based on medial axis to compute collision-free trajectories in dense traffic situations. We evaluate the performance of our algorithm in real-world dense traffic scenarios and highlight the benefits over prior reciprocal collision avoidance schemes.

1 Introduction

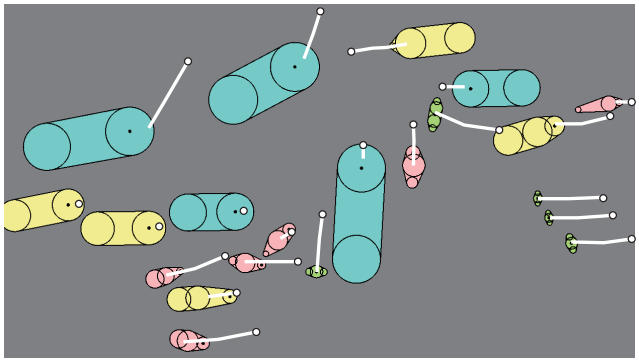
Multi-agent local navigation is an important problem in robotics, crowd simulation, and traffic modeling. At a broad level, the goal is to compute a collision-free trajectory for each agent in a distributed manner. Furthermore, it is important to satisfy other constraints corresponding to kinematics, dynamics, and smoothness.

Some of the most widely used algorithms for decentralized multi-agent navigation are based on velocity obstacles [Fiorini and Shiller, 1998]. They have been extended to perform reciprocal collision avoidance between a large number of active agents and applied to simulate human-like crowds [Berg *et al.*, 2011] and multiple car-like robots [Alonso-Mora *et al.*, 2012]. Furthermore, they have been extended to take dynamic constraints into account [Wilkie *et al.*, 2009; Bareiss and van den Berg, 2015].

There is considerable interest in developing multi-agent navigation algorithms for autonomous driving and simulating real-world traffic scenarios [Ziegler *et al.*, 2014; Turri *et al.*, 2013; Best *et al.*, 2017]. These algorithms consider dynamic constraints of vehicles, environmental factors, and traffic rules. However, current autonomous driving navigation algorithms are limited to simple scenarios with sparse traffic or few vehicles that are moving in lanes and simple traffic conditions. They do not model the movement of pedestrians or bicycles or their close interactions with vehicles or two-wheelers (i.e. road-agents), and instead maintain large



(a)



(b)

Figure 1: **Dense traffic and navigation**: (a) One frame of a top-down-view dense traffic video with different vehicles and pedestrians from a real-world scene. We highlight different road-agents in this dense scenario. Red curves show trajectories of road-agents in 50 frames of the video; (b) Simulated traffic scenario with our medial-axis-based agent representation agent representation. We model heterogeneous road-agents corresponding to pedestrians, bicycles, tricycles, and cars, as green, pink, yellow, and blue, respectively. Our algorithm, AutoRVO, models the dynamics of these agents and computes collision-free trajectories in similar time of (a) shown as white curves. We observe high accuracy with real-world vehicle and pedestrian trajectories.

distances for safety. As a result, current autonomous driving simulators or systems cannot model dense traffic scenarios with heterogeneous road-agents of varying shapes and dynamics.

Most prior work on reciprocal collision avoidance is limited to simple agent shapes, including circles or ellipses. They use geometric properties of these simple shapes to design efficient navigation algorithms. However, such disk-based agent representations can be overly conservative in dense traffic scenarios which may consist of large or small vehicles, pedestrians, bicycles, two-wheelers, etc. in close proximity to each other(Fig. 2). As a result, we need more efficient and less conservative multi-agent navigation algorithms that can simulate real-world traffic scenarios.

Main Contributions: We present a novel multi-agent simulation algorithm, AutoRVO, for local navigation with dynamic constraints in dense, heterogeneous scenarios. Our approach is designed for traffic-like environments, which have different agents of varying shapes and different dynamic constraints. Our approach assumes that the exact positions and velocities of all the agents are known. Our algorithm uses the CTMAT-based agent representation [Ma *et al.*, 2018] and computes a new velocity for each agent based on local optimization. A key aspect of our approach is that it can handle the non-linear dynamics of different vehicles and compute collision-free trajectories in dense situations without making any assumptions about their movement patterns or trajectories. We have evaluated the performance of our algorithm on real-world traffic scenarios with vehicles that are different in terms of size and dynamic constraints, pedestrians, and bicycles. We computed their trajectories using AutoRVO and compared them with the real-world trajectories extracted using a tracking algorithm from a video. We evaluate the accuracy based on the Entropy metric and observe considerable improvement over prior multi-agent navigation schemes. Overall, AutoRVO is the first algorithm that can generate collision-free trajectories for road-agents in dense scenarios.

The rest of the paper is organized as follows. Section 2 offers an overview of related work in local navigation with dynamic constraints. We introduce the kinematic model and state space of vehicles and road-agents in Section 3. In Section 4, we give a detailed description of our novel multi-agent navigation algorithm, AutoRVO. We highlight the performance of our algorithm on challenging scenarios in Section 5 and analyze algorithm complexity in Section 6.

2 Related Work

In this section, we give a brief overview of prior work on collision avoidance, motion planning, kinematic and dynamic modeling, and autonomous navigation.

2.1 Multi-Agent Navigation using Velocity Obstacles

Based on the approach of velocity obstacles (VO) [Fiorini and Shiller, 1998], reciprocal collision avoidance ORCA [Berg *et al.*, 2011] allows each agent to take half of the responsibility for avoiding pairwise collisions. In the ORCA algorithm, each agent has the same state space and can change its velocity instantaneously. Some subsequent approaches take various dynamic constraints into consideration, such as AVO algorithm [Berg *et al.*, 2011], which computes free and collision-free velocities by using velocity-space reasoning with acceleration constraints; CCO algorithm [Rufli *et al.*, 2013], which considers continuous control obstacles; and LQR-obstacles algorithm [Bareiss and den Berg, 2013]. All these techniques are designed for circular agents. The ORCA algorithm has been extended to elliptical agents [Best *et al.*, 2016].

(a) (b)

Figure 2: Comparison between the representation of disk (black) and our medial-axis-based representation (CTMAT in red). Our representation is less conservative and can accurately model such dense traffic scenarios. On the other hand, disk-based representations are overly conservative and unable to compute collision free trajectories.

2.2 Kinematic and Dynamic Modeling

There are many approaches to model vehicles with kinematic and dynamic constraints. Some of the simplest methods are based on the linear dynamics of vehicles [LaValle, 2006], but these may not be accurate. Other methods make use of non-linear dynamic forces [Borrelli *et al.*, 2005], which are more accurate, but the resulting algorithms are more time consuming. The Reeds-Shepp formulation [Reeds and Shepp, 1990] supports the forward and backward motion of a car. Other kinematic and dynamic models for a moving car are described in [Margolis and Asgari, 1991].

Many velocity-obstacle-based methods have been extended to account for dynamic constraints, including differential-drive [Alonso-Mora *et al.*, 2013], double-integrator [Lalish and Morgansen, 2012], arbitrary integrator [Rufli *et al.*, 2013], car-like [Alonso-Mora *et al.*, 2012], linear quadratic regulator (LQR) controllers [Bareiss and den Berg, 2013], non-linear equations of motion [Bareiss and van den Berg, 2015], etc. Some other algorithms, like NH-ORCA [Alonso-Mora *et al.*, 2013], transfer non-linear equations of motion into a linear formulation. Based on non-linear velocity obstacles NLVO [Shiller *et al.*, 2001], and GVO [Wilkie *et al.*, 2009], GRVO [Bareiss and van den Berg, 2015] can account for non-homogeneous agents with non-linear equations of motion. However, all these methods are restricted to simple disc-based representations and can be overly conservative in terms of handling dense scenarios and heterogeneous agents.

2.3 Maneuver Planning for Autonomous Driving

There is considerable work on maneuver planning of autonomous vehicles, including driving corridors [Hardy and Campbell, 2013], potential-field methods [Galceran *et al.*,

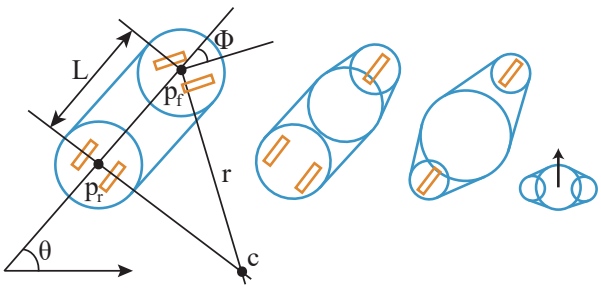


Figure 3: **Representation of vehicles and pedestrians**. From left to right are the agent representations for a car, a tricycle, a bicycle, and a pedestrian. The blue shape is our CTMAT agent representation, the brown rectangles denote the tires, and the black arrow on the pedestrian indicates the forward-facing direction.

2015b], random-exploration [Kuwata *et al.*, 2009], occupancy grids methods [Kolski *et al.*, 2006], etc. Some approaches limit the vehicles to staying in lanes to avoid collisions with obstacles or other vehicles [Turri *et al.*, 2013; Fritz *et al.*, 2004] or consider a driver's behaviors [Sadigh *et al.*, 2016; Galceran *et al.*, 2015a; Best *et al.*, 2017].

3 Problem Formulation and Notation

In this section, we introduce the notation, representation of road-agents, kinematic and dynamic models of different vehicles, and their state space.

3.1 Representation and Kinematic Models

To represent different shapes corresponding to heterogeneous vehicles in the real world, we use the CTMAT representation [Ma *et al.*, 2018]. CTMAT is a medial-axis-based representation that can provide a tighter fitting shape for different road-agents. The underlying representation consists of circles and tangent line segments between any pair of adjacent circles. Fig. 3 shows four examples of CTMAT for different vehicles and pedestrians. A key issue is to model the dynamic constraints of different agents. Therefore, we extend the simple-car kinematic model [LaValle, 2006; Laumond *et al.*, 1998] to different vehicle or agent types. As the figure indicates, if the steering wheel is turned, the vehicle will rotate around the center c, which is determined by the steering angle ϕ and body length L. Let's assume that the orientation is represented as θ and the speed is denoted as v. The vehicle's motion can be denoted as follows:

$$\dot{\vec{p}} = (v\cos(\theta), v\sin(\theta)), \qquad \dot{\theta} = \frac{\tan(\phi)}{L}v.$$

We give more details on computing $\dot{v}$ and $\dot{\phi}$ in Section 4. Pedestrians do not exhibit steering behaviors as their dynamic constraints are different from that of vehicles. Instead, they can change their orientation instantly and always move according to their forward-facing direction.

3.2 State Space

The simulator state includes all the entities in the scenario, including all obstacles and agents. We always use an n-dimensional space to describe an agent's physical state and properties. Our approach is designed for different road-agents corresponding to pedestrians, bicycles, tricycles, and cars with different shapes. The state space of the pedestrians is denoted as $X_p = \{T, v, v^o, \theta, \theta^o\}$. T records the components of CTMAT representation, including circles and their tangent line segments. v and θ denote current speed and orientation, respectively. v^o and θ^o are the preferred speed and the preferred orientation, respectively.

The bicycles, tricycles, and cars have kinematic and dynamic constraints on their turning motion. The state space for vehicles is represented as $X_v = \{T, p_f, p_r, v, \phi, v^o, \phi^o, u_t, u_\phi, \theta, b\}$ (see Fig. 3). T also stands for the CTMAT representation like that of pedestrians. p_f is the position of the front wheel for bicycles and tricycles, or the position of the middle point between two front wheels for cars. p_r is similar to p_f but for rear wheels. v and ϕ represent vehicles' speed and steering, respectively. v^o and ϕ^o are preferred speed and steering, respectively. Every vehicle has two degrees of control, throttle u_t and steering u_ϕ. We define $-1 \leq u_t \leq 1$, where -1 denotes the maximum braking effort and 1 represents the maximum throttle. $-1 \leq u_\phi \leq 1$ indicates the steering effort from $-\phi_{max}$ to ϕ_{max}. The boundary values of these dynamic variables are distinctive for different types of vehicles. θ stands for the orientation. b is a label to record a vehicle's current behavior (turn left or turn right, wait, or go ahead), which is related to the choice of dynamic constraints.

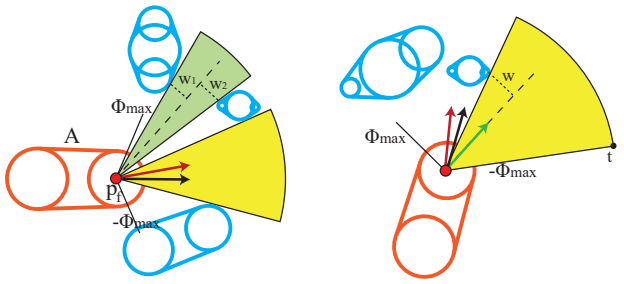
In addition, we define a label C_{type} to record the road-agent's type, i.e. 1 for pedestrian, 2 for bicycle, so that they are distinguishable from each other. We regard the middle point of T as the reference point of pedestrians and p_f as the reference point of vehicles.

4 AutoRVO: Our Navigation Algorithm

We assume that each road-agent has smart sensors that can capture surrounding environmental information such as nearby obstacles and the current speed, steering, position, and orientation of other agents. Our approach is based on a reciprocal collision avoidance method and uses optimization method to compute a local trajectory. In particular, our algorithm proceeds using three main steps: first, we compute the preferred speed and steering (v^o, ϕ^o) for vehicles and preferred speed and orientation (v^o, θ^o) for pedestrians. Second, we sample around (v^o, ϕ^o) or (v^o, θ^o) to get a set of solution candidates for new speed and steering or orientation. Finally, we use an optimization function to select the best solution for (v, ϕ) or (v, θ). After that, we update the state of each road-agent and change its trajectory for time τ. For each step, we present the details for the vehicles, and then explain the differences for pedestrians.

4.1 Preferred Velocity Computation

The trajectory is determined by the velocity of the road-agent. The velocity depends on the speed and steering for vehicles, and hinges on the speed and orientation for pedestrians. Before choosing the velocity for a road-agent, we compute parameters (v^o, ϕ^o) or (v^o, θ^o) first to guide the final velocity selection.



(a) (b)

Figure 4: **Search for free-space for collision-free local navigation**. The black arrow represents the orientation direction $\vec{a}$, the red arrow is the destination direction $\vec{h}$, and the green arrow is the preferred direction $\vec{d^o}$. $\vec{d^o}$ coincides with $\vec{h}$ in (a). All the detected fan spaces for the road-agent in the 2D plane are drawn in green and yellow. The yellow fan denotes the free-space.

Preferred Steering Computation

First, we compute preferred steering ϕ^o, which is the preferred angle for turning left or right. Let $\vec{a}$ denote the orientation direction of the road-agent. For pedestrians, $\vec{a}$ is the forward-facing direction. For vehicles, $\vec{a} = p_f - p_r$. We define the direction to the destination as $\vec{h}$ and the preferred direction as $\vec{d^o}$. In general, $\vec{d^o} = \vec{h}$. However, in dense traffic scenarios, it is sometimes a good strategy to find a detour space.

The range of the steering $(-\phi_{max}, \phi_{max})$, the considering distance for neighbors and the set of neighbors of the agent are used to compute a set of fan spaces for a vehicle A (Fig. 4). These fan spaces do not contain any obstacle or other road-agent in the detection range. The width of the fan space can be computed by two tangent points of the neighbors and their distance to the middle line of the space. For the green fan space in Fig. 4(a), the width is the sum of w_1 and w_2. If the side of a space is decided by $-\phi_{max}$ or ϕ_{max}, like the yellow fan in Fig. 4(b), the tangent point can be replaced by the vertex t of the fan. We regard a fan space as a free-space where is uncrowded, if its width is σ times bigger than the road-agent's width ($\sigma = 1.5$ in our benchmarks). If there is no feasible space or $\vec{h}$ is already in a free-space (Fig. 4(a)), we set $\vec{d^o} = \vec{h}$. Otherwise, $\vec{d^o}$ changes to the green direction (see Fig. 4(b)), which satisfies the requirement that w is one half of the width of A. Next, according to the angle between $\vec{a}$ and $\vec{d^o}$, we can compute ϕ^o for vehicles by a dynamic formulation.

$$\phi^o = f(\vec{a}, \vec{d^o}), \phi^o \in (-\phi_{max}, \phi_{max}) \tag{1}$$

f differs for different types of vehicles and could be computed using real-world data. In terms of pedestrians, they will change their orientation to $\vec{d^o}$ directly.

Preferred Speed Computation

When $\phi \neq 0$, vehicles will move along a circle C with radius r and center c (see Fig. 3). According to the centripetal force equation, we can compute the upper bound of the speed:

$$v_{max_1} = \sqrt{g\mu r} \tag{2}$$

where g is the acceleration due to gravity and μ is the friction force coefficient. If we take the minimum distance between the road-agent and its neighbors in its current moving direction as l. According to vehicle's braking control, we can compute another upper bound of speed:

$$l = l_{react} + l_{braking} = v_{max_2} t + \frac{{v_{max_2}}^2}{2g\mu} \tag{3}$$

where l_{react} is the perception-reaction distance, $l_{braking}$ indicates the braking distance, and t is the time for increasing braking force ($t = 1.5$, $\mu = 0.7$ for common baseline value). l reflects the minimum safe distance for a specified speed under the dynamic constraint of the braking system of the vehicle. Moreover, each vehicle has its own maximum speed v_{max_3}. Therefore, we can get the maximum speed for the road-agent under a specified steering ϕ:

$$v_{max}(\phi) = min\{v_{max_1}, v_{max_2}, v_{max_3}\} \tag{4}$$

For pedestrians, we only need to consider v_{max_2} and v_{max_3}. We choose $v^o = v_{max}/2$ in our benchmarks.

Velocity Prediction

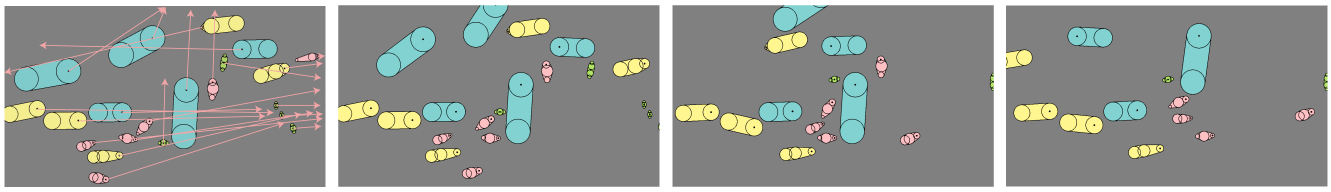
In the real world, a driver or a pedestrian always has the ability to predict the state and consider that prediction before making further decisions. We therefore compute the state space of the neighboring agents after a time interval κ, and then compute the free-spaces for the road-agent once again. If the road-agent's $\vec{h}$ was not in a free-space, but after time κ, $\vec{h}$ is in a free-space, it will choose to stop moving in the next state update, because waiting in such a situation will help the road-agent avoid unnecessary detouring behavior and save energy. If the road-agent's $\vec{h}$ was in a free-space, but after time κ, $\vec{h}$ is no longer in a free-space, the road-agent will speed up within a reasonable range, because it should pass the uncrowded space as soon as possible to avoid being locked after time κ.

4.2 Velocity Sampling

For disc-based representation, we can use generalized velocity obstacle (GVO) [Bareiss and van den Berg, 2015] to compute the new collision-free velocity for each vehicle directly. However, such a disk representation can be too conservative. Instead, we use the CTMAT representation, but the resulting computation of the exact boundary of a control obstacle is too expensive. Since we have already know the preferred speed v^o and steering ϕ^o of the vehicles, we can use a sampling approach to search for a better solution for (v, ϕ). The sampling range can be defined as follows.

$$\begin{aligned}(v_{min}, v_{max}) &= T(v^o, \tau),\\ (\phi_{min}, \phi_{max}) &= S(\phi^o, \tau),\end{aligned} \tag{5}$$

where T is the dynamic function to get the speed range when the vehicle is using the highest throttle and braking effort for time interval τ. S is the dynamic function to compute the steering range for next τ time. We perform even sampling in this range. In particular, we choose the collision-free samples as candidates by using the Minkowski sum of CTMAT between the road-agent and its neighbors. We also use Equation 2 to further filter candidates for (v, ϕ). We use the same method to sample (v, θ) for pedestrians.



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

Figure 5: A sequence of frames of a simulated dense traffic scenario from the input of Fig. 1. Red arrows represent designated destinations according to the real video. AutoRVO can compute collision-free trajectories for all road-agents in such dense traffic scenarios.

4.3 Trajectory Computation

After computing a set of candidates, we use the following cost function to select the best solution for (v, ϕ) or (v, θ).

$$\min F = af_1 + bf_2 + cf_3 + df_4 + ef_5, \tag{6}$$

where a, b, c, d, e are coefficients that can be adjusted. They are the weights of making trajectories smoother or safer or faster to arrive the destination.

$$f_1 = (v - v^o)^2 + (\phi - \phi^o)^2, \tag{7}$$

f_1 indicates the distance to v^o and ϕ^o. This term is used to select a solution close to the computed preferred speed and steering.

$$f_2 = |v - v'| + |\phi - \phi'|, \tag{8}$$

f_2 denotes the most recent changes to the previous speed and steering. We use this term to control the changes of vehicles' behaviors and results in smoother trajectories for the vehicles. (ϕ, ϕ^o, ϕ') are replaced by $(\theta, \theta^o, \theta')$ for pedestrians in f_1 and f_2.

$$f_3 = -\sum_{n=1}^{N} (1 + C_{type} - N(n)_{type}) \cdot \|p - p_n\|, \tag{9}$$

where $N(n)_{type}$ is defined as the nth neighbor of the road-agent. p is the vehicle's position if current candidate for (v, ϕ) or (v, θ) is adopted. p_n is its nth neighbor's position under the assumption that they proceed at their current speed and steering. f_3 denotes an attempt to keep the vehicle away from nearby pedestrians, vehicles, or obstacles.

$$f_4 = -\sum_{n=1}^{N} d(S_n, origin), \tag{10}$$

We compute whether the coordinate's origin lies in the Minkowski sum of the vehicle and each of its neighbors to determine whether the sample (v, ϕ) is collision-free. The probability of causing a collision is lower if the distance from the origin to the S is bigger. We use f_4 to reduce the risk of collision.

$$f_5 = d(p, Goal), \tag{11}$$

f_5 stands for the distance to the destination. This term is related to energy consumption. It is better to reach the destination as soon as possible to save the passenger time and to save the vehicle electricity or fuel.

Scenario	V	P	T	AutoRVO	CND	ORCA
traffic-1	16	5	4	3.77	12.72	15.11
traffic-2	12	2	4	2.56	5.34	8.12
traffic-3	8	2	3	2.69	8.98	10.13
traffic-4	10	1	4	4.25	4.03	5.41
traffic-5	15	1	4	2.45	5.77	14.12
traffic-6	8	2	3	3.33	4.33	5.63

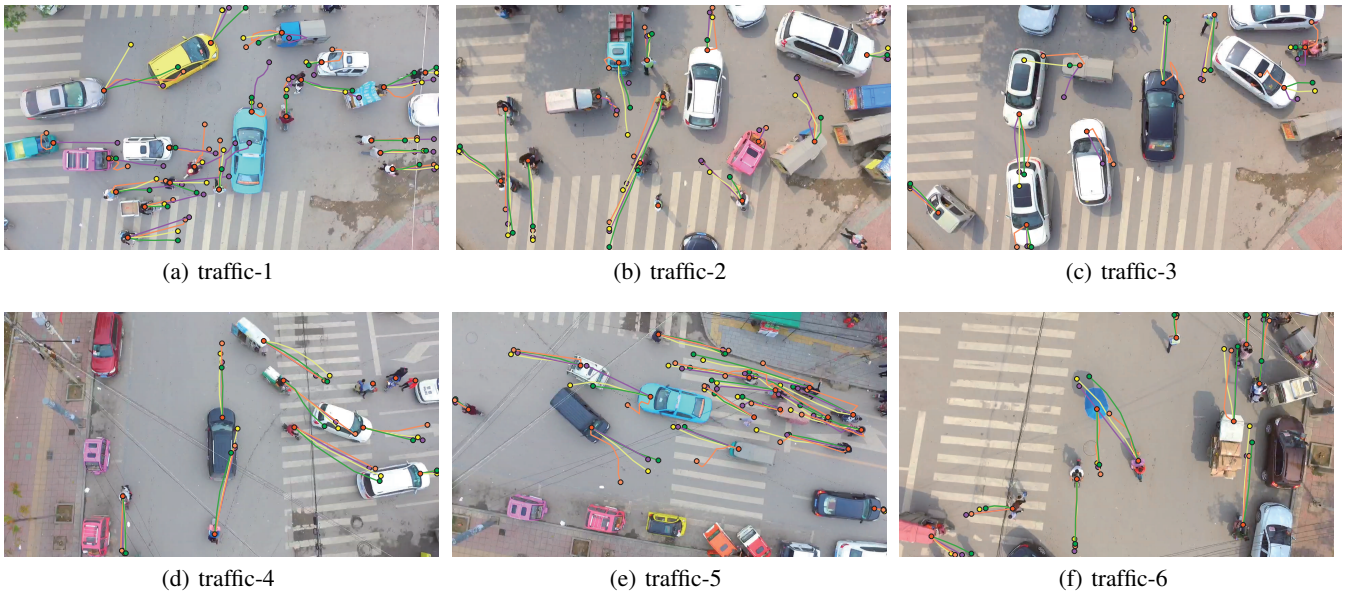
Table 1: Evaluation of different multi-agent navigation algorithms on dense traffic scenarios shown in Fig. 6. We show the total number of vehicles in the second column and the number of pedestrians in the third column. The number of different types of road-agents is shown in the third column. The last three columns illustrate the Entropy metric (lower is better for validation) for different simulation results by our algorithm AutoRVO, CTMAT representation with no dynamics (CND), and ORCA algorithm with disk representation.

5 Results

In this section, we highlight the performance of our algorithm in local navigation with dynamic constraints in dense scenarios with heterogeneous vehicles. All the traffic scenarios are from a city traffic scene and the original traffic images (Fig. 1 and 6) were captured using a drone camera.

Fig. 5 shows a sequence of frames in our simulation of dense traffic with different vehicles. Before we run our algorithm, we select any frame in one video as the input and then compute CTMAT representation according to the road-agents' contours (see Fig. 1). Because the view for a given camera is limited, we assign goal positions for road-agents based on the corresponding positions where they stop or disappear in the video. We represent the destinations of vehicles and pedestrians using the red arrows in Fig. 5(a). These traffic scenarios are very dense and include various types of vehicles and pedestrians. As we can see from the navigation results generated by our algorithm, all the pedestrians and vehicles move in collision-free trajectories and behave realistically waiting or detouring behaviors without creating any gridlocks or congestion scenarios.

Fig. 6 shows comparisons between road-agents' trajectories and simulated trajectories of AutoRVO, CTMAT representation without dynamic constraints (denoted as CND) and ORCA. We use 50 continuous frames of a video as one sample to make the comparisons. For each sample, we take the first frame as input and use the positions of road-agents when they disappear or their positions after 100 frames as the destinations. Then, we select similar number of frames or discrete positions of simulated video for comparison. We can see from the simulation results by AutoRVO that, apart from exhibit-



(a) traffic-1 (b) traffic-2 (c) traffic-3

(d) traffic-4 (e) traffic-5 (f) traffic-6

Figure 6: Comparison of real trajectories of 50 continuous frames and simulated trajectories. (a)-(c) are three different moments from one video. (d)-(f) are three moments from three different videos. Green lines indicate the real trajectories extracted from videos captured using a drone. Trajectories generated by AutoRVO, CTMAT representation with no dynamics (CND), and ORCA with disk representation are drawn in yellow, purple, and orange respectively. We observe higher accuracy with AutoRVO. Red points represent beginning reference positions.

ing similar trajectories, some road-agents wait during this period as in real-world scenarios, which means our prediction ability also works in solving congestion. In Table 1, we use Entropy metric [Guy *et al.*, 2012] to measure the similarity between simulated trajectories by three algorithms and real trajectories. The Entropy metric compares the accuracy of the trajectories computed by different simulated algorithm with real-world trajectories extracted from videos. A lower value of Entropy metric indicates higher accuracy (as observed for AutoRVO). By adding dynamic constraints, we observe considerable accuracy improvement in AutoRVO over CND. The disk representation in ORCA-based methods cannot compute accurate trajectories in such cases. Especially for denser scenarios like traffic-1 and traffic-5, the Entropy Metric values of CND and ORCA are much bigger than that of AutoRVO, which illustrates our capability in handling dense traffic situations.

6 Runtime Analysis

We implemented the algorithm in C++ and conducted experiments on a Windows 10 laptop with an Intel i7-6700 CPU and 8GB RAM. Our algorithm can be parallelized on multiple cores, but we generate all the results on a single CPU core. For the benchmarks shown in Fig. 5, the average number of neighbors for a vehicle is 4, and the average time for updating the state of one road-agent is about 10ms with about 100 sampling candidates of (v, ϕ). With the number of neighbors or the number of samples increase, the simulation time per frame increases linearly. The run time R for updating the state space for road-agent A is given as:

$$R(A) = N \cdot M \cdot t_{sum}(A) + t_{search},$$

where N is the the number of neighboring obstacles and other road-agents of A, M is the number of sampling points of (v, ϕ), and $t_{sum}(A)$ indicates the average time of computing the Minkowski sum of a pair of CTMAT. t_{search} is $O(N)$ for searching detour space. Therefore, R is $O(NM)$.

7 Conclusion and limitations

We present a novel algorithm, AutoRVO, for local navigation of heterogeneous vehicles and pedestrians in dense traffic situations with kinematic and dynamic constraints. Our formulation is based on a tight-fitting media-axis-based agent representation and we present an efficient algorithm to handle kinematic and dynamic constraints of different vehicles and pedestrians. We use an optimization-based local planning method to help road-agents choose a velocity that would result in smooth trajectories.. We have demonstrated the performance of our algorithm in the simulation of dense traffic scenarios and compared its performance with the trajectories of real-world road-agents and other algorithms.

Our approach has some limitations. We assume perfect sensing abilities in terms of the exact position and velocity of all road-agents. In terms of dynamic constraints, we make use of empirical values for some parameters in our equations corresponding to the motion computation, which may different from the real-world data in the videos. In the future, we would like to consider sensor errors and extend our algorithm to handling noisy perception data. Furthermore, we will collect data from different types of vehicles and environmental information to make the dynamic constraints of road-agents closer to real conditions.

References

[Alonso-Mora *et al.*, 2012] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Reciprocal collision avoidance for multiple car-like robots. In *ICRA, 2012 IEEE International Conference on*, pages 360–366. IEEE, 2012.

[Alonso-Mora *et al.*, 2013] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed Autonomous Robotic Systems*, pages 203–216. Springer, 2013.

[Bareiss and den Berg, 2013] D. Bareiss and J. Van den Berg. Reciprocal collision avoidance for robots with linear dynamics using lqr-obstacles. In *ICRA, 2013 IEEE International Conference on*, pages 3847–3853. IEEE, 2013.

[Bareiss and van den Berg, 2015] D. Bareiss and J. van den Berg. Generalized reciprocal collision avoidance. *IJRR*, 34(12):1501–1514, 2015.

[Berg *et al.*, 2011] J. Van Den Berg, J. Snape, S.J. Guy, and D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. In *ICRA, 2011 IEEE International Conference on*, pages 3475–3482. IEEE, 2011.

[Best *et al.*, 2016] A. Best, S. Narang, and D. Manocha. Real-time reciprocal collision avoidance with elliptical agents. In *ICRA, 2016 IEEE International Conference on*, pages 298–305. IEEE, 2016.

[Best *et al.*, 2017] A. Best, S. Narang, L. Pasqualin, D. Barber, and D. Manocha. Autonovi: Autonomous vehicle planning with dynamic maneuvers and traffic constraints. *arXiv preprint arXiv:1703.08561*, 2017.

[Borrelli *et al.*, 2005] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat. Mpc-based approach to active steering for autonomous vehicle systems. *IJVAS*, 3(2-4):265–291, 2005.

[Fiorini and Shiller, 1998] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *IJRR*, 17(7):760–772, 1998.

[Fritz *et al.*, 2004] H. Fritz, A. Gern, H. Schiemenz, and C. Bonnet. Chauffeur assistant: a driver assistance system for commercial vehicles based on fusion of advanced acc and lane keeping. In *IV, 2004 IEEE*, pages 495–500. IEEE, 2004.

[Galceran *et al.*, 2015a] E. Galceran, A.G. Cunningham, R.M. Eustice, and E. Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction. In *RSS*, 2015.

[Galceran *et al.*, 2015b] E. Galceran, R.M. Eustice, and E. Olson. Toward integrated motion planning and control using potential fields and torque-based steering actuation for autonomous driving. In *IV, 2015 IEEE*, pages 304–309. IEEE, 2015.

[Guy *et al.*, 2012] S.J. Guy, J. Van Den Berg, W.X. Liu, R. Lau, M.C. Lin, and D. Manocha. A statistical similarity measure for aggregate crowd dynamics. *TOG*, 31(6):190, 2012.

[Hardy and Campbell, 2013] J. Hardy and M. Campbell. Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. *IEEE Transactions on Robotics*, 29(4):913–929, 2013.

[Kolski *et al.*, 2006] S. Kolski, D. Ferguson, M. Bellino, and R. Siegwart. Autonomous driving in structured and unstructured environments. In *IV, 2006 IEEE*, pages 558–563. IEEE, 2006.

[Kuwata *et al.*, 2009] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J.P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[Lalish and Morgansen, 2012] E. Lalish and K.A. Morgansen. Distributed reactive collision avoidance. *Autonomous Robots*, 32(3):207–226, 2012.

[Laumond *et al.*, 1998] Jean-Paul Laumond, S Sekhavat, and F Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. *Robot motion planning and control*, pages 1–53, 1998.

[LaValle, 2006] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[Ma *et al.*, 2018] Y. Ma, D. Manocha, and W. Wang. Efficient reciprocal collision avoidance between heterogeneous agents using ctmat. `http://mayuexin.me/zxg_css/AAMAS_Final.pdf`, 2018.

[Margolis and Asgari, 1991] D.L. Margolis and J. Asgari. Multipurpose models of vehicle dynamics for controller design. Technical report, SAE Technical Paper, 1991.

[Reeds and Shepp, 1990] J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

[Rufli *et al.*, 2013] M. Rufli, J. Alonso-Mora, and R. Siegwart. Reciprocal collision avoidance with motion continuity constraints. *T-RO*, 29(4):899–912, 2013.

[Sadigh *et al.*, 2016] D. Sadigh, S. Sastry, S.A. Seshia, and A.D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *RSS*, 2016.

[Shiller *et al.*, 2001] Z. Shiller, F. Large, and S. Sekhavat. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *ICRA. IEEE International Conference on*, volume 4, pages 3716–3721. IEEE, 2001.

[Turri *et al.*, 2013] V. Turri, A. Carvalho, H.E. Tseng, K.H. Johansson, and F. Borrelli. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. In *ITSC, 2013 16th International IEEE Conference on*, pages 378–383. IEEE, 2013.

[Wilkie *et al.*, 2009] D. Wilkie, J. Van Den Berg, and D. Manocha. Generalized velocity obstacles. In *IROS 2009. IEEE/RSJ International Conference on*, pages 5573–5578. IEEE, 2009.

[Ziegler *et al.*, 2014] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke,

N. Appenrodt, C.G. Keller, et al. Making bertha drive—an autonomous journey on a historic route. *ITSM*, 6(2):8–20, 2014.