Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 501689, 12 pages
http://dx.doi.org/10.1155/2014/501689

Research Article

Optimal Acceleration-Velocity-Bounded Trajectory Planning in Dynamic Crowd Simulation

Fu Yue-wen,[1] Li Meng,[2] Liang Jia-hong,[1] and Hu Xiao-qian[1]

[1] *College of Information System and Management, National University of Defense Technology, Changsha, Hunan 410073, China*
[2] *Army Officer Academy of PLA, Hefei 230031, China*

Correspondence should be addressed to Fu Yue-wen; thdinf@126.com

Received 24 April 2014; Accepted 28 July 2014; Published 31 August 2014

Academic Editor: Yuxin Zhao

Creating complex and realistic crowd behaviors, such as pedestrian navigation behavior with dynamic obstacles, is a difficult and time consuming task. In this paper, we study one special type of crowd which is composed of urgent individuals, normal individuals, and normal groups. We use three steps to construct the crowd simulation in dynamic environment. The first one is that the urgent individuals move forward along a given path around dynamic obstacles and other crowd members. An optimal acceleration-velocity-bounded trajectory planning method is utilized to model their behaviors, which ensures that the durations of the generated trajectories are minimal and the urgent individuals are collision-free with dynamic obstacles (e.g., dynamic vehicles). In the second step, a pushing model is adopted to simulate the interactions between urgent members and normal ones, which ensures that the computational cost of the optimal trajectory planning is acceptable. The third step is obligated to imitate the interactions among normal members using collision avoidance behavior and flocking behavior. Various simulation results demonstrate that these three steps give realistic crowd phenomenon just like the real world.

1. Introduction

Crowds, ubiquitous in the real world from groups of humans to flocks of insects, are vital features to model in a virtual environment. The simulation of pedestrians is a difficult challenge that is beginning to capture the attention of researchers and practitioners in evacuation simulation and urban planning. The field of computer graphics, in which virtual human animation has been an important research interest for decades, has contributed technologies fundamental to the computer-assisted visualization, including the automatic animation of pedestrian [1–3]. Until now, various simulation models and architectures have been developed. Whether the behaviors of the crowd are realistic or not has many relations with the methods used in constructing the crowd models. This paper focuses on one special crowd phenomenon which is significant in real world. The crowd which is composed of urgent members and normal ones has the following characteristics: the urgent members are the people who move forward along a fixed, given path for something urgent to do. The normal ones are the people who move in normal ways, just like most of the people in their daily lives. In real world, urgent people do not allow others to get in their way. Consequently, if the normal members are in a predefined range of the urgent ones, the latter will push them away. However, we must ensure that the urgent members are collision-free with the dynamic obstacles in the environment. Moreover, the interactions among the normal members are important issues to improve the authenticity of the crowd behaviors. For this special crowd, all of these propose demands for the studies of its behaviors.

Given the observations in real world, how is it possible that the urgent people can safely navigate through crowds and dynamic obstacles along a predefined path? The key insight is that the urgent people typically engage in moving forward just considering the dynamic obstacle: they push the normal members away from their trajectories to make room for their navigations.

In our work, we have been developing an autonomous, self-animating model of pedestrian crowd capable of performing a broad variety of natural activities in urban

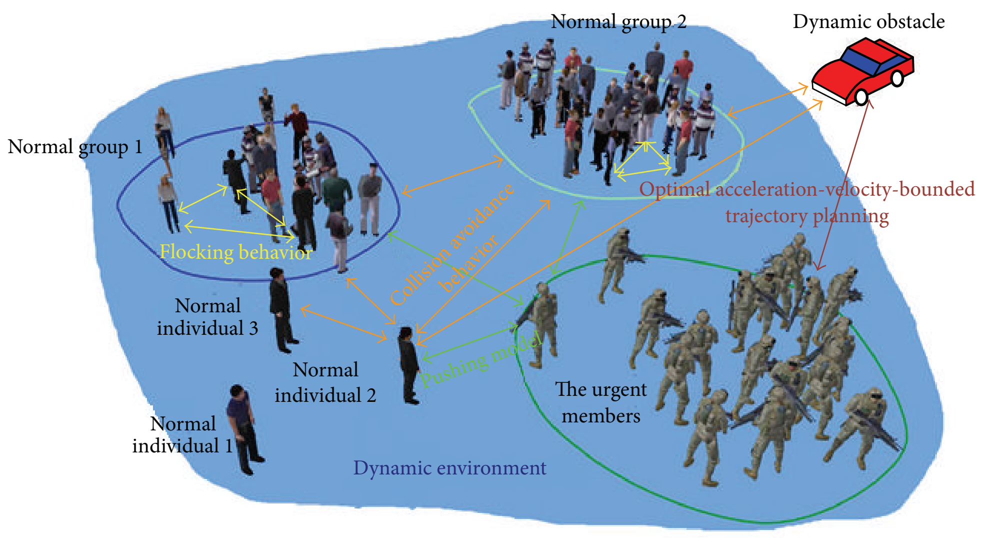


Figure 1: Components of the crowd behaviors.

dynamic environments. Because the acceleration and velocity of real person are bounded, we have adopted an optimal acceleration-velocity-bounded trajectory planning method in planning area and a comprehensive artificial life approach (e.g., collision avoidance method and flocking method) to addressing the problem of pedestrian behaviors. Our approach is inspired most heavily by the work of Johnson and Hauser [4] on optimal trajectory planning and by Foudil and Noureddine [5] on collision avoidance modeling in crowd simulation with priority rules. Also, Olfati-Saber's work on flocking model is used for reference [6].

In summary, our main contributions exist in three aspects.

(1) Construct the behaviors of the urgent members with an optimal acceleration-velocity-bounded trajectory planning method. This novel method ensures that the urgent members reach their destinations as soon as possible and are collision-free with the dynamic obstacles.

(2) Simulate the interactions between the urgent members and the normal ones. A novel pushing model is adopted to ensure that the normal members will not get in the way of the urgent ones. In this way, the normal members will not be treated as dynamic obstacles which promotes that the computational cost of the trajectory planning is acceptable for real time application.

(3) Imitate the interactions among the normal members. These interactions play a major role in synthesizing the animations of the crowd. The key insight is that the normal members typically engage in joint collision avoidance: they adapt their moving to each other to make room for navigation. So the collision avoidance behavior and the flocking behavior are exploited.

The above three aspects are combined together to generate realistic behaviors for our special crowd. The components of the crowd are represented in Figure 1.

As illustrated in Figure 1, the crowd behaviors consist of four parts: optimal trajectory planning, pushing model, collision avoidance behavior, and flocking behavior, each of which is responsible for dealing with different interactions. By designing and combining the respective behaviors of these four parts, realistic and complex behavior phenomena are achieved.

The organization of the rest of this paper is as follows. At first, Section 2 describes background and related works in trajectory planning, collision avoidance model, and flocking behavior. In Section 3, we give the detailed process of the optimal acceleration-velocity-bounded trajectory planning method and the pushing model. In Section 4, the collision avoidance model and flocking behavior for crowd simulation are presented briefly. After performing experiments with 3D virtual environments, simulation results and conclusions are stated at the end of this paper.

2. Related Work

In this paper, the behavior of the urgent members is a trajectory planning problem which has a vast difference with path planning. In respect of optimal trajectory planning problem with dynamic constraints in dynamic environments, Fraichard first addressed optimal trajectory planning in dynamic environments, that is, motion planning for robot subject to dynamic constraints and moving in a workspace with moving obstacles [7]. A novel concept of state-time space was introduced and a near-time-optimal trajectory was obtained by searching a set of canonical trajectories. The state-time space for planning in a dynamic environment is the Cartesian product of configuration space and time, that is, $C \times T$. Fraichard defined the canonical trajectories as the ones which had piecewise constant acceleration and changed its value at discrete times [7]. Fiorini and Shiller also presented a two-stage approach to plan a local time-optimal trajectory for a manipulator arm in the dynamic environments [8]. An optimization-based planning was used by Park et al. to handle the planning problem in arbitrary dynamic environments [9]. However, the dynamic constraints were ignored in their work [9]. Fiorini and Shiller presented a global optimal motion planning for Mars Rover which accounted for traversability and dynamic stability [10]. Recently, Maček et al. adopted rapidly exploring random trees and B-splines to develop a collision-free trajectory planning method in dynamic urban-like scenarios. This method can run automatically with the natural acceleration/deceleration phases according to the dynamic obstacles along the predefined path. The advantage of this method is that the vehicle velocities can be searched with a minimum time. Maček et al. have used kinodynamic planning techniques to explore the vehicle's action space [11]. While these above algorithms are able to find paths as well as velocity profiles, they have the defect of ignoring the acceleration constraint or obtaining near-optimal trajectories.

More recently, Johnson and Hauser have presented an optimal, exact, polynomial-time planner for optimal bounded-acceleration trajectories along a fixed, given path with dynamic obstacles [4]. In their method, the planner uses a velocity interval propagation technique to compute reachable velocity sets at obstacle vertices in the path-time space. In this process, the acceleration and velocity constraints are considered in path-velocity plane. This approach gives us great promotions in fulfillment of this paper to generate the optimal trajectories of urgent members. However, in this method the goal velocity is an interval rather than a fixed value. This is not the case in real life; for example, the goal velocity of the virtual human at the goal position is equal to zero. So the optimal trajectory planning in this paper is more complex and intractable.

In respect of collision avoidance behaviors for individuals and groups, techniques for simulating a crowd as a single entity have been proposed, as well as those which consider each person in the crowd separately. Collision avoidance is one of the crucial problems in crowd simulation. Without collision avoidance, crowd simulation does not look realistic. In crowd simulation, static and dynamic objects are the two types of obstacle that must concern us. The motionless objects can be seen as static obstacles, while dynamic objects refer to human, animals, and vehicles in the urban environments [12]. However, devising collision avoidance behavior among a number of individuals is more complex than designing the one between just two individuals. Morini et al. presented a short-term avoidance algorithm [13]. Aiming at the applications of densely populated urban environment, Tecchia and Chrysanthou proposed a simple and fast collision detection method. In their method, two different approaches for detecting and avoiding collision of the moving objects were used. Feurtey proposed an algorithm for collision avoidance which was planned from their current position to the goal based on the predicting and modifying trajectories. So in their method, the agents could predict their way by using the position and speed information of the obstacles [13]. Musse and Thalmann developed a multiresolution collision avoidance behavior for group interrelationship in multiresolution [14]. Foudil and Noureddine proposed a novel collision avoidance behavior for crowd simulation with priority rules. This method also provides us with great inspirations for our paper to generate various collision avoidance strategies for normal members [5].

In respect of flocking behavior for groups, many researchers used flocking algorithm to model flock, herd, and large crowds of people. Flocking is one of the pioneer techniques used to model a crowd of animals or humans. Flocking is basically composed of three rules: separation-steer away from neighbors and obstacle, cohesion-steer towards the centre of nearby entities, and alignment-steer in the average velocity and direction with the neighbors. In recent years, various distributed control algorithms have been proposed for the motion of multiple dynamic agents and the multiagent systems. For example, the flocking problem with one virtual leader was considered by Shi et al. [15]. Su et al. investigated the problem of controlling a group of autonomous agents to track multiple virtual leaders. They proved that the agents with the same virtual leader attained the same velocity, and the center and average velocity of the whole group converged to the weighted center and average velocity of all the leaders [16]. Olfati-Saber proposed a groundbreaking work on flocking for multiagent dynamic systems. He proposed three types of flocking algorithm. The first one is the basic flocking algorithm with only separation, cohesion, and alignment. Then one single leader agent was added to obtain the second algorithm. His third algorithm was obligated to deal with flocking problem in the presence of multiple obstacles. Also, a systematic method is provided for analyzing the split/rejoin maneuver of flocking [6].

All these above techniques can be seen as the lamp to light up our researches in generating realistic crowd behaviors of this paper.

3. Optimal Acceleration-Velocity-Bounded Trajectory Planning

3.1. Preliminaries. Suppose the virtual humans are simplified into cylinder with a unified radius. It is assumed that they move in the workspace $\mathscr{W} = R^2$. The configuration of a virtual human is uniquely defined by the triple $(x, y, \theta) \in C$, where $C = R^2 \times [0, 2\pi]$ denotes the configuration space. Suppose one urgent individual $\mathscr{A}$ is given a fixed parameterized geometric path S which is a continuous sequence of configurations, that is, a straight line or a curve in the configuration space. In this paper, we assume that S must be piecewise of class C^1 (a curve is of class C^n if it is differentiable n times and if its nth derivative is continuous). Because the urgent individual moves along S, we can reduce its configuration to a single variable s which satisfies $q(s) : [0, 1] \to C$. $s(t)$ is a time scaling function $s : [0, t_f] \to [0, 1]$ which assigns value s to each time $t \in [0, t_f]$. $s \cdot L_S$ represents the distance traveled along S in which L_S represents the total length of S. Moreover, the time scaling $s(t)$ should be twice-differentiable and monotonic ($\dot{s}(t) \geq 0$ for all $t \in [0, t_f]$).

The twice-differentiability of $s(t)$ ensures that the urgent individual's acceleration $\ddot{q}(t)$ is well defined and bounded [17].

The dynamic planning problem occurs in the path-velocity-time state space, that is, $s \times \dot{s} \times t$ space which is denoted as PVT [4]. So, the state of one urgent individual $x = (s, \dot{s}, t)$ consists of variable $s \in [0, 1]$, velocity $\dot{s}$, and time $t \in [0, t_{\max}]$. For a given $q(s)$, the following holds:

$$\begin{aligned} \dot{q} &= \frac{dq}{ds}\dot{s}, \\ \ddot{q} &= \frac{d^2q}{ds^2}\dot{s}^2 + \frac{dq}{ds}\ddot{s}. \end{aligned} \tag{1}$$

The dynamic constraints of the urgent individuals must satisfy velocity and acceleration bounds:

$$\begin{gathered} 0 \le \sqrt{\dot{q}_x^2 + \dot{q}_y^2} \le v_{\max}, \quad \dot{q}_x \ge 0,\ \dot{q}_y \ge 0, \\ 0 \le \sqrt{\ddot{q}_x^2 + \ddot{q}_y^2} \le a_{\max}. \end{gathered} \tag{2}$$

Substituting (2) into (1), we get the following dynamic constraints on variable s:

$$\dot{s} \in \left[\dot{s}_{\min}\left(s, a_{\max}, v_{\max}\right), \dot{s}_{\max}\left(s, a_{\max}, v_{\max}\right)\right], \tag{3}$$

$$\ddot{s} \in \left[\ddot{s}_{\min}\left(s, \dot{s}, a_{\max}, v_{\max}\right), \ddot{s}_{\max}\left(s, \dot{s}, a_{\max}, v_{\max}\right)\right] \tag{4}$$

with $\dot{s}_{\min}(s, a_{\max}, v_{\max}) \ge 0$ and $\ddot{s}_{\min}(s, \dot{s}, a_{\max}, v_{\max}) < 0 < \ddot{s}_{\max}(s, \dot{s}, a_{\max}, v_{\max})$. A trajectory $x(t) = (s(t), \dot{s}(t), t)$ defined over $t \in [t_0, t_f]$ in state space is dynamically feasible if and only if (3) and (4) are satisfied at the same time. Then, we say that a goal state x_f is dynamically reachable from initial state x_0 if a dynamically feasible trajectory $x(t)$ over interval $[t_0, t_f]$ such that $x(t_0) = x_0$ and $x(t_f) = x_f$ is existed. Note that there is a one-to-one correspondence between the trajectory $x(t)$ and its time scaling function $s(t)$.

Let O_i $i \in \{1, \ldots, m\}$ be the set of moving obstacles. Let $O_i(t)$ denote the region of $\mathscr{W}$ occupied by O_i at time t and $\mathscr{A}(q(s(t)))$ the region of $\mathscr{W}$ occupied by $\mathscr{A}$ at position $q(s(t))$ along S. Then the obstacle region in state space is defined as

$$X_{\text{obs}} = \left\{(s, \dot{s}, t) \mid \exists i \in \{1, \ldots, m\}, \mathscr{A}\left(q\left(s\left(t\right)\right)\right) \cap O_i\left(t\right) \ne \emptyset\right\}. \tag{5}$$

Obviously, the X_{obs} corresponds to $(s(t), t)$ points in $s \times t$ space. The trajectory $x(t)$ is collision-free if $(s(t), t) \notin X_{\text{obs}}$ for $t \in [t_0, t_f]$.

Above all, we can formally state the problem which is to be solved. Let $(s_0, \dot{s}_0, 0)$ be the start state of $\mathscr{A}$ and $(s_f, \dot{s}_f, t_f)$ its goal state.

A trajectory $\Gamma : [0, 1] \rightarrow$ PVT is a solution to the problem if and only if

(1) $\Gamma(0) = (s_0, \dot{s}_0, 0)$, $\Gamma(1) = (s_f, \dot{s}_f, t_f)$, and $\dot{s}_0, \dot{s}_f \ge 0$,

(2) Γ is dynamically feasible and collision-free for $t \in [0, t_f]$ and $t_f \le t_{\max}$. $t_{\max}$ is the maximal time given for the planner,

(3) $\dot{s}_f$ is a specific value (e.g., $\dot{s}_f = 0$) rather than a velocity interval.

We are interested in finding an exact time-optimal trajectory, that is, a trajectory such that t_f should be minimal. Moreover, $\dot{s}_f$ is an interval in Johnson's method rather than a specific value. So the optimal trajectory planning in this paper is more complex and intractable.

3.2. Optimal Acceleration-Velocity-Bounded Trajectory Planning. The method that we have developed in order to solve the above problem is initially motivated by the work described in [4]. We improve Johnson's method in order to find out the exact time-optimal trajectory between an initial and a specific goal state. The flowchart of our method is illustrated in Figure 2.

Next we will describe every part of Figure 2 in detail.

3.2.1. Compute Reachable Velocity Sets

(I) Compute Reachable Velocity Sets from One Specified Initial State. The goal of this section is to facilitate the computing reachable velocity intervals from one specified initial state. Without loss of generality, we suppose the specified initial state is $x_I = (s_I, \dot{s}_I, t_I)$ which can be any state in $\{(s_0(t_0), \dot{s}_0(t_0), t_0), \ldots, (s_{2m+1}(t_{2m+1}), \dot{s}_{2m+1}(t_{2m+1}), t_{2m+1})\}$. The dynamically reachable set of velocities $R(t; x_I)$ from x_I at any point in time $t \ge t_I$ is the set of velocities attainable at t from x_I via dynamically feasible trajectories. It can be computed in $s \times \dot{s}$ space. In order to simplify the computation, we assume that S is a straight line. Then (3) and (4) can be simplified into

$$\dot{s} \in \left[0, \frac{v_{\max}}{L_S}\right],$$

$$\text{that is, } \dot{s}_{\min}\left(s, a_{\max}, v_{\max}\right) = 0; \tag{6}$$

$$\dot{s}_{\max}\left(s, a_{\max}, v_{\max}\right) = \frac{v_{\max}}{L_S},$$

$$\ddot{s} \in \left[-\frac{a_{\max}}{L_S}, \frac{a_{\max}}{L_S}\right],$$

$$\text{that is, } \ddot{s}_{\min}\left(s, \dot{s}, a_{\max}, v_{\max}\right) = -\frac{a_{\max}}{L_S}; \tag{7}$$

$$\ddot{s}_{\max}\left(s, \dot{s}, a_{\max}, v_{\max}\right) = \frac{a_{\max}}{L_S}.$$

Johnson showed that $R(t; x_I)$ is bounded by at most six curves [4] which are denoted in Figure 3 in our paper.

In Figure 3, parabolic segment (a) corresponds to the process in which $\mathscr{A}$ firstly takes the minimal acceleration for time t_s and then switches to maximal acceleration for time $t - t_s$. It must be noted that the terminal velocity of the process of deceleration and the one of the process of acceleration must

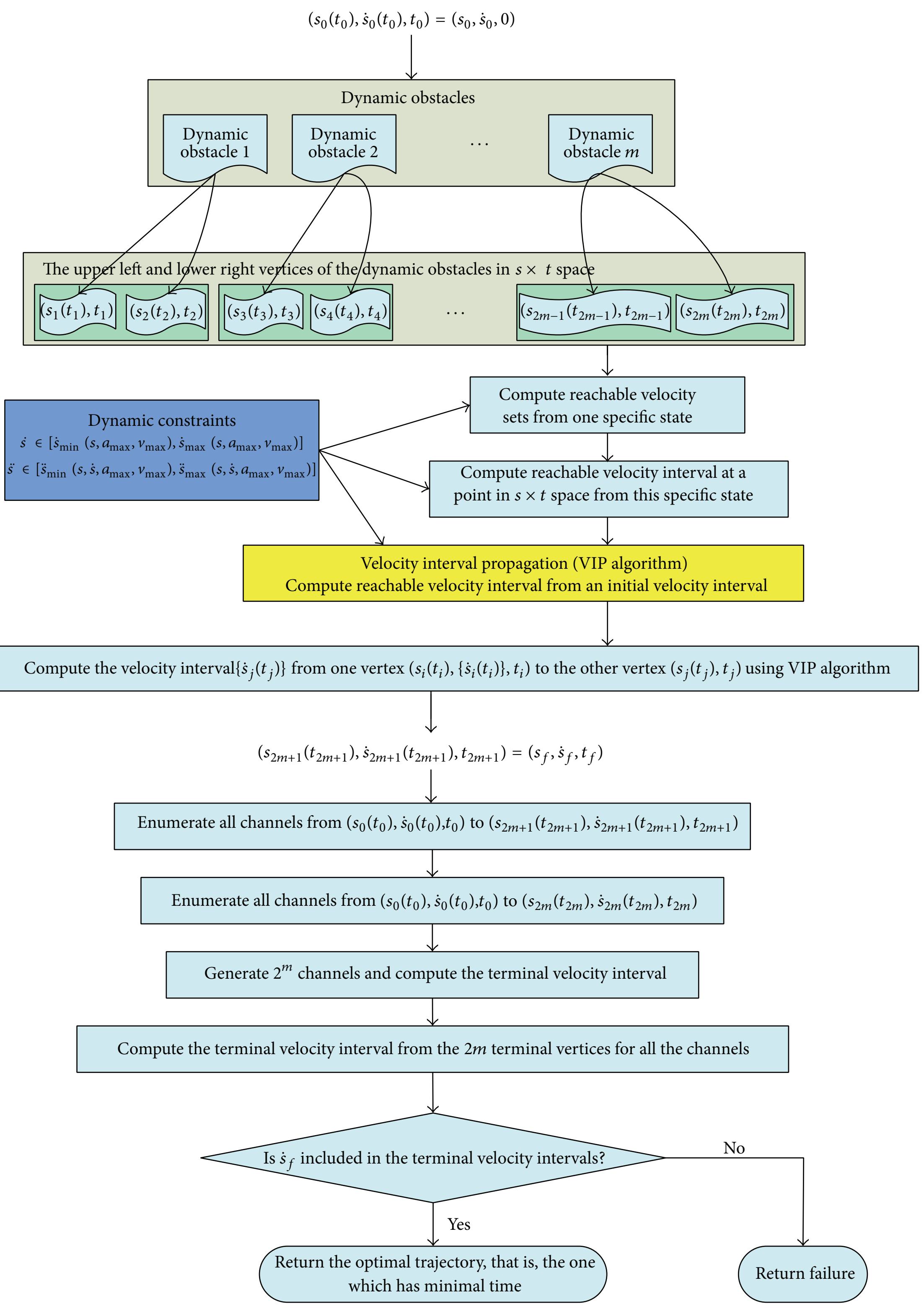

$(s_0(t_0), \dot{s}_0(t_0), t_0) = (s_0, \dot{s}_0, 0)$
Dynamic obstacles
Dynamic obstacle 1
Dynamic obstacle 2
...
Dynamic obstacle m
The upper left and lower right vertices of the dynamic obstacles in $s \times t$ space
$(s_1(t_1), t_1)$
$(s_2(t_2), t_2)$
$(s_3(t_3), t_3)$
$(s_4(t_4), t_4)$
...
$(s_{2m-1}(t_{2m-1}), t_{2m-1})$
$(s_{2m}(t_{2m}), t_{2m})$
Dynamic constraints
$\dot{s} \in [\dot{s}_{\min}(s, a_{\max}, v_{\max}), \dot{s}_{\max}(s, a_{\max}, v_{\max})]$
$\ddot{s} \in [\ddot{s}_{\min}(s, \dot{s}, a_{\max}, v_{\max}), \ddot{s}_{\max}(s, \dot{s}, a_{\max}, v_{\max})]$
Compute reachable velocity sets from one specific state
Compute reachable velocity interval at a point in $s \times t$ space from this specific state
Velocity interval propagation (VIP algorithm)
Compute reachable velocity interval from an initial velocity interval
Compute the velocity interval $\{\dot{s}_j(t_j)\}$ from one vertex $(s_i(t_i), \{\dot{s}_i(t_i)\}, t_i)$ to the other vertex $(s_j(t_j), t_j)$ using VIP algorithm
$(s_{2m+1}(t_{2m+1}), \dot{s}_{2m+1}(t_{2m+1}), t_{2m+1}) = (s_f, \dot{s}_f, t_f)$
Enumerate all channels from $(s_0(t_0), \dot{s}_0(t_0), t_0)$ to $(s_{2m+1}(t_{2m+1}), \dot{s}_{2m+1}(t_{2m+1}), t_{2m+1})$
Enumerate all channels from $(s_0(t_0), \dot{s}_0(t_0), t_0)$ to $(s_{2m}(t_{2m}), \dot{s}_{2m}(t_{2m}), t_{2m})$
Generate 2^m channels and compute the terminal velocity interval
Compute the terminal velocity interval from the $2m$ terminal vertices for all the channels
Is $\dot{s}_f$ included in the terminal velocity intervals?
No
Yes
Return the optimal trajectory, that is, the one which has minimal time
Return failure

Figure 2: Flowchart of optimal trajectory planning.

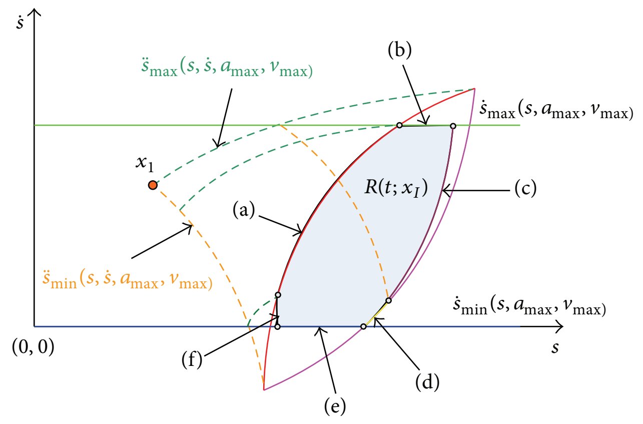

$\dot{s}$
$\ddot{s}_{\max}(s,\dot{s},a_{\max},v_{\max})$
(b)
$\dot{s}_{\max}(s,a_{\max},v_{\max})$
x_1
$R(t;x_I)$
(c)
(a)
$\ddot{s}_{\min}(s,\dot{s},a_{\max},v_{\max})$
$\dot{s}_{\min}(s,a_{\max},v_{\max})$
(0, 0)
s
(f)
(d)
(e)

Figure 3: Reachable velocity sets from one initial state.

satisfy formulation (6). The states on (a) can be formulated as in the following equation:

$$\begin{gathered}x_{t_I+t}=\Big(s_I+\dot{s}_It_s+\frac{1}{2}\ddot{s}_{\min}t_s^2+(\dot{s}_I+\ddot{s}_{\min}t_s)(t-t_s)\\+\frac{1}{2}\ddot{s}_{\max}(t-t_s)^2,\dot{s}_I+\ddot{s}_{\min}t_s+\ddot{s}_{\max}(t-t_s),\\t_I+t\Big),\\\dot{s}_I+\ddot{s}_{\min}t_s\geq\dot{s}_{\min},\qquad\dot{s}_I+\ddot{s}_{\min}t_s+\ddot{s}_{\max}(t-t_s)\leq\dot{s}_{\max}.\end{gathered}\tag{8}$$

Similarly, parabolic segment (d) corresponds to the process, takes the maximal acceleration for time t_s, and then switches to minimal acceleration for time $t-t_s$. The states on (d) can be formulated as

$$\begin{gathered}x_{t_I+t}=\Big(s_I+\dot{s}_It_s+\frac{1}{2}\ddot{s}_{\max}t_s^2+(\dot{s}_I+\ddot{s}_{\max}t_s)(t-t_s)\\+\frac{1}{2}\ddot{s}_{\min}(t-t_s)^2,\dot{s}_I+\ddot{s}_{\max}t_s+\ddot{s}_{\min}(t-t_s),\\t_I+t\Big),\\\dot{s}_I+\ddot{s}_{\max}t_s\leq\dot{s}_{\max},\qquad\dot{s}_I+\ddot{s}_{\max}t_s+\ddot{s}_{\min}(t-t_s)\geq\dot{s}_{\min}.\end{gathered}\tag{9}$$

Parabolic segment (c) corresponds to the process, takes the maximal acceleration until reaching $\dot{s}_{\max}(s,a_{\max},v_{\max})$ at the time t_v, progresses with $\dot{s}_{\max}(s,a_{\max},v_{\max})$ until time t_s elapse, and then decelerates for time $t-t_s$. The states on (c) can be formulated as

$$\begin{gathered}x_{t_I+t}=\Big(s_I+\dot{s}_It_v+\frac{1}{2}\ddot{s}_{\max}t_v^2+\dot{s}_{\max}(t_s-t_v)+\dot{s}_{\max}(t-t_s)\\+\frac{1}{2}\ddot{s}_{\min}(t-t_s)^2,\dot{s}_{\max}+\ddot{s}_{\min}(t-t_s),t_I+t\Big),\\t_v=\left(\frac{\dot{s}_{\max}-\dot{s}_I}{\ddot{s}_{\max}}\right);\qquad\dot{s}_I+\ddot{s}_{\max}t_s\geq\dot{s}_{\max}.\end{gathered}\tag{10}$$

Similarly, parabolic segment (f) corresponds to the process, takes the minimal acceleration until reaching $\dot{s}_{\min}(s,a_{\max},v_{\max})$, progresses with $\dot{s}_{\min}(s,a_{\max},v_{\max})$ until time t_s elapse, and then accelerates for time $t-t_s$. The states on (f) can be formulated as

$$\begin{gathered}x_{t_I+t}=\Big(s_I+\dot{s}_It_v+\frac{1}{2}\ddot{s}_{\min}t_v^2+\dot{s}_{\min}(t_s-t_v)+\dot{s}_{\min}(t-t_s)\\+\frac{1}{2}\ddot{s}_{\max}(t-t_s)^2,\dot{s}_{\min}+\ddot{s}_{\max}(t-t_s),t_I+t\Big),\\t_v=\left(\frac{\dot{s}_{\min}-\dot{s}_I}{\ddot{s}_{\min}}\right);\qquad\dot{s}_I+\ddot{s}_{\min}t_s\leq\dot{s}_{\min}.\end{gathered}\tag{11}$$

Line segment (b) corresponds to the process, takes the minimal acceleration for time t_s, and then switches to maximal acceleration until reaching $\dot{s}_{\max}(s,a_{\max},v_{\max})$, finally progressing with $\dot{s}_{\max}(s,a_{\max},v_{\max})$ until time t elapse. The states on (b) can be formulated as

$$\begin{gathered}x_{t_I+t}=\Big(s_I+\dot{s}_It_s+\frac{1}{2}\ddot{s}_{\min}t_s^2+(\dot{s}_I+\ddot{s}_{\min}t_s)t_v\\+\frac{1}{2}\ddot{s}_{\max}t_v^2+\dot{s}_{\max}(t-t_s-t_v),\dot{s}_{\max},t_I+t\Big),\\t_v=\left(\frac{\dot{s}_{\max}-\dot{s}_I-\ddot{s}_{\min}t_s}{\ddot{s}_{\max}}\right);\qquad\dot{s}_I+\ddot{s}_{\min}t_s\geq\dot{s}_{\min};\\\dot{s}_I+\ddot{s}_{\min}t_s+\ddot{s}_{\max}t_v\geq\dot{s}_{\max}.\end{gathered}\tag{12}$$

Similarly, line segment (e) corresponds to the process, takes the maximal acceleration for time t_s, and then switches to minimal acceleration until reaching $\dot{s}_{\min}(s,a_{\max},v_{\max})$, finally progressing with $\dot{s}_{\min}(s,a_{\max},v_{\max})$ until time t elapse. The states on (e) can be formulated as

$$\begin{gathered}x_{t_I+t}=\Big(s_I+\dot{s}_It_s+\frac{1}{2}\ddot{s}_{\max}t_s^2+(\dot{s}_I+\ddot{s}_{\max}t_s)t_v\\+\frac{1}{2}\ddot{s}_{\min}t_v^2+\dot{s}_{\min}(t-t_s-t_v),\dot{s}_{\min},t_I+t\Big),\\t_v=\left(\frac{\dot{s}_{\min}-\dot{s}_I-\ddot{s}_{\max}t_s}{\ddot{s}_{\min}}\right);\qquad\dot{s}_I+\ddot{s}_{\max}t_s\leq\dot{s}_{\max};\\\dot{s}_I+\ddot{s}_{\max}t_s+\ddot{s}_{\min}t_v\leq\dot{s}_{\min}.\end{gathered}\tag{13}$$

All these six curves enclose the reachable velocity set $R(t;x_I)$ from one single initial state $x_I=(s_I,\dot{s}_I,t_I)$ which is graphically showed as the shadow region in Figure 3.

(II) Compute Reachable Velocity Interval at a Point in $s\times t$ Space from $x_I=(s_I,\dot{s}_I,t_I)$. After obtaining the reachable velocity set in $s\times\dot{s}$ space, we can then compute the reachable velocity interval at a point $(s_G(t_G),t_G)$ in $s\times t$ space from the initial state $x_I=(s_I(t_I),\dot{s}_I,t_I)$. We use $\{\dot{s}_G(t_G)\}$ to represent the reachable velocity interval at $(s_G(t_G),t_G)$. We use $R(t_G-t_I;x_I)$ to compute $\{\dot{s}_G(t_G)\}$ in the following formulation:

$$\{\dot{s}_G(t_G)\}=\left[\underline{\dot{s}}_G,\bar{\dot{s}}_G\right]=R(t_G-t_I;x_I)\cap\{(s,\dot{s})\mid s=s_G\}\tag{14}$$

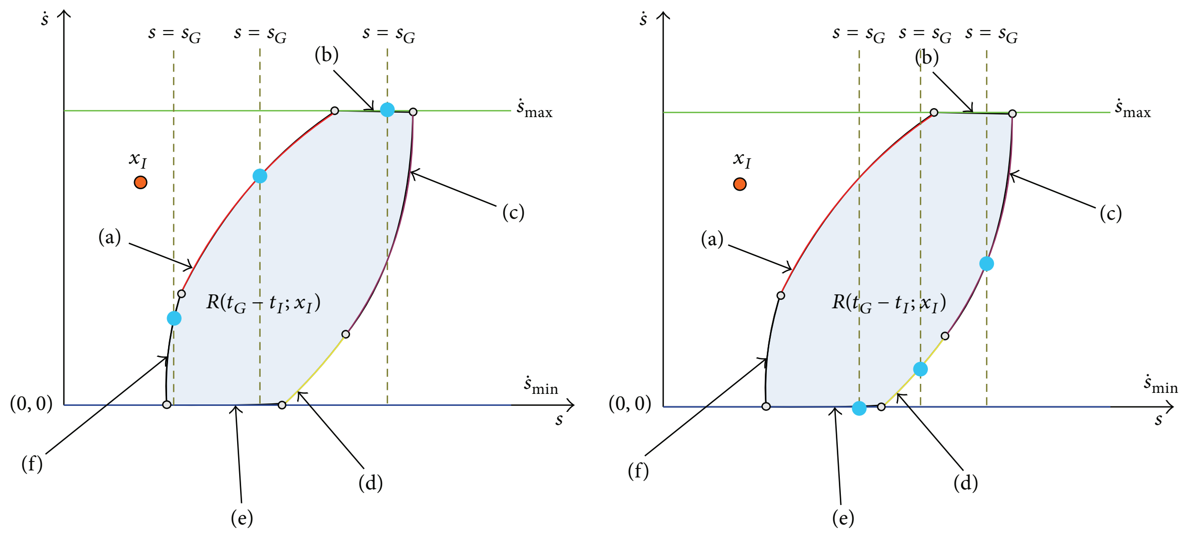


Figure 4: Reachable velocity sets from one initial state.

in which $\bar{\dot{s}}_G$ and $\underline{\dot{s}}_G$ represent the upper and lower bounds of velocity interval $\{\dot{s}_G(t_G)\}$, respectively. $\bar{\dot{s}}_G$ is defined as the intersection of a boundary curve of $R(t_G - t_I; x_I)$ with $s = s_G$.

We only need to consider three boundary cures in computing $\bar{\dot{s}}_G$ (curves (a), (b), and (f)) and $\underline{\dot{s}}_G$ (curves (c), (d), and (e)), as illustrated in Figure 4.

In this process, we determine the switching time t_s by substituting $s = s_G$ into (8), (11), and (12) for $\bar{\dot{s}}_G$ and (9), (10), and (13) for $\underline{\dot{s}}_G$, respectively, and solving the resulting quadratic equation. At the same time, the constraints in (8)~(13) are validated to determine if some trajectories exist from $x_I = (s_I, \dot{s}_I, t_I)$ to $(s_G(t_G), \{\dot{s}_G(t_G)\}, t_G)$.

(III) VIP Algorithm Computing Reachable Velocity Interval from an Initial Velocity Interval. After computing the reachable velocity interval from an initial specified velocity, we can extend it to compute the reachable velocity interval from an initial velocity interval which forms the backbone of optimal trajectory planner. It takes $(s_G(t_G), \{\dot{s}_G(t_G)\}, t_G)$ and a target point $(s_H(t_H), t_H)$ in $s \times t$ space as input. It outputs a range of target velocities $\{\dot{s}_H(t_H)\} = [\underline{\dot{s}}_H, \bar{\dot{s}}_H]$ reachable from $(s_G(t_G), \{\dot{s}_G(t_G)\}, t_G)$. $\{\dot{s}_H(t_H)\} = [\underline{\dot{s}}_H, \bar{\dot{s}}_H]$ is computed in the following steps.

Step 1. Compute the reachable velocity interval $V_1 = R(t_H - t_G; x_G) \cap \{(s, \dot{s}) \mid s = s_H\}$ for initial velocity $\dot{s}_G = \bar{\dot{s}}_G$ from $(s_G(t_G), \bar{\dot{s}}_G, t_G)$.

Step 2. Compute the reachable velocity interval $V_2 = R(t_H - t_G; x_G) \cap \{(s, \dot{s}) \mid s = s_H\}$ for initial velocity $\dot{s}_G = \underline{\dot{s}}_G$ from $(s_G(t_G), \underline{\dot{s}}_G, t_G)$.

Step 3. The goal of this step is maximizing the terminal velocity at $(s_H(t_H), t_H)$. It does so by constructing a parabolic trajectory with acceleration $\ddot{s}_{\max}(s, \dot{s}, a_{\max}, v_{\max})$ that interpolates $(s_G(t_G), t_G)$ and $(s_H(t_H), t_H)$. If the initial velocity $\dot{s}_{G,\max}^{\text{interpolate}}$ of this interpolating parabolic trajectory belongs to $\{\dot{s}_G(t_G)\}$, then $V_3 = \{\dot{s}_{G,\max}^{\text{interpolate}}\}$.

Step 4. The goal of this step is minimizing the terminal velocity at $(s_H(t_H), t_H)$. It also does so by constructing a parabolic trajectory with acceleration $\ddot{s}_{\min}(s, \dot{s}, a_{\max}, v_{\max})$ that interpolates $(s_G(t_G), t_G)$ and $(s_H(t_H), t_H)$. If the initial velocity $\dot{s}_{G,\min}^{\text{interpolate}}$ of this interpolating parabolic trajectory belongs to $\{\dot{s}_G(t_G)\}$, then $V_4 = \{\dot{s}_{G,\min}^{\text{interpolate}}\}$.

Step 5. Finally, the output velocity interval $\{\dot{s}_H(t_H)\} = [\underline{\dot{s}}_H, \bar{\dot{s}}_H] = V_1 \cup V_2 \cup V_3 \cup V_4$. Note that either V_1, V_2, V_3, or V_4 may be empty.

Because the goal of the method in this section is propagating the velocity intervals, from one velocity interval to the other as long as we know the initial and terminal point in $s \times t$ space, we call this method velocity interval propagation (VIP algorithm). Steps 1~4 can be represented as $\{\dot{s}_H(t_H)\} = \mathrm{VIP}((s_G(t_G), \{\dot{s}_G(t_G)\}, t_G), (s_H(t_H), t_H))$ in short.

3.2.2. Computing the Optimal Trajectory.

3.2.2. Computing the Optimal Trajectory. Suppose there are m dynamic obstacles in the environment; then only $2m$ vertices are important to our optimal planning which correspond to the upper left and lower right vertex of the obstacles [4]. Let $(s_1(t_1), t_1), \ldots, (s_{2m}(t_{2m}), t_{2m})$ be the sequence of upper left and lower right obstacle vertices such that $t_0 \le t_i \le t_f$ for all $i = 1, \ldots, 2m$. Then we sort these $2m$ vertices by increasing t coordinate and denote $(s_0(t_0), \dot{s}_0(t_0), t_0) = (s_0, \dot{s}_0, 0)$ and $(s_{2m+1}(t_{2m+1}), \dot{s}_{2m+1}(t_{2m+1}), t_{2m+1}) = (s_f, \dot{s}_f, t_f)$. Our planner consists of four stages.

In the first stage, using the VIP algorithm, we can get the velocity interval from any vertex of one dynamic obstacle to any vertex of another one for $(s_0(t_0), t_0), \ldots, (s_{2m}(t_{2m}), t_{2m})$; that is, we carry out $\{\dot{s}_j(t_j)\} = \mathrm{VIP}((s_i(t_i), \{\dot{s}_i(t_i)\}, t_i), (s_j(t_j), t_j))$ for all $i, j \in [1, 2m]; i \neq j$.

In the second stage, we get 2^m channels and 2^m terminal velocity intervals of these channels. These terminal velocity intervals can be merged into $2m$ velocity intervals $\{\{\dot{s}_{1,\text{union}}(t_1)\},\ldots,\{\dot{s}_{2m,\text{union}}(t_{2m})\}\}$ corresponding to the $2m$ vertices with union operation.

In the third stage, we use VIP algorithm to connect $(s_{2m}(t_{2m}),t_{2m})$ with $(s_{2m+1}(t_{2m+1}),t_{2m+1})$ by taking these $2m$ velocity intervals as input to determine the terminal velocity interval at given terminal time. We use discrete $t_{i,f}^k$ to represent the given terminal time:

$$t_{i,f}^k = t_i + k\cdot\Delta t \quad k = 1,\ldots,\left\lfloor\frac{t_{\max}-t_i}{\Delta t}\right\rfloor. \tag{15}$$

Then we can get the terminal velocity interval at every given terminal time $t_{i,f}^k$ by calling $\{\dot{s}_k(t_{i,f}^k)\} = \text{VIP}((s_i(t_i), \{\dot{s}_{i,\text{union}}(t_i)\}, t_i), (s_k(t_{i,f}^k), t_{i,f}^k))$ $i = 1,\ldots,2m$; $k = 1,\ldots,\lfloor(t_{\max}-t_i)/\Delta t\rfloor$.

In the fourth stage, we judge if $\dot{s}_f$ is in these $\sum_{i=1}^{2m}\lfloor(t_{\max}-t_i)/\Delta t\rfloor$ terminal velocity intervals or not. If true, we return the trajectory with minimal terminal time $t_{i,f}^k$ which is the optimal trajectory and its final velocity equals $\dot{s}_f$. If not, the planner returns failure. It means that no trajectory exists between the initial state and the goal state of $\mathscr{A}$ which satisfies the dynamic constraints and is collision-free with dynamic obstacles.

Complexity Analysis. The computational cost of the planner mainly comes from the first and the third stage. In the first stage, we must call the VIP algorithm for 2^m times. In the third stage, VIP algorithm is called $\sum_{i=1}^{2m}\lfloor(t_{\max}-t_i)/\Delta t\rfloor$ times. Then the complexity of our planner is $O(2^m+\sum_{i=1}^{2m}\lfloor(t_{\max}-t_i)/\Delta t\rfloor) < O(2^m+2m\lfloor t_{\max}/\Delta t\rfloor)$ which is equivalent to $O(2^m)$. This computational cost is vast for big m values. However, there are only several dynamic obstacles in our environment and this is why we apply pushing behavior to deal with the interaction between the urgent members and the normal members.

3.3. Pushing Model. Pushing model is used to model the interactions between the urgent members and the normal ones; that is, the normal members will be pushed away when they are in a predefined range of the urgent ones. We use the four following rules to model pushing behavior between any normal individual i and urgent one j:

$$\begin{array}{ll} \text{If } d_{ij} = 0 & \text{Then } V_{\overrightarrow{ij}} = -V_{\max} \\ \text{If } 0 < d_{ij} < \mu & \text{Then } V_{\overrightarrow{ij}} = -M\cdot V_{\max}\cdot e^{-d_{ij}} + V_0 \\ \text{If } \left|V_{\overrightarrow{ij}}\right| > \sqrt{V_{\max}^2-\left(V_{\overrightarrow{ij}}^{\perp}\right)^2} & \text{Then } V_{\overrightarrow{ij}} = -\sqrt{V_{\max}^2-\left(V_{\overrightarrow{ij}}^{\perp}\right)^2} \\ \text{If } d_{ij} \ge \mu & \text{Then } V_{\overrightarrow{ij}} = V_0 \end{array} \tag{16}$$

in which d_{ij} denotes the distance between i and j. μ represents the threshold of the interaction distance. $V_{\overrightarrow{ij}}$ represents the velocity of i at the direction of vector $\overrightarrow{ij}$. $V_{\overrightarrow{ij}}^{\perp}$ represents the velocity of i at the perpendicular direction of vector $\overrightarrow{ij}$. $V_{\max}$ indicates the maximal velocity at which a normal individual moves. And V_0 represents the velocity of i before the interaction between i and j happens.

This novel pushing model ensures that the urgent members will never collide with the normal ones. So we do not need to consider the normal members in the process of optimal trajectory planning whose cost time will greatly reduce owing to only the dynamic obstacles being considered.

4. Collision Avoidance Behaviors and Flocking Behaviors

4.1. Collision Avoidance Behaviors. In this section, we present the collision avoidance behaviors on the basis of Foudil and Noureddine's work. As a general approach, this method can be combined with different crowd and multiagent simulation algorithms. In each time step of the simulation, we need to predict if every agent will collide with other agents in the crowd. Then we must determine the type of collision which may happen. In real life, there are three possible types of collision [5]. The first type is *toward collision behavior*. It happens when two agents are moving head-on toward each other. The second one is *away collision*. It happens when one agent who is behind the other one is moving with a bigger velocity and whose moving direction is consistent with the line from the back agent to the front one. The third collision behavior is *glancing collision* which happens when two agents are walking in roughly the same direction.

These three types of collision behaviors are shown in Figure 5.

We use a series of rules to realize these three collision avoidance behaviors separately. In *toward collision behavior*, we use the following strategy to avoid the forthcoming collision: the agent who has a low priority can select waiting or changing its moving direction and the agent who has an upper priority selects keeping its moving on with unchanged direction and velocity. In respect to *away collision*, the back agent can select slowing its velocity or changing its moving direction to any side. We deal with *glancing collision* with the same manner as the *toward collision behavior.*

Particularly, when these three collision avoidance behaviors conflict with each other, we resolve this problem by predefining the priority order of them. In this paper, the priority order of these three types of collision avoidance behaviors is toward collision, away collision, and glancing collision from high to low.

4.2. Flocking Behaviors. Flocking behaviors are used to model the collective gathering behaviors of the normal groups. Aiming at the agents group consisting of one leader agent and N follower ones, we use a distributed control method to simulate the flocking behavior [15, 16]. The motion of each virtual agent is described by two integrators as

$$\begin{aligned} \dot{\mathbf{p}}_i(t) &= \mathbf{v}_i(t) \\ \dot{\mathbf{v}}_i(t) &= \mathbf{a}_i(t) \end{aligned} \quad i = 1,2,\ldots,N, \tag{17}$$

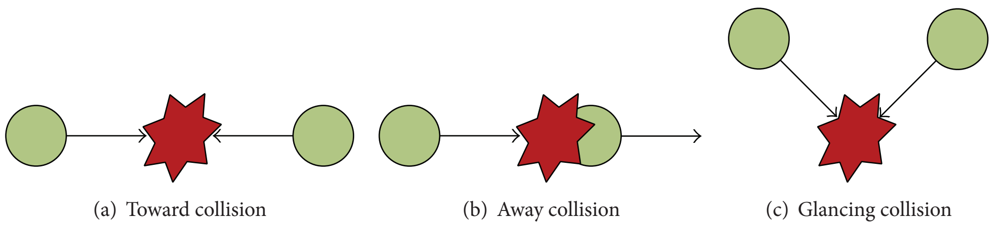


Figure 5: Three types of the collision behaviors.

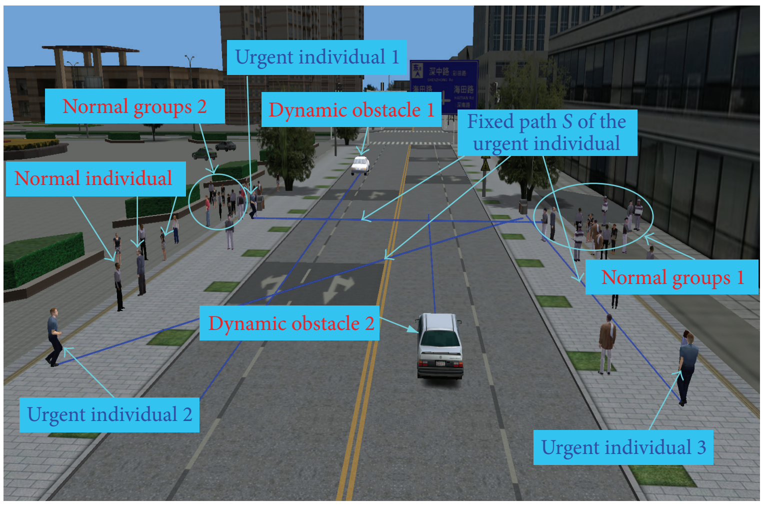


Figure 6: The dynamic environment of the experiments.

where $\mathbf{p}_i$, $\mathbf{v}_i$, and $\mathbf{a}_i$ are denoted as the position, velocity, and acceleration of agent i, respectively. Similarly, the virtual leader has the following dynamics of motion:

$$\begin{aligned}\dot{\mathbf{p}}_\gamma(t) &= \mathbf{v}_\gamma(t), \\ \dot{\mathbf{v}}_\gamma(t) &= \mathbf{a}_\gamma(t),\end{aligned} \tag{18}$$

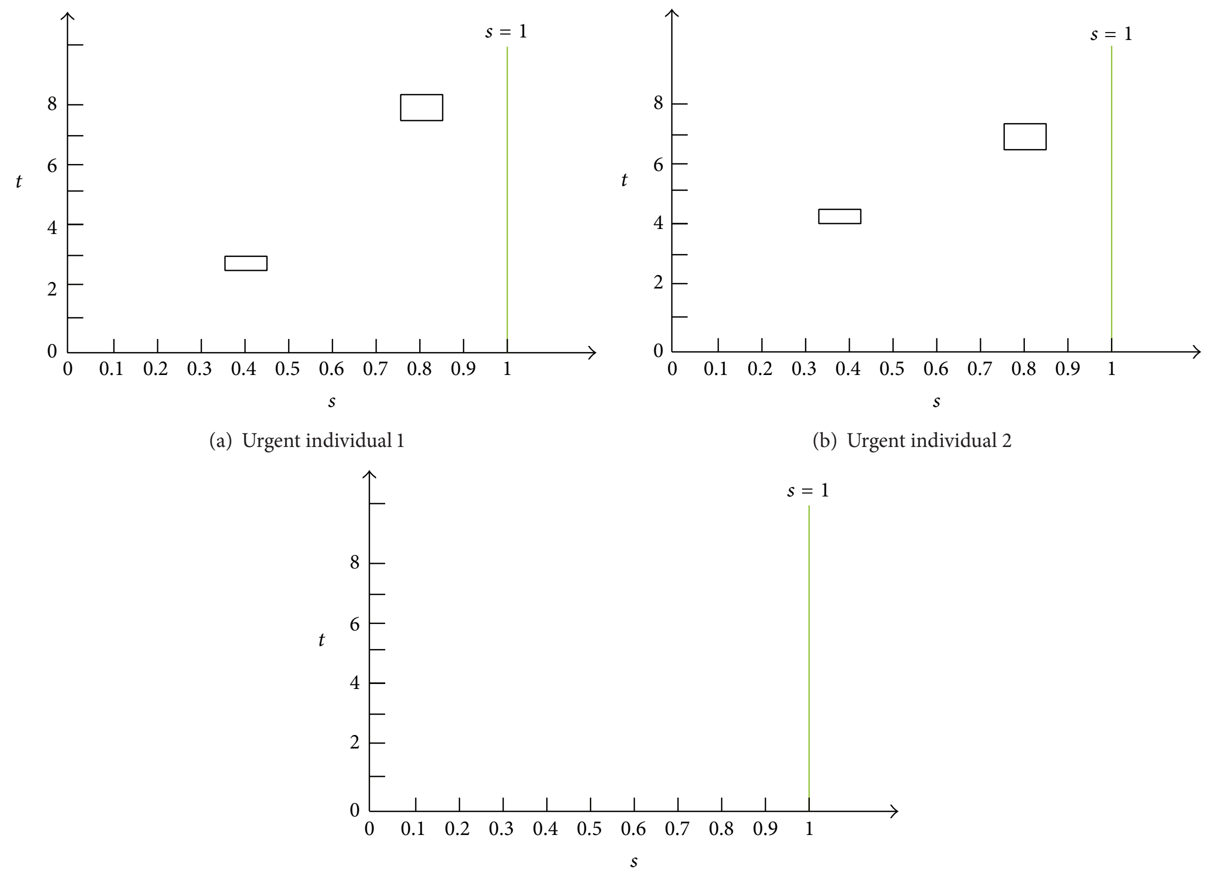
where $\mathbf{p}_\gamma$, $\mathbf{v}_\gamma$, and $\mathbf{a}_\gamma$ represent the position, velocity, and acceleration of leader agent, respectively. The control algorithm of the flocking method is given as

$$\begin{aligned}\mathbf{a}_i = &-\sum_{j\in\mathcal{N}_i(t)} \nabla_{\mathbf{p}_i}\psi_\alpha\left(\left\|\mathbf{p}_i-\mathbf{p}_j\right\|_\sigma\right) + \sum_{j\in\mathcal{N}_i(t)} a_{ij}(t)\left(\mathbf{v}_j-\mathbf{v}_i\right) \\ &+ \mathbf{a}_\gamma^i - c_1\left(\mathbf{p}_i-\mathbf{p}_\gamma^i\right) - c_2\left(\mathbf{v}_i-\mathbf{v}_\gamma^i\right) \\ &\qquad i = 1,\ldots,N \quad c_1, c_2 > 0,\end{aligned} \tag{19}$$

where $\mathcal{N}_i(t) = \{j : \|\mathbf{p}_i-\mathbf{p}_j\| \le R,\ j = 1,2,\ldots,N, j \ne i\}$ is the set of spatial neighbors of agent i. R represents the interaction range. $\|\cdot\|_\sigma$ is the Euclidean norm and $\|\mathbf{z}\|_\sigma = (1/\varepsilon)[\sqrt{1+\varepsilon\|\mathbf{z}\|^2}-1]\ \varepsilon > 0$ for a vector $\mathbf{z}$. ψ_α is the artificial nonnegative smooth pairwise potential function whose characteristics depend on the relative distances between agent i and its neighbors: ψ_α reaches its maximal value as $\|\mathbf{p}_i-\mathbf{p}_j\|_\sigma \to 0$, and ψ_α acquires its unique minimal value at a predefined distance $\|d\|_\sigma$. When $\|\mathbf{p}_i-\mathbf{p}_j\|_\sigma < \|d\|_\sigma$, agent i attains repulsion force from j by $\nabla_{\mathbf{p}_i}\psi_\alpha(\|\mathbf{p}_i-\mathbf{p}_j\|_\sigma)$. When $\|\mathbf{p}_i-\mathbf{p}_j\|_\sigma = \|d\|_\sigma$, the repulsion force and attraction force between i and j become balance. When $\|d\|_\sigma < \|\mathbf{p}_i-\mathbf{p}_j\|_\sigma < \|R\|_\sigma$, agent i attains attraction force from j. And when $\|\mathbf{p}_i-\mathbf{p}_j\|_\sigma \ge \|R\|_\sigma$, ψ_α becomes constant; then no force exists between i and j. c_1 and c_2 are positive constants which reflect the influence degree from leader's position and velocity. $a_{ij}(t)$ is the adjacent weight coefficient.

5. Experimental Results and Discussion

In our simulation, we employ a human animation software package called DI-Guy, which is commercially available from Boston Dynamics Inc. We control the moving of the urgent members and normal members using SDK interface by C++ programs. The scenario can be depicted as follows: in a city populated with a majority of civilians and 2 moving cars. The civilians consist of three urgent individuals and 50 normal members comprising 30 normal individuals and 2 normal groups with 10 members, respectively. The cars' moving characteristics are known previously. The priority values of the 50 normal agents are selected randomly from the interval $[0, 1]$. The interaction range $R = 10$ m and the predefined distance $d = 8$ m. The values of a_{ij} and ψ_α are set the same as those in [6], and $c_1 = 0.81$, $c_2 = 2\sqrt{c_1}$, and $N = 10$. The dynamic environment of the experiments is depicted in Figure 6.



(a) Urgent individual 1

(b) Urgent individual 2

(c) Urgent individual 3

FIGURE 7: The $s \times t$ spaces of urgent individuals.

In Figure 6, we set $a_{\max} = 1.2\,\mathrm{m/s^2}$ and $v_{\max} = 8\,\mathrm{m/s}$. Length of the specified paths of urgent individuals is $L_{S1} = 20\,\mathrm{m}$ and $L_{S2} = 30\,\mathrm{m}$ and $L_{S3} = 25\,\mathrm{m}$. The first dynamic obstacle (moving cars) moves at a constant speed of 10 m/s (Car1) and the second moves at a constant speed of 5 m/s (Car2). The $s \times t$ spaces of the three urgent individuals in the dynamic environment are depicted in Figure 7.

Figure 8 illustrates the reachable sets and the optimal trajectory for each urgent individual.

The visualization simulation results of the crowd behaviors in 3D space are illustrated in Figure 9.

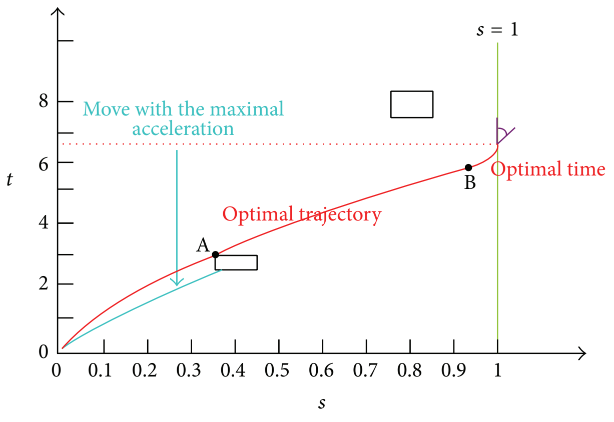
From Figure 8, we can see that the gradient in $s \times t$ space is $+\infty$ which means that the terminal velocity of the urgent individual is zero, and this satisfies the requirement of the problem definition in Section 3.1; for example, in Figure 8(b), urgent individual 2 takes the maximal acceleration until reaching A point before Car1 arrives at his path and then switches to deacceleration for the path A-B in case of colliding with Car2. After that, he continues to take the maximal acceleration for path B-C and then deaccelerates to zero from C point to the terminal point of the whole path. Figure 9 depicts realistic and believable crowd behaviors in city environment: the urgent individual can reach the destination along a given path in minimal time and be collision-free with dynamic cars. Simultaneously, the agents of the two normal groups show real gathering behaviors according to the flocking method and the normal individuals try to keep away from the urgent ones and avoid collision with each other.

The computation cost of our approach is illustrated in Figure 10 when we change the numbers of the dynamic obstacles. Similarly, Figure 11 depicts the cost time of our approach according to the number of the normal members in the crowd.

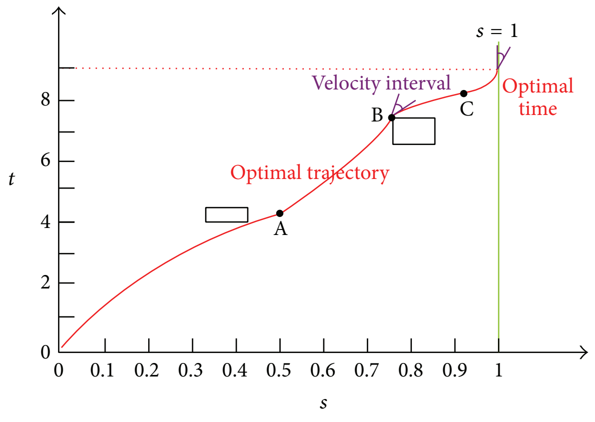
From Figure 10, we can see that the cost time of our approach is influenced vastly by the number of dynamic obstacles in the environment. However the experiment results in Figure 11 show that the cost time of our approach changes slightly according to the number of the normal members. This means that the increase in the number of the normal members does not generate much computational cost. And all the above results are in accordance with previous analyses.

6. Conclusions

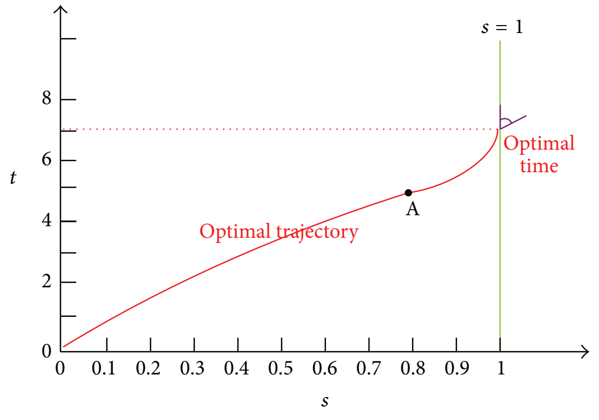
Although there have been some research studies on optimal trajectory planning for various purposes, few efforts have been conducted to simulate the realistic crowd behaviors with it, such as pedestrian navigation behavior in dynamic environment. In this paper, we first present an optimal



(a) Urgent individual 1

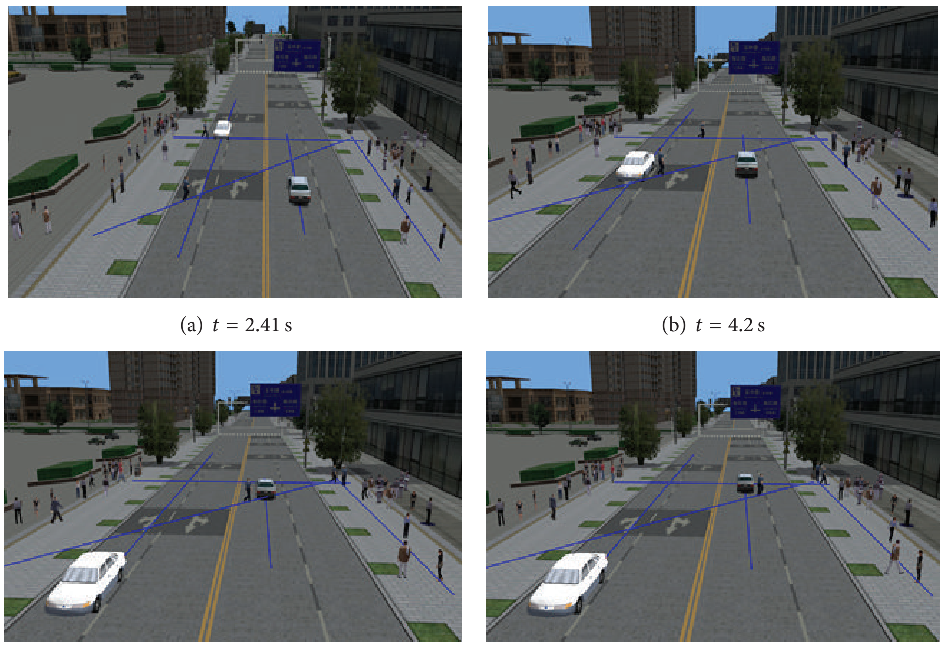


(b) Urgent individual 2



(c) Urgent individual 3

FIGURE 8: The optimal trajectory, time, and the velocity interval of urgent individuals.



(a) $t = 2.41$ s (b) $t = 4.2$ s (c) $t = 6.5$ s (d) $t = 7.5$ s

FIGURE 9: Visualization simulation results of the crowd behaviors.

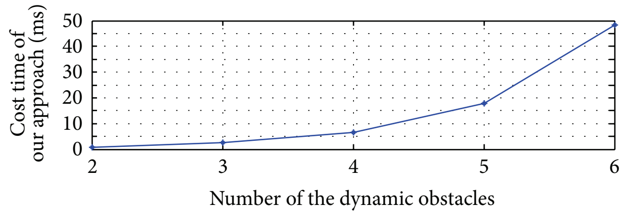


Figure 10: Cost time of our approach according to the number of the dynamic obstacles.

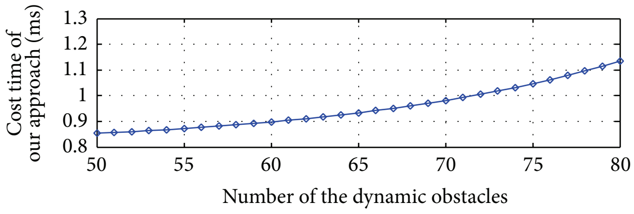


Figure 11: Cost time of our approach according to the number of the normal members in the crowd.

acceleration-velocity-bounded trajectory planning method along a fixed, given path with dynamic obstacles. We use it to generate the optimal trajectory of the urgent members under the dynamic constraints. This planner ensures that the moving time of the urgent members is minimal by using a velocity interval propagation algorithm to compute reachable velocity sets at obstacle vertices in $s \times t$ space. Moreover, the cost time of this optimal trajectory planning method is acceptable by applying a novel pushing model. Finally, combining with the collision avoidance behavior and the flocking behavior, the crowd simulation with dynamic vehicles is implemented. So, the potential of our approach for planning the optimal trajectories and modeling the social behaviors of the crowd is promising.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This paper is supported by the National Natural Science Foundation of China (Grant nos. 61170160 and 61374185).

References

[1] D. Thalmann and S. Raupp Musse, *Crowd Simulation*, Springer, London, UK, 2013.

[2] N. Pelechano, J. Allbeck, and N. I. Badler, *Virtual Crowds: Methods, Simulation, and Control*, Morgan & Claypool Publishers, 2008.

[3] S. Lemercier, A. Jelic, R. Kulpa et al., "Realistic following behaviors for crowd simulation," *Computer Graphics Forum*, vol. 31, no. 2, pp. 489–498, 2012.

[4] J. Johnson and K. Hauser, "Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2035–2041, 2012.

[5] C. Foudil and D. Noureddine, "Collision avoidance in crowd simulation with priority rules," *European Journal of Scientific Research*, vol. 15, no. 1, pp. 6–17, 2006.

[6] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[7] T. Fraichard, "Trajectory planning in a dynamic workspace: a "state-time space" approach," *Advanced Robotics*, vol. 13, no. 1, pp. 75–94, 1999.

[8] P. Fiorini and Z. Shiller, "Time optimal trajectory planning in dynamic environments," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1553–1558, 1996.

[9] C. Park, J. Pan, and D. Manocha, "Real-time optimization-based planning in dynamic environments," in *Proceedings of the International Symposium on Combinatorial Search (SOCS '12)*, 2012.

[10] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[11] K. Maček, D. Vasquez, T. Fraichard, and R. Siegwart, "Safe vehicle navigation in dynamic urban scenarios," in *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems (ITSC '08)*, pp. 482–489, December 2008.

[12] N. A. B. M. Abdullah, B. B. Abdullah, and S. Kari, "A review of collision avoidance technique for crowd simulation," in *Proceedings of the International Conference on Information and Multimedia Technology*, pp. 388–392, 2009.

[13] F. Morini, B. Yersin, J. Maim, and D. Thalmann, "Real-time scalable motion planning for crowds," in *Proceedings of the International Conference on Cyber Worlds (CW '07)*, pp. 144–151, Hannover, Germany, October 2007.

[14] S. R. Musse and D. Thalmann, "A model of human crowd behavior: group inter-relationship and collision detection analysis," in *Proceedings of the Euro Graphics Workshop*, 1997.

[15] H. Shi, L. Wang, and T. Chu, "Flocking of multi-agent systems with a dynamic virtual leader," *International Journal of Control*, vol. 82, no. 1, pp. 43–58, 2009.

[16] H. Su, X. Wang, and W. Yang, "Flocking in multi-agent systems with multiple virtual leaders," *Asian Journal of Control*, vol. 10, no. 2, pp. 238–245, 2008.

[17] H. Choset, *Principles of Robot Motion-Theory, Algorithm and Implementation*, MIT Press, 2005.