Report: A Comparative Study of Collision Avoidance Algorithms

Anja Hager (6536832) Lu Guowei (6374395) Nina Vehovec (6592112)

October 12, 2020

Abstract

The paper "Implicit Crowds" [1] provides a state-of-the-art method for collision avoidance in crowd simulation. Based on this paper, there are two major goals in our assignment 2: First, we implement this method in the UUCS crowd simulation framework; second, we conduct a comparative study among this "Implicit Crowds" (short name "IC") method and four other collision avoidance methods to evaluate the benefits and shortcomings of the algorithms. This report introduces the problems we solved, the solutions we provided, results of the comparative study and our conclusion.

1 Introduction

Crowd forces depend on position and velocities. In this paper, it proposes a simple and effective optimization-based integration scheme to combine these two forces (position and velocity). The Implicit Crowds approach provides guaranteed collision-free motion while simultaneously maintaining high-quality behavior such as smooth trajectories and it still operates well at big time step sizes.

Meanwhile, UUCS framework is a crowd simulation system. In the local planning stage, it provides four collision avoidance methods that can be enabled and used interchangeably:

1. Moussaid [2]: vision-based method, by retrieving information about the surroundings in a field of vision, a heuristic based approach is applied to find the right strategy for walking in crowds
   SGN [3]: vision-based method, Social Groups and Navigation, more coherent and socially friendly group behaviour through adding social forces
2. Karamouzas [4]: velocity-based method, based on the simple hypothesis that an individual tries to resolve collisions long in advance by slightly adapting its motion
3. ORCA [5]: Approach based on reciprocal velocity obstacles (RVO), each robot selects its optimal velocity from the intersection of all permitted half-planes of velocities that are allowed to be selected in order to guarantee collision avoidance

In the assignment, we first added the proposed 'IC' method in the collision avoidance module of UUCS framework, and after that, we provided four testing scenarios to compare the robustness, performance and energy efficiency for these five methods.

Our report contains the following three parts:

1. **Implementation:** The problems we met and solved when implementing this algorithm in the UUCS framework

2. **Comparative study:** The design and results of this comparative study

3. **Conclusion:** Our reviews and suggestions about UUCS framework based on the implementation and comparative study

2 Implementation

2.1 Problems solved in the paper

Stability of time step size. Generally, the force-based methods are robust. However, the Euler method applied to interpolate the velocity in each step has an accumulation of precision loss, this means very small time steps are required to avoid collision. The Backward Euler method can solve this problem. So in this paper [1] a new method, which enables to enlarge the time step size is provided. But the method is non-linear, so the computation is slow.

Continuous power law. Human anticipation depends on position and velocities. In this paper, they try to find a way to measure the velocity in a force-based method to predict the motion. The problem of velocity-based method is that it always obtains the conservative velocity to avoid collisions. Another important problem is the discontinuity of the velocity obstacle cone. As the cost function in a force-based method must be continuous. In this paper a continuous time-to-collision function R_{TTC} is created to smooth over this discontinuity, as shown in Figure 1:

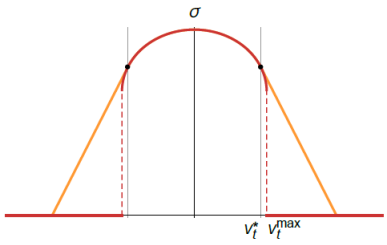

Figure 1: R_{TTC}

Meanwhile, this collision avoidance method should be robust enough in most crowd simulation scenarios so that it guarantees collision free motion, good time performance and energy efficiency. We will talk about this in the comparative study of the report.

2.2 Problems solved in implementation

The original C++ source code for the Implicit Crowds method is provided in [6]. For implementation in UUCS we need to adjust this with respect to the given global and local planning behaviour.

The calculation of new velocity When we implemented this method in the collision avoidance module of UUCS framework, the first problem we encountered was the calculation of new velocity for each character in one time step. In UUCS framework, there is a loop to calculate the new velocity of each character. But in the 'IC' algorithm a matrix calculation is applied to get the new velocity for all characters. The process is different, so we needed to adjust this module slightly based on the current UUCS specifications. As Figure 2 shows, we split the *newvelicity* function into two parts: calculation and update. So, it can support the 'IC' algorithm while following the current specifications.

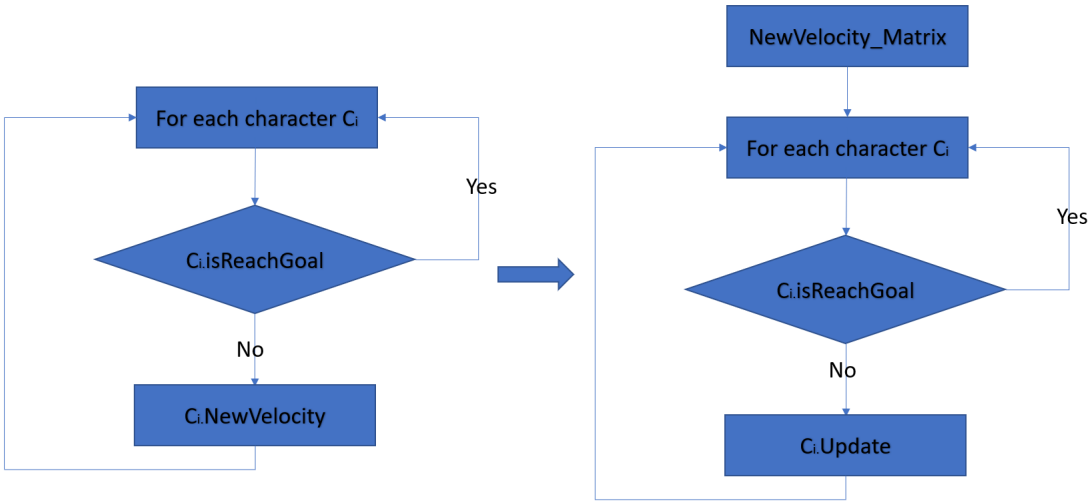

Figure 2: The process of the calculation of new velocity

Preferred velocity from global planning. In the global planning part of UUCS framework first the preferred velocity is found which is suitable to avoid static obstacles and after that the local planning will alter this velocity to avoid collisions among moving characters. But the 'IC' method does not have the same preferred velocity from the global planning. It only considers the destination to calculate preferred velocity. So our approach was that, when we calculate the preferred velocity of each character, we keep the direction from the global planning, which is enough to avoid obstacles. Then, we could adjust the speed based on the current speed and the distance between the position and destination.

Static Character. 'IC' method does not detect static characters. As Figure 3 shows, there are two characters and the dash circles are their corresponding destinations. When character 1 reaches its goal, it does not participate in the calculation of collision avoidance. So, character 2 does not know that there is a static character in front of him and will overlap with it. In the paper the road map solution in which these static characters are defined as static obstacles is mentioned as a solution, but this solution is not practical, so we were forced to consider the static characters, but the result was not good. The problem is that the head character does not notice that it will block other character and still tries its best to reach the goal. At one moment, the back character realizes there is no way to walk across the head character, there is no solution for the matrix computation and all characters are stuck. But it does avoid collision and could work as long as there is enough free space.

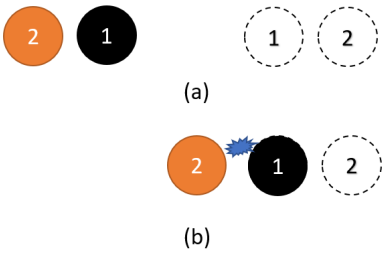

Figure 3: Static Character

Others. Because large matrix computation is needed in this algorithm, we added an open

source library named 'Eigen' in UUCS framework. The license of this library is **Mozilla Public License**.

After designing the implementation and adjusting it for our needs in UUCS, there was no real problem with the coding. Then we focused on the comparative study to test the efficiency of the new method compared to other methods in UUCS framework and we also did some optimizations to the 'IC' algorithm.

3 Comparative Study

3.1 Scenario and metric

Based on the evaluations that were already conducted in the paper [1] and the **SteerBench** benchmark suite [7] scenarios and metrics, we built three major scenarios and chose the metrics we wanted to use.

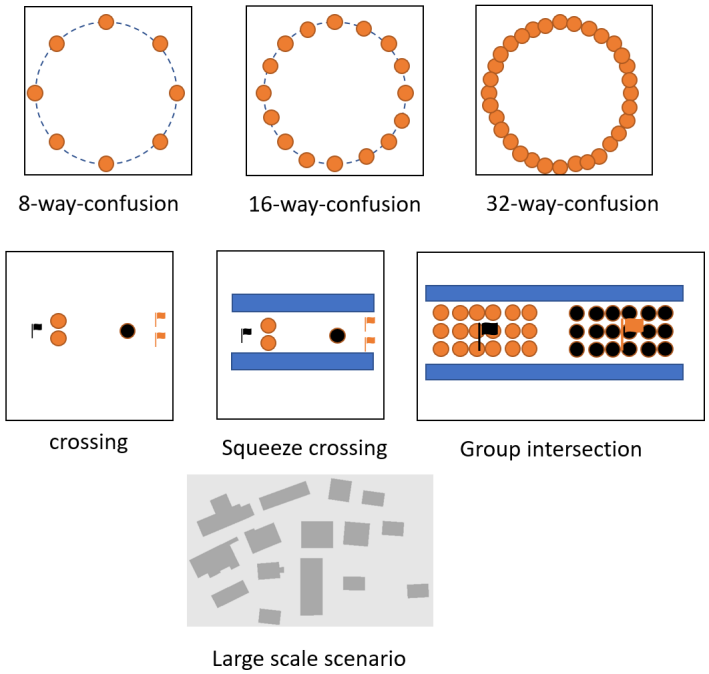

Figure 4: Testing scenarios

1. **Confusion scenario.** We set N characters along this circle and their goals are the opposite points along the diameter. The radius of the character is $1.1m$, the speed is $[0.7, 2.0]$, unit is m/s.

2. **Crossing scenario.** The first scenario is three characters without boundary, the second sub scenario is also three characters with blue obstacles as the boundary, the third sub scenario is six groups of characters. The flag with the same color is the corresponding goal.

3. **Large scale scenario.** We apply the military scene to test the simulation system. There are 200 characters in this testing scenario.

Metrics for evaluation. Table 1 lists the metrics for evaluation. We think these are the most informative figures to compare and in each scenario, the importance of these metrics could be different. Meanwhile, as the 'IC' method is supposed to support large time steps, we also test

this ability. We applied the same testing scenarios to all five collision avoidance methods in this comparative study.

Metric	Interpretation
isSucess	"passing" or "failing" the test case
Collision	are there collision events?
Time	time efficiency (lower is better)
Efficiency	the length of the path (lower is better)
Smoothness	is the track smooth?
Time Step	enlarge the time step to $0.5s$ or $1s$ for each update

Table 1: Metrics to evaluate

3.2 Evaluation

We provide more testing data and results in the attachment. Here, we only show the most important details and results of our tests and evaluation.

3.2.1 N-way-Confusion

In this testing scenario our results show that IC and ORCA perform the best.Figure 5 shows the time spent to finish the task for all five methods. When there are 8 characters, we can see that all five methods pass the task and the motion is smooth and fluent. When there are 16 characters, they can pass the task, but MOUSSAID method needs a lot of time to finish this task. When there are 32 characters, only IC and ORCA pass the task, they spend $112s$ and $228s$ respectively. Figure 6 shows the tracked positions of each character. We used 32 characters, left is 'IC' and right is 'ORCA'. Visually, the right one is more entangled and discrete. The trajectories for agents with the IC method look more smooth and don't include sudden direction changes.

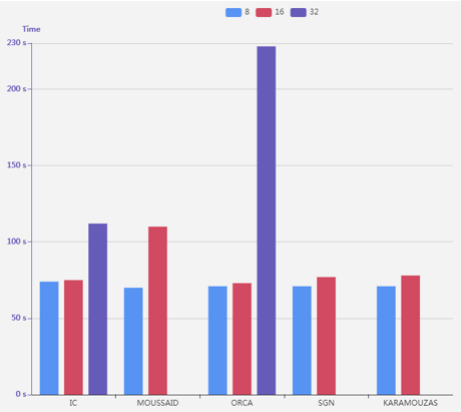

Figure 5: Time spent to finish the task

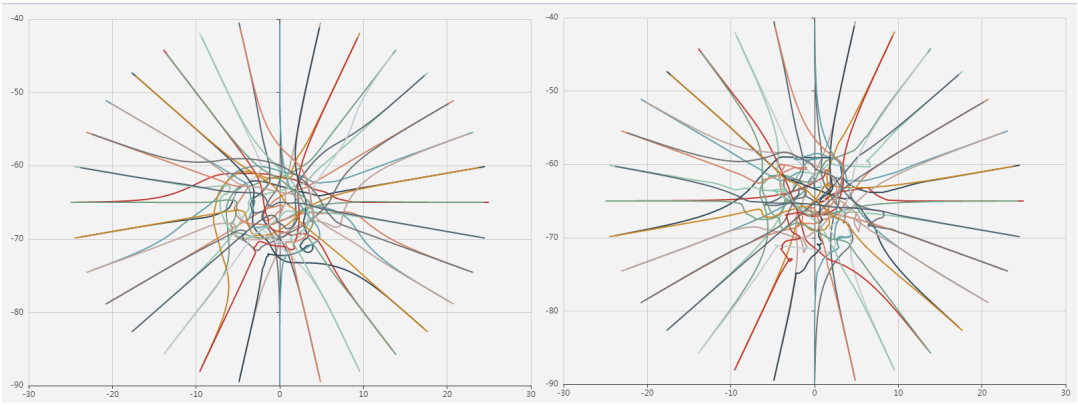

Figure 6: Tracking of IC and ORCA, 32 characters

Figure 7 shows the quality of the method. We can see, when all characters reach the centre of the circle, there is a significant congestion. The IC method (red curve) can avoid this collision in a short time, so only few characters endure low speed. Finally, when they reach the goal they remain static $speed = 0$. This figure shows that ORCA can also deal with this congestion, but more characters have to endure low speed and need longer time to finish the task. Unfortunately, the other three methods fail, they reach the first peak and get stuck with very small speed and can not solve the congestion (e.g. MOUSSAID).

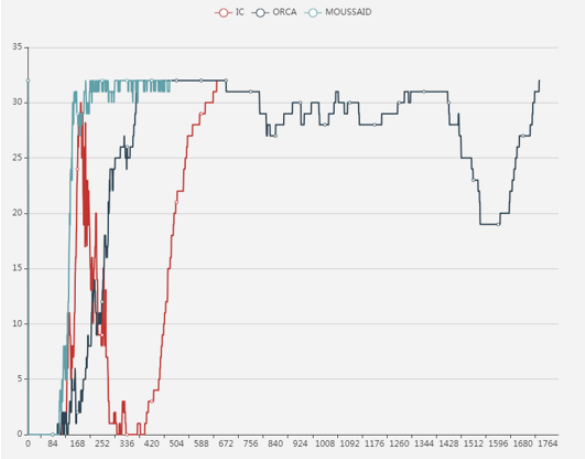

Figure 7: Congestion: number of characters with speed $< 0.2m/s$ in each time step

Table 2 is the detail of this testing scenario.

Method	Agent	Radius(m)	Speed(m/s)	Time(s)	Length(m)	Success
IC	8	1	0.7∼2	74	49	pass
	16	1	0.7∼2	75	50	pass
	32	1	0.7∼2	112	53	pass
MOUSSAID	8	1	0.7∼2	70	49	pass
	16	1	0.7∼2	110	51	pass
	32	1	0.7∼2			fail
ORCA	8	1	0.7∼2	71	49	pass
	16	1	0.7∼2	73	50	pass
	32	1	0.7∼2	228	54	pass
SGN	8	1	0.7∼2	71	49	pass
	16	1	0.7∼2	77	50	pass
	32	1	0.7∼2			fail
KARAMOUZAS	8	1	0.7∼2	71	49	pass
	16	1	0.7∼2	78	50	pass
	32	1	0.7∼2			fail

Table 2: Confusion Result

3.2.2 Crossing and group

This testing scenario has shown that vision-based methods (MOUSSAID and SGN) are the best and IC is also good. KARAMOUZAS method can pass the task, but endures the conservative velocity and ORCA fails. Figure 8 shows the tracking of characters and their speed in each time step in the crossing with boundary testing scenario. In the 'IC' method, the orange characters will walk backward slightly, so the green one can pass through them and finish this task. So there is a period when all characters endure low speed, after that, they recover the speed. With the velocity-based method, the orange characters stop and wait for the passing of green one, then, they can finish the task. But all three characters have to endure the low speed for a long time. With the vision-based method, the orange characters can walk on the down side and the green ones walk on the upper side, so they can avoid the congestion and have no influence on each others speed.

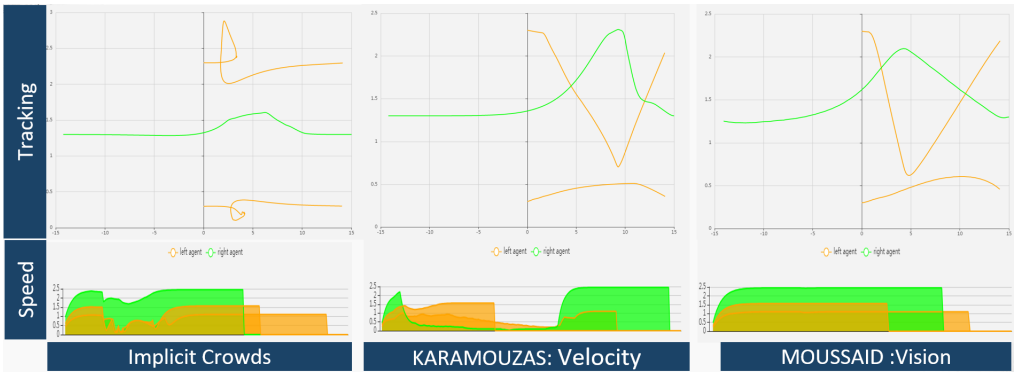

Figure 8: Crossing with boundary

Figure 9 shows the quality of paths in the crossing scenario when used with a group test. The left one is IC method and right one is MOUSSAID method. We can see that the group of the right one is disturbed while the left one can keep the group integral naturally, the paths are coordinated and look more realistic.

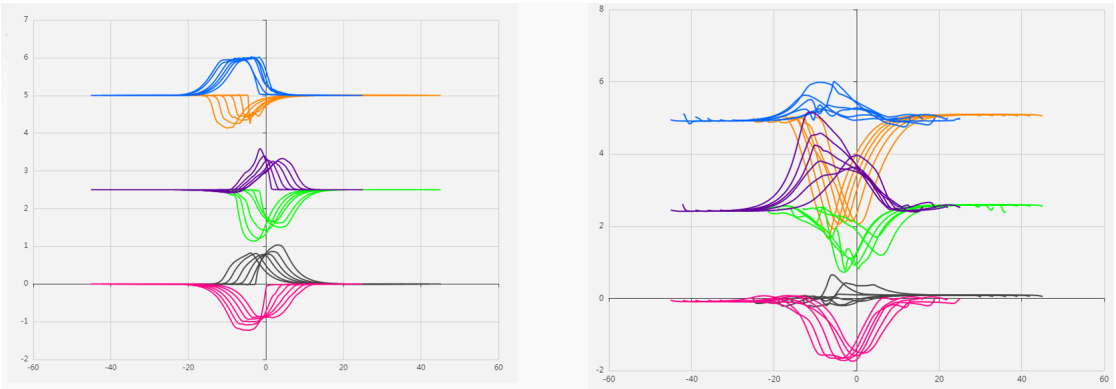

Figure 9: Crossing with group: 'IC' and 'MOUSSAID'

Table 3 shows the details of the crossing scenario.

Method	Type	Time(s)	Length(m)	Success
IC	Crossing	21	42	Pass
	Crossing With Boundary	19	64	Pass
	Crossing With Group	54	1988	Pass
MOUSSAID	Crossing	21	42	Pass
	Crossing With Boundary	14	57	Pass
	Crossing With Group	51	1995	Pass
ORCA	Crossing	17	42	Pass
	Crossing With Boundary			Fail
	Crossing With Group			Fail
SGN	Crossing	21	42	Pass
	Crossing With Boundary	14	57	Pass
	Crossing With Group	51	1993	Pass
KARAMOUZAS	Crossing	16	42	Pass
	Crossing With Boundary	25	57	Pass
	Crossing With Group			Fail

Table 3: Crossing result

3.2.3 Large Scale Scenario

In this testing scenario, we add 200 characters in the military scene, and test whether these characters can avoid obstacles and collision, solve congestion in these narrow region and finally reach the destination.

The result shows that the 'IC' method is compatible with global planning, so it can avoid these static obstacles fluently. When there is a congestion in the corner, it could find a solution to cope with this problem. However, the calculation is time-consuming, so the performance is not good. In the paper, this point is also mentioned and the solution is enlarging the time step to reduce the computational cost.

3.2.4 Large Time Step

Because IC method applies the backward Euler method, theoretically, it supports large time steps. We test this function in both Confusion and Crossing scenarios. The result shows only IC can support this ability, when using others it will lead to collision among characters.

Figure 10 shows the tracking of 32-way-confusion. Figure 11 shows the tracking of crossing with group scenario. The time step is $0.5s$(left) or $1s$(right). There is no significant difference between them, this means that the 'IC' method has quite good ability to enlarge time step.

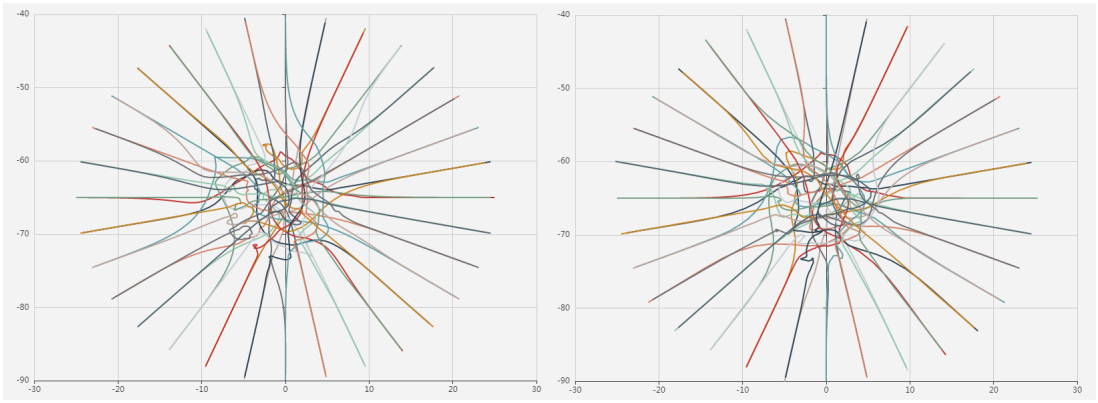

Figure 10: Large Time Step, confusion, 32 characters

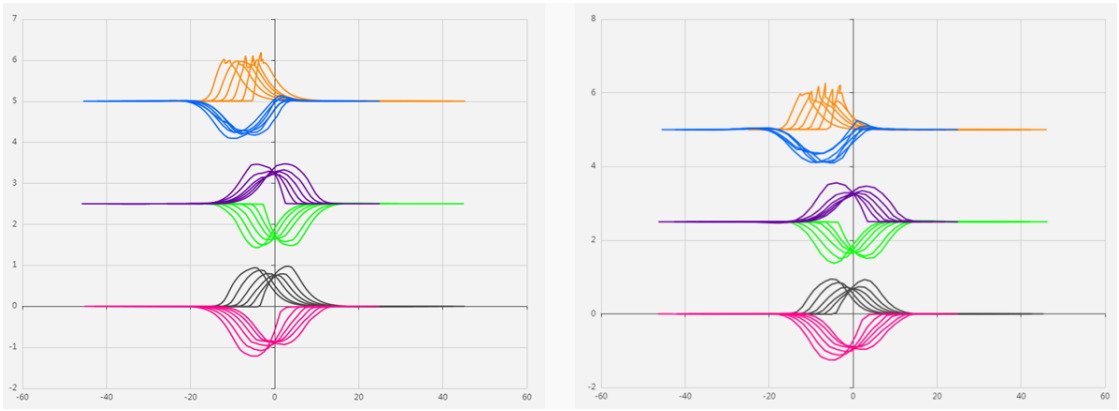

Figure 11: Large Time Step, crossing with group

4 Conclusion

Based on our implementation and comparative study, we can see that the Implicit Crowds method can pass these three basic scenarios, it seems that, the more difficult the scenario is, the higher possibility it has to pass it. So we can conclude that the IC method is robust in most scenarios.

Still IC also has many obvious drawbacks. The first problem is the matrix computation of new velocity for all characters. If there is only one character who can not avoid the collision, it will lead no solution for the entire matrix computation, which means all characters will be stuck. The second drawback is the divergence of the speed. Compared with other methods, when the speeds of characters are different with others, the IC method always finds a better solution. But when the speed is the same among characters it becomes stupid and doesn't work properly. Another drawback is the performance. For every time step, there is a $m*m$ matrix computation in the 'IC' method, m is the number of characters. Although $L-BFGS$ is applied as the numerical method, it is still time-consuming, performance could be the bottleneck if there are so many characters or in a complicated scenario which needs many iterations to find the solution.

About the comparative study, we have to admit that the comparative study is not unbiased. Even in a simple scenario, there are so many possible parameters which are potential to influence

the result. For example, we can modify the radius of the character, the preferred speed of the character, the gap between characters, and we can insert an obstacle in the pivot position. So, it is difficult to find a perfect method which could pass all scenarios, practically, the goal of the comparative study is to find the features of different methods and which scenarios the different methods are good at.

About UUCS framework, overall, it is well designed, But there are some suggestions which can make it better. In our opinion, it would be good to additionally provide an analysis toolkit to monitor this simulation system. We can record the positions and velocities in each time step ad visualize the result of analysis. It is helpful for developers to debug and meaningful for users to notice the crowd pressure, such as locating the stuck characters or potential congestion position. Furthermore, we could create a *RenderObject* module as a bridge between simulation and visualization, so we can have a clearer boundary among data, simulation and visualization. For example, in the *RenderObject* module, we could provide a class named RenderCharacter, it can store and maintain the attributes of a Character such as position, colour and radius. So when we need to render these characters, we can render these characters directly without constructing a new GraphicsElement; when there is a large volume of characters to render, we could package these characters as $BatchedCharacters$ or $InstancedCharacters$ in this module to gain a higher performance through *OpenGL* technique. Furthermore, if we support large time step ability, there should be two time steps, one is to update the position in a common time step, one is to update the velocity in a large time step. Finally, document and demo are not so good in some parts. When we create a new character for example, we need to execute computePathForCharacterID at the end, otherwise, this character does not work. To solve these issues we need to read source code.

References

[1] Ioannis Karamouzas, Nick Sohre, Rahul Narain, and Stephen J. Guy. Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Transactions on Graphics*, 36(4), July 2017.

[2] Mehdi Moussaïd, Dirk Helbing, and Guy Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884–6888, 2011.

[3] Norman Jaklin, Angelos Kremyzas, and Roland Geraerts. Adding sociality to virtual pedestrian groups. In *VRST*, 2015.

[4] Jur van den Berg, Stephen Guy, Ming Lin, and Dinesh Manocha. *Reciprocal n-Body Collision Avoidance*, volume 70, pages 3–19. 04 2011.

[5] Ioannis Karamouzas and Mark Overmars. Simulating human collision avoidance using a velocity-based approach. In *VRIPHYS 10: 7th Workshop on Virtual Reality Interactions and Physical Simulations*, pages 125–134. Eurographics Association, 2010.

[6] Karamouzas et al. Implicit crowds. `https://github.com/johnoriginal/implicit-crowds`. Accessed: 2019-01-26.

[7] Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman. Steerbench: a benchmark suite for evaluating steering behaviors. *Journal of Visualization and Computer Animation*, 20:533–548, 09 2009.

A Appendices

A.1 Further Graphics

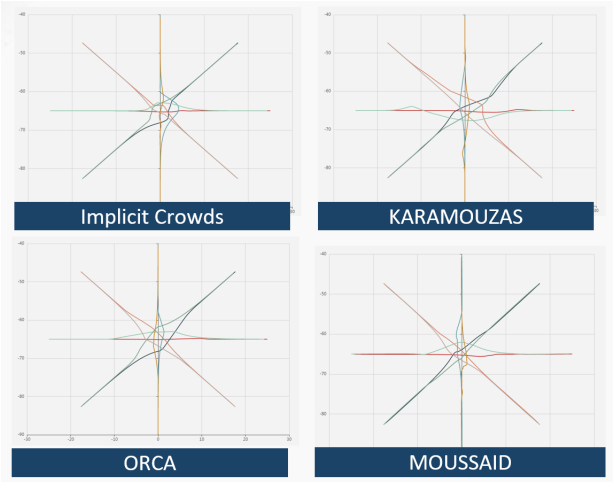

Figure 12: Tracked positions of various methods, Confusion scenario, 8 characters

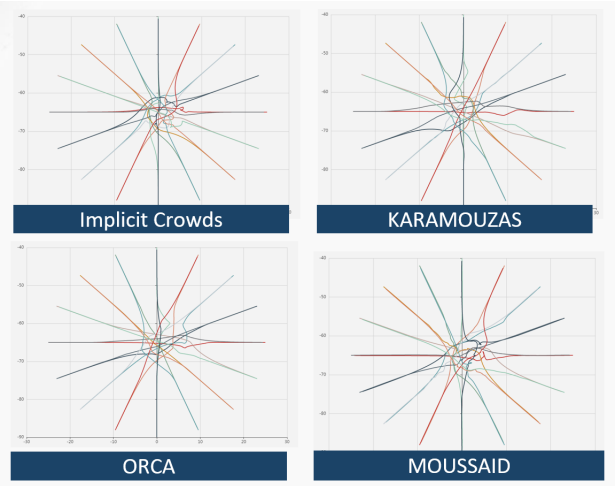

Figure 13: Tracked positions of various methods, Confusion scenario, 16 characters

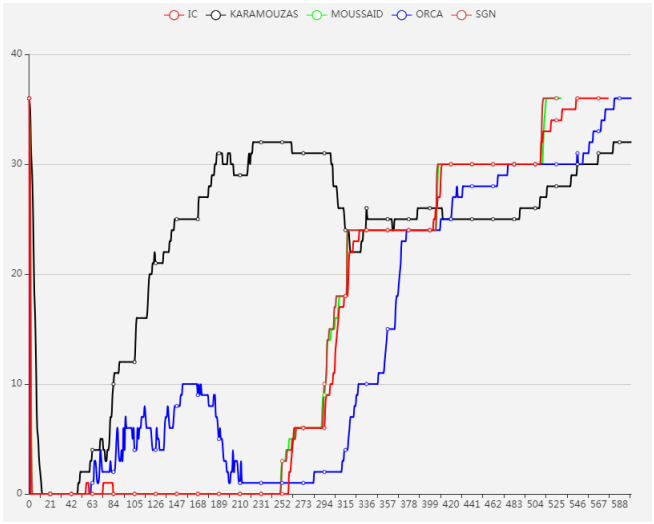

Figure 14: Congestion: number of characters with speed $< 0.2m/s$ in each time step, Crossing and group scenario

Figure 15: Large Time Step (0.5s), ORCA, Confusion, 32 characters

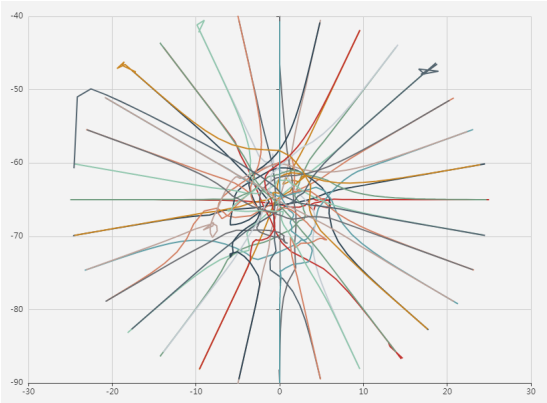

Figure 16: Large Time Step (1s), ORCA, Confusion, 32 characters

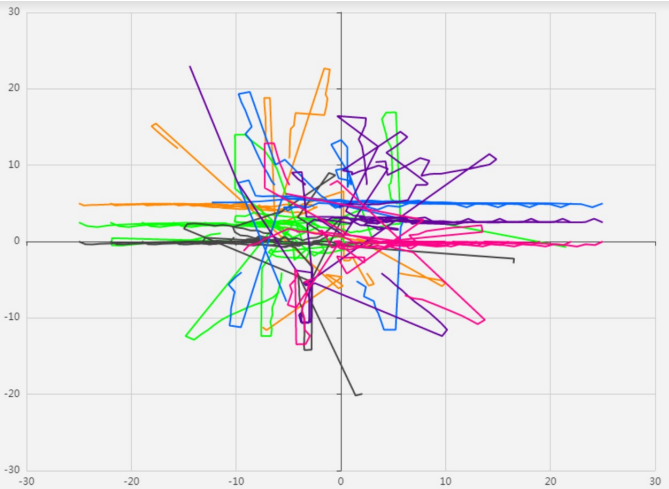

Figure 17: Moussaid: Time step size 0.5 s

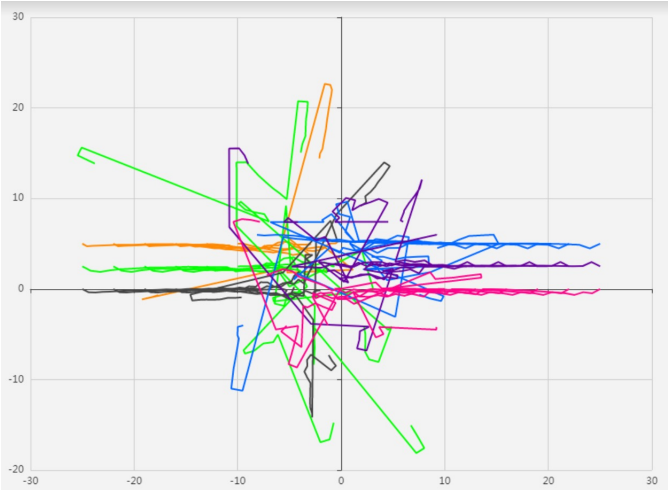

Figure 18: SGN: Time step size 0.5 s