Taking Less Detour When Avoiding The Collisions

Li Ning[1], Rong Zhou[1], Yong Zhang[1], Dongxiao Yu[2] and Francis C.M. Lau[3]

***Abstract*— Existing collision avoidance approaches for multi-agent navigation often cause excessive detouring as a side effect. In this work, we propose an approach that navigates the agents with much less deviation from the ideal route, while efficiently avoiding potential collisions.**

I. Introduction

Multiagent navigation studies the navigation of a group of agents (robots, vehicles, etc.) to their targets. In complex environments, neither gathering global information nor spreading monitor commands are easy to accomplish at runtime, and so people turn to decentralized algorithms. There are various domains in which the technique of multiagent navigation may apply. For applications like swarm robots and autonomous vehicles, it is common that the agents are required to, while moving towards their targets, fulfil certain conditions such as avoiding collisions.

In fact, collision avoidance can be modeled as an optimization problem, in which any "smart" agents (i.e. agents that are capable to perform advanced calculations) could plan to move safely with an additional computational cost.

The velocity-based approach, **ORCA** [1], was introduced to compute collision-free velocities, and has been adopted in many existing works. In some cases, the agents may detour substantially from the goal-oriented path in order to avoid the collisions. This will not be a problem if the agents still have enough energy to achieve their goals. However, energy is generally limited, and hence it is preferable that the agents would deviate not so much from their desired paths, in order to save energy. Furthermore, less detour of one agent will introduce less disruption to the navigation of other agents.

a) Our contribution.: We propose a **Fresh** approach to navigate the agents with collision-free velocities and less deviation (compared to other existing approaches) from the goal-oriented paths. The name **Fresh** comes from the strategy that during turning, the agents can act slowly instead of keeping to the original speed. Our proposed approach has been experimentally validated for causing less deviation comparing to the following two approaches.

- The recently introduced **CNav** [2] approach has been reported to be more efficient than previous ones by allowing communications between agents.
- The **ALAN** approach [3] applies learning techniques in the navigation and considers collision-avoidance during the construction of preferred velocities.

In a common framework of velocity-based multiagent navigation with collision avoidance, the final velocity is generated after two stages. In the first stage, a preferred velocity is calculated, which is usually intended for navigating the agent to get closer to its target. Then in the second stage, a collision-avoidance routine applies to adjust the preferred velocity. **ORCA** [1] is usually adopted as the collision-avoidance routine, which computes a range for the current agent's velocity to ensure collision avoidance within a short time into the future. This two-stage framework is logically clear and promising. However, we argue in this paper that dealing with collision avoidance only in the second stage will cause more deviation from the goal-oriented paths as a side effect.

Here is the intuition behind the argument: if there is no consideration of collision avoidance during the first stage, the resulting preferred velocity may turn out to be a risky one in the view of the collision avoidance routine. Moreover, collision-freedom often takes the highest priority in practice. Thus, the final adopted velocity may vary greatly form the initial preferred velocity. Consequently, we should also consider avoidance of potential collisions in the construction of preferred velocity. Although this idea is not new (it was for instance introduced in [3]), we harness several heuristics to make our approach more effective in keeping the agent less deviating from the goal-oriented path.

II. Related Works

A. Collision-free navigation

The problem of multiagent navigation has been extensively studied for decades. Although there are many excellent works that have demonstrated efficient navigation of a group of agents, collision avoidance between agents is always and remains to be a big challenge. This motivated much research on studying velocity-based approaches which navigate the agents with collision-free velocities [4]. Several works [5], [6], [7], [8] studied the case of static obstacles; and [9], [10], [11] studied the dynamic case by estimating the obstacles' future positions based on their current velocities. In particular, **VO** [4] and **RVO** [12] were introduced to provide a sufficient and necessary condition for the agent to select suitable collision-free velocities, which have to satisfy certain specific conditions. The approach of **ORCA** [1] was proposed to generate collision-free velocities in a wide range of situations.

[1]Li Ning, Rong Zhou and Yong Zhang are with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China {li.ning, rong.zhou, zhangyong}@siat.ac.cn

[2]Dongxiao Yu is with the School of Computer Science and Technology, Huazhong University of Science and Technology, China dxyu@hust.edu.cn

[3]Francis C.M. Lau is with the Department of Computer Science, The University of Hong Kong, China fcmlau@cs.hku.hk

B. Coordination and machine learning

There are also works considering how to efficiently navigate the agents by assuming that they can coordinate among themselves. The agents can exchange messages via communication in order to find a feasible plan [13], [14]. Recently, the **CNav** approach was proposed [2], which assumes the agents can count their neighbors' goals while planning their own velocities. Machine learning techniques have also been employed [15], [16], [17], [18]. The **ALAN** approach [3] was proposed to efficiently navigate the agents to their targets without offline training.

III. Problem Background

a) Problem definition: Consider n agents in the $2D$-plane, denoted by $a_0, a_1, \ldots, a_{n-1}$. Each agent $a_i, i \in \{0, 1, \ldots, n-1\}$ is assigned a start position $s_i \in \mathbb{R}^2$ and a target position $d_i \in \mathbb{R}^2$ ($\mathbb{R}$ is the set of real numbers). The task of navigation is to navigate each agent from its start position to its target position without collisions with other agents and obstacles. More specifically, for agent a_i, we define the *ideal route* to be the straight line connecting the start position s_i and the target d_i. At a certain time point when agent a_i is at position p_i, the *deviation* is defined as the vector pointing from the closest point on the *ideal route* to a_i's current position p_i (Figure 1).

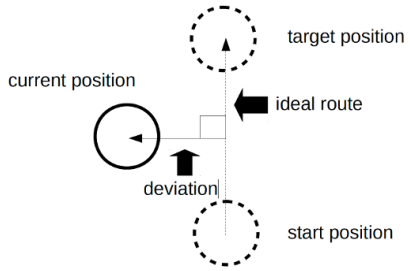


Fig. 1: Ideal route and deviation.

In this work, we aim to reduce the deviations (Definitions 3 and 4) from the goal-oriented paths, as well as to meet the general requirement of time efficiency (Definition 1).

b) The shape of the agents: In this work, we assume that all the agents are modeled as uniform discs. That is, each agent is represented by a disc in the scene, and all the discs have the same radius, denoted by γ. The obstacles, if any, are modeled as line segments in the $2D$-plane.

c) A general framework of the solution: As in the existing works, the velocity-based navigation approach usually consists of two stages.

1) In the first stage, a preferred velocity $\tilde{v}_i$ for agent a_i is calculated according to its current status and the observation of the environment. In the naïve case, $\tilde{v}_i$ can be set as the vector directed to a_i's target position, with the predefined maximum magnitude $\nu^{max} \in \mathbb{R}$.
2) In the second stage, the commonly used algorithm **ORCA** [1] is applied to generate the collision-free velocity $\hat{v}_i$ based on $\tilde{v}_i$ and the observation of the other agents' velocities v_j with $j \neq i$.

Our approach follows this framework, and the details will be presented in the next section.

IV. Our Approach: The Fresh Approach

Algorithm 1 shows the brief structure of the proposed **Fresh** approach. With this approach, each agent is assumed to know the other agents' positions and velocities. This knowledge is required for any approach that involves the **ORCA** routine because it is necessary for avoiding collisions between agents. In our approach, this knowledge also helps to calculate the preferred velocity. With **Fresh**, the preferred velocity is selected from a set of candidates, called *actions*, which are constructed based on the difference between the current position and the target, as well as the current velocity and the observations of the others' positions and velocities.

Algorithm 1: *Fresh* algorithm for agent a_i

Input:
d_i // target position of agent a_i
$P = \{p_0, p_1, \ldots, p_{n-1}\}$ // positions of all the agents
$V = \{v_0, v_1, \ldots, v_{n-1}\}$ // velocities of all the agents
1 $f_i = d_i - p_i$
2 $A = UpdateAct(v_i, f_i)$ // update the set of actions
3 $\tilde{v}_i = Pref(f_i, A, P, V)$ // the preferred velocity
4 $\hat{v}_i = ORCA(\tilde{v}_i, P, V)$ // avoid collision
5 **Perform**: $\hat{v}_i$

a) Construct the actions.: The set of actions is constructed by Algorithm 2. $\|\cdot\|$ denotes the $L2$ norm of a vector, and several actions are defined in this algorithm:

- *stay*. With velocity $(0, 0)$, the agent will immediately stop and stay at the current position.
- *slow down*. Reduce the magnitude of current velocity by a factor of $1 - \delta^{down}$.
- *turn*. Move towards the direction pointing to the target, and restrict the magnitude of velocity to a low level ν^{slow}.
- *speed up*. Increase the magnitude of the current velocity by a factor of $1+\delta^{up}$, not exceeding the maximum speed of ν^{max}.
- *keep*. Keep moving with the current velocity.

The first step of Algorithm 2 is to check whether the agent has come close enough to the target (within a distance less than ϵ). If yes, only the *stay* action and the *slow down* action would be allowed. The *slow down* action is allowed to further reduce the distance to the target, in case when staying at the current "close-to-target" position might cause collisions.

If the agent is still sufficiently far away from the target, more actions should be considered to navigate the agent. Our first two heuristics then apply.

*Heuristic 1: (**Turn immediately if needed.**)* If the agent's moving deviates from the direction pointing to the target, the agent should perform a turn (i.e. to point to the target) with highest priority.

Algorithm 2: *UpdateAct* routine for agent a_i

Input: v_i and f_i

1 $A = \emptyset$
2 **if** $\|f_i\| < \epsilon$ **then**
3 $\quad A = \{v_i \cdot (1 - \delta^{down}), (0, 0)\}$// slow down, stay/stop
4 **else**
5 $\quad$ **if** $f_i/\|f_i\| \neq v_i/\|v_i\|$ **then**
$\quad\quad$ // current orientation deviates from the target
6 $\quad\quad$ **if** $\|v_i\| > \nu^{slow}$ **then**
7 $\quad\quad\quad A = \{v_i \cdot (1 - \delta^{down})\}$
8 $\quad\quad$ **else**
9 $\quad\quad\quad A = \{\nu^{slow} \cdot f_i/\|f_i\|\}$ // direct to the target
10 $\quad$ **else**
11 $\quad\quad \nu^{up} = \min\{\nu^{max}, \|v_i \cdot (1 + \delta^{up})\|\}$
12 $\quad\quad$ **if** $\|v_i\| > \nu^{slow}$ **then**
$\quad\quad\quad$ // slow down, speed up, keep
13 $\quad\quad\quad A = \{v_i \cdot (1 - \delta^{down}), \nu^{up} \cdot v_i/\|v_i\|, v_i\}$
14 $\quad\quad$ **else**
15 $\quad\quad\quad$ **if** $\|v_i\| > \nu^{small}$ **then**
16 $\quad\quad\quad\quad A = \{v_i \cdot (1 - \delta^{down}), \nu^{up} \cdot v_i/\|v_i\|, v_i, (0, 0)\}$
17 $\quad\quad\quad$ **else**
18 $\quad\quad\quad\quad \nu^{start} = \min\{\nu^{max}, \max\{\nu^{up}, \nu^{slow}\}\}$
19 $\quad\quad\quad\quad$ // start up, stay
20 $\quad\quad\quad\quad A = \{\nu^{start} \cdot v_i/\|v_i\|, (0, 0)\}$

Output: A

*Heuristic 2: (**Move slowly to turn.**)* If the agent is moving at a high speed, it has to slow down before it makes a turn; when the agent has just turned to a new direction, it should move with a low speed.

Based on these two heuristics, if the agent is moving towards a "wrong" direction, we require the agent to make an adjustment immediately, unless the current speed is too high, in which case, a *slow down* action should be performed as soon as possible. When there is no need to turn, the agent's actions are linearly dependent on its current velocity, which could include *stay*, *slow down*, *speed up* and *keep*. However, these four actions are not always allowed in all cases when a *turn* is not needed. Here comes another heuristic.

*Heuristic 3: (**Brake gently if not facing an emergency.**)* The agent should not brake when it is moving at a high speed. The only exception (the case of emergency) is when the agent has come really close to the target, as in line 3 of Algorithm 2.

That is, when the agent is moving with speed higher than ν^{slow}, it cannot just stop. Furthermore, we also disallow *slow down* if the agent's current velocity is already very slow (less than ν^{small}).

b) Select the "best" action.: The preferred velocity is finally selected from the actions in Algorithm 3. The principle behind determining an action is "better" or even "best" is that the fewer potential collisions the better. This requires an estimate early on of how many collisions will be caused if an action is performed.

Algorithm 3: *Pref* routine for agent a_i

Input: f_i, A, P and V

1 **for** $\alpha \in A$ **do**
2 $\quad p_\alpha = p_i + \Delta \cdot \alpha$
3 $\quad c_\alpha = 0$
4 $\quad$ **for** $j \in \{0, 1, \ldots, n-1\} \setminus \{i\}$ **do**
5 $\quad\quad$ // consider agent a_i and estimate a_i's possible positions after Δ time
6 $\quad\quad P_j = \emptyset$
7 $\quad\quad \nu^{up} = \min\{\nu^{max}, \|v_j \cdot (1 + \delta^{up})\|\}$
8 $\quad\quad \nu^{down} = \|v_j \cdot (1 - \delta^{down})\|$
9 $\quad\quad$ **if** $\|v_j\| > \nu^{slow}$ **then**
10 $\quad\quad\quad P_j = \{p_j + \Delta \cdot \nu^{up} \cdot v_j/\|v_j\|, p_j + \Delta \cdot \nu^{down} \cdot v_j/\|v_j\|, p_j + \Delta \cdot v_j\}$
11 $\quad\quad$ **else**
12 $\quad\quad\quad$ **if** $\|v_j\| > \nu^{small}$ **then**
13 $\quad\quad\quad\quad P_j = \{p_j + \Delta \cdot \nu^{up} \cdot v_j/\|v_j\|, p_j + \Delta \cdot \nu^{down} \cdot v_j/\|v_j\|, p_j + \Delta \cdot v_j, p_j\}$
14 $\quad\quad\quad$ **else**
15 $\quad\quad\quad\quad P_j = \{p_j + \Delta \cdot \nu^{up} \cdot v_j/\|v_j\|, p_j\}$
16 $\quad\quad$ **if** $\exists p \in P_j$ *such that* $\|p_\alpha - p\| < 2 \cdot \gamma$ **then**
17 $\quad\quad\quad c_\alpha = c_\alpha + 1$
18 $\tilde{v}_i = argmin_\alpha c_\alpha$ // select the "best" action

Output: $\tilde{v}_i$

In this work, we also assume that all the agents adopt the same **Fresh** approach, and every agent is aware of this fact. This offers an insight for the prediction of the others' next positions. Specifically, when considering agent a_j with $j \neq i$,

- if agent a_j is moving at a high speed, then it may *keep*, *slow down* or *speed up* without exceeding the limit which is the maximum speed of ν^{max};
- if agent a_j is moving at a low speed but not very slow, then it may *stay*, *keep*, *speed down* or *speed* up;
- if agent a_j is moving very slowly, then it may *stay* or *speed up*.

A collision will be attributed to an action if performing the action will probably cause the agent's next position (position after a time duration Δ) overlaps with a possible next position of another agent. Note that we do not try to estimate the other agents' next positions as they might turn, and since the preferred velocities of the other agents are assumed inaccessible with our approach, there are no clues for performing the estimation.

c) Break ties between actions.: Naturally, we want to select the action with the fewest potential collisions for the preferred velocity (line 18 of Algorithm 3). When there are more than one such "best" choice, it will be too

indiscriminate if we just arbitrarily pick one. With the **Fresh** approach, we break the ties with a more cautious strategy.

- If the agent is very close to the target, we prefer to *stay*. This means further reduction of the difference to the target is applied only if staying leads to more potential collisions.
- If possible, we prefer *speed up* to *keep* and prefer *keep* to *slow down*.

d) Scale to large groups.: In the above arguments, we exactly require every agent to consider all the other agents' status (positions and velocities). This is obviously not friendly to those situations when the number of agents is large. However, one can restrict the approach to having an encounter with only those other agents that are within a certain distance.

V. Experiments

As motivated by the idea to "keep the agents in queues as long as possible", we focus on cases in which the agents start in several queues, and initially pointing to their targets. In the experiments, the approach is applied to each of the agents in a decentralized fashion. Each agent updates its velocity in steps with time interval Δ. During a step, the agent runs the approach to derive and perform a velocity, which guarantees collision-freedom before the next step.

A. Scenarios

We evaluate the performance of the proposed **Fresh** approach in three scenarios: *crossroads*, *narrow road* and *very narrow road*. In each of these scenarios, we compare **Fresh** with two other representative approaches: **CNav** [2] and **ALAN** [3].

In the scenario of *crossroads* (Figure 2), two roads cross at the center of the simulation area. There is a queue of agents for each of the following settings:

1) *starting at the west, and targeting the east*;
2) *starting at the east, and targeting the west*;
3) *starting at the south, and targeting the north*;
4) *starting at the north, and targeting the south*.

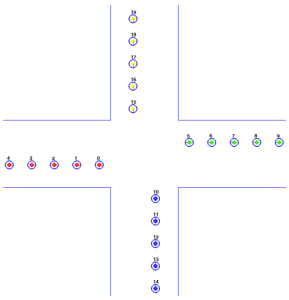


Fig. 2: The scenario of crossroads.

In the scenarios of *narrow road* and *very narrow road* (Figure 3), two queues of agents passing through a road in opposite directions. These two scenarios are almost the same, except the *narrow road* cannot allow 3 agents passing at the same time while the *very narrow road* is further narrowed to allowing exactly only 2 agents passing through. That is, 2 agents can easily pass the "narrow" road but they must be adjusted to be exactly in the right positions in order to pass the "very narrow" road.

Fig. 3: The scenario of (very) narrow road.

In all the three scenarios, there are 5 agents in each of the queues. One should consider harnessing the "just-consider-objects-in-close-range" technique (mentioned at the end of the last section), to allow many agents in a queue. In the experiments, we set the values of the hyperparameters according to Table I.

Parameter	Value	Parameter	Value
γ	$0.1\ m$	δ^{down}	0.6
ϵ	$0.01\ m$	ν^{small}	$0.01\ m/s$
Δ	$0.02\ s$	ν^{slow}	$0.15\ m/s$
δ^{up}	0.4	ν^{max}	$0.2\ m/s$

TABLE I: Specifics of hyperparameters.

B. Measurements

To compare the performance of the approaches in our experiments, we use the metrics including *arriving time*, *travel distance*, *union of deviations*, and *average deviation* which are formally defined below. In our experiments, the unit of time is in seconds (i.e. s) and the unit of distance is in meters (i.e. m).

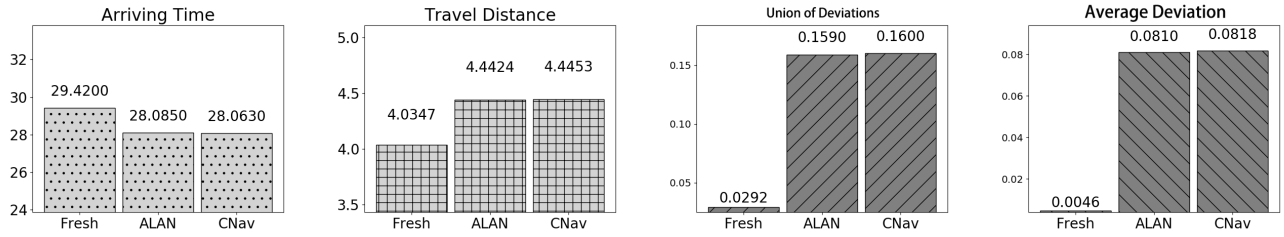
Definition 1 (Arriving Time.): The arriving time of an agent is the time duration from the start until it stops at a position that is sufficiently close to the target (i.e. within distance ϵ).

Definition 2 (Travel Distance.): The travel distance of an agent is the actual distance it travels before it arrives at the target.

Definition 3 (Union of Deviations.): For each step, the agents' deviations are calculated. Recall that the deviations are all vectors in the $2D$ space. The *union of deviations* of an agent is the summation of deviations over all steps, normalized by the number of steps. Note that a deviation could be negative or positive.

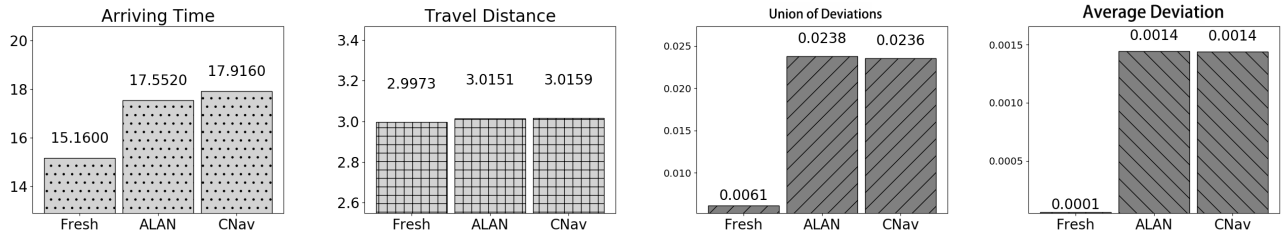
Definition 4 (Average Deviation.): For each agent, we define its *average deviation* to be the summation of the squares of the deviations over all steps, normalized by the number of steps.

For each approach in a specific scenario, we measure its performance by respectively averaging each of the measurements over all the involved agents. The results are presented in Figures 4, 5 and 6. Note that while the *average deviation* reflects how much the agent deviates from the *ideal route*, the *union of the deviations* reflects how much the deviations are symmetric along the *ideal route* (i.e. if the agent always travels on one side of the *ideal route* or not). Having both measurements gives us more details to investigate the deviations.



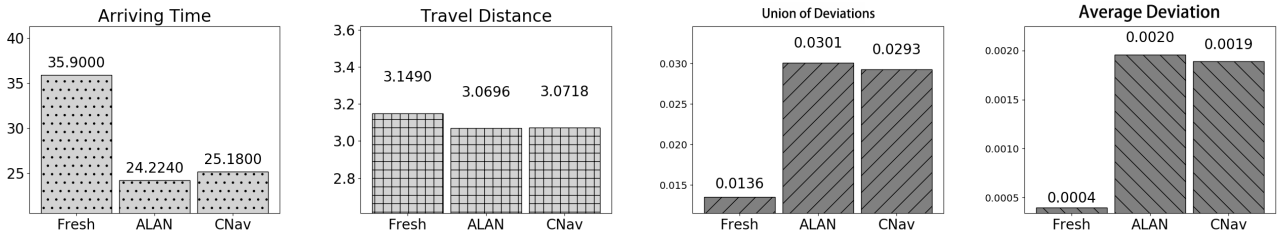
(a) (b) (c) (d)

Fig. 4: Performances in the scenario of *crossroads*



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

Fig. 5: Performances in the scenario of *narrow road*.



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

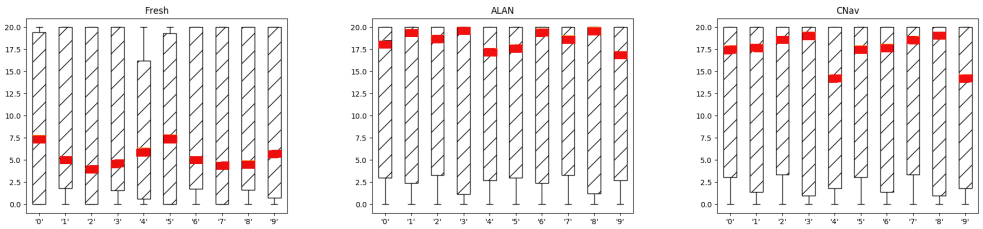
Fig. 6: Performances in the scenario of *very narrow road*.

C. Results

Figure 4 shows the approaches' performance in the scenario of *crossroads*. With **ALAN** and **CNav**, the average deviation is over $0.08\ m^2$, whereas the proposed **Fresh** approach only causes a deviation of about $0.0045\ m^2$ on average. On the other hand, the performances on the union of deviations also show a significant advantage for **Fresh**.

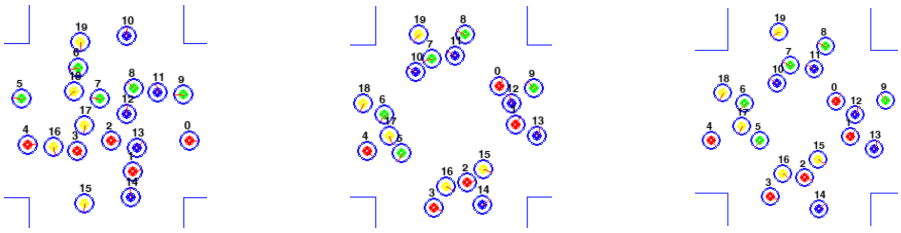
As shown in Figures 5 and 6, for the scenarios of *narrow road* and *very narrow road*, **Fresh** also leads to much better performances on the union of deviations and the average deviation, compared to **ALAN** and **CNav**. Note that in the case of *very narrow road*, all three approaches cause slightly more deviations. This is as expected because of the extremely narrow spaces between the walls. As long as it is possible, the agents will try to move away from the walls and thus deviate from the *ideal routes*. In cases where the simulation area is wide, **Fresh** also leads to a shorter travel distance. In particular, for the scenario of *crossroads*, the *ideal route* for every agent has a travel distance of $4.0\ m$, and **Fresh** gives an almost-best result. For the scenario of *narrow road* in which the *ideal route* for every agent has a travel distance of $3.0\ m$, **Fresh** also results in a very short travel distance. Note that as shown in Figure 5b, the travel distance of **Fresh** is slightly smaller than $3.0\ m$. This is because we allow the agent to stop when it is close enough to the target. Unfortunately, agents using **Fresh** do not always travel less. As shown in the scenario of *very narrow road*, the agents with **Fresh** finally arrive at the target with longer travel distances. On the other hand, compared with **ALAN** and **CNav**, the proposed **Fresh** approach has no advantage in arriving time, except for a few cases (like in the scenario of *narrow road*). The reason for **Fresh** outperforming the others in the union of deviations and average deviation partly explains why the agents with **Fresh** would travel longer and arrive later. Slower speeds and more turning actions are excuted when the agents are closed to each other in **Fresh**, so more wandering happen in the small range of deviation from the expected route.

Figure 7 shows the speed distribution for all agents in the scenario of *very narrow road*. In this case, we can clearly identify that the median speeds (the red bars in Figure 7) of agents with **Fresh** are much slower. We have shown by numbers that less deviation is caused by **Fresh**. Actually, this fact can be easily recognized by visual inspection, if



(a) (b) (c)

Fig. 7: The speed of agents in the scenario of *very narrow road*.



(a) **Fresh** (b) **ALAN** (c) **CNav**

Fig. 8: Agents are passing through the cross.

one watches the navigation processes that are shown in the video demonstration.

Figure 8 gives snapshots of moments when the agents are passing through the cross. While the agents with **ALAN** or **CNav** are spread out at the cross, the agents with **Fresh** are more or less keeping to their goal-oriented paths.

VI. Conclusions and Future Work

In this work, we investigate the cases when the goal oriented paths of the agents are really simple, i.e. a straight line from the starting position to the target position. It will be meaningful to investigate whether the proposed **Fresh** approach applies well also to more complicated situations. Also we did not scale the experiments up to trying large-sized groups of agents. We plan to investigate the performance of the proposed approach with hundreds or even thousands of agents in the future.

Acknowledgement

This work is supported by NSFC(No. 61402461, 61433012, U1435215), Shenzhen Basic Research Grant JCYJ20160229195940462, Hong Kong GRF grant 17210017.

References

[1] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[2] J. E. Godoy, I. Karamouzas, S. J. Guy, and M. L. Gini, "Implicit coordination in crowded multi-agent navigation." in *AAAI*, 2016, pp. 2487–2493.

[3] J. E. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Adaptive learning for multi-agent navigation," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '15. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1577–1585. [Online]. Available: http://dl.acm.org/citation.cfm?id=2772879.2773353

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

[5] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[6] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[7] T. Fraichard and H. Asama, "Inevitable collision statesa step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[8] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A local collision avoidance method for non-strictly convex polyhedra," *Proceedings of robotics: science and systems IV*, 2008.

[9] J. G. De Lamadrid, "Avoidance of obstacles with unknown trajectories: Locally optimal paths and periodic sensor readings," *The International journal of robotics research*, vol. 13, no. 6, pp. 496–507, 1994.

[10] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 2210–2215.

[11] L. Martinez-Gomez and T. Fraichard, "Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 100–105.

[12] J. Van Den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[13] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "An asynchronous complete method for distributed constraint optimization," in *AAMAS*, vol. 3, 2003, pp. 161–168.

[14] B. Ottens and B. Faltings, "Coordinating agent plans through distributed constraint optimization," in *Proceedings of the ICAPS-08 Workshop on Multiagent Planning*, 2008.

[15] L. Torrey, "Crowd simulation via multi-agent reinforcement learning." in *AIIDE*, 2010.

[16] F. S. Melo and M. Veloso, "Decentralized mdps with sparse interactions," *Artificial Intelligence*, vol. 175, no. 11, pp. 1757–1789, 2011.

[17] J. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Online learning for multi-agent local navigation," in *CAVE Workshop at AAMAS*, 2013.

[18] F. Martinez-Gil, M. Lozano, and F. Fernández, "Marl-ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups," *Simulation Modelling Practice and Theory*, vol. 47, pp. 259–275, 2014.