Development of an Autonomous Vehicle Controller for Simulation Environments

Vitor Peixoto Menezes, Cesar Tadeu Pozzer
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFSM - Universidade Federal de Santa Maria
Santa Maria, Brazil
{*vmenezes, pozzer*}@*inf.ufsm.br*

***Abstract*— An autonomous vehicle controller for simulation and games environments must coordinate the movement and behavior of multiple vehicles in their chosen path (itinerary) in the given environment. This work presents the development of a controller that guides the movement and behavior of vehicles inserted in a 3D world, so they can respond accordingly to accomplish the given movement orders. Our proposed solution to this problem is mainly based on steering behavior, which were improved with adjustments and predictions to be able to better manage static and dynamic obstacles in the simulated environment. It also uses group movement models using communication and planning between agents as well as solutions to obstacle deviation in real time in order to avoid collisions between vehicles while maintaining the behavior as realistic as possible. We also use accelerating data structures to enable the solution to run in real time even in very large terrains with a large number of obstacles.**

***Keywords*-Autonomous vehicle controller; Steering Behavior; Group Management**

I. Introduction

The reproduction of human behavior in simulated vehicles by artificial intelligence is of extreme importance in any virtual environment as it gives the sensation of dynamic realism. An autonomous vehicle must have intelligent behavior with the ability to improvise given adverse situations by choosing the proper action given its limited view on the environment. There are multiple adverse situations a moving agent can face in a virtual 3D scenario such as avoid static and dynamic obstacles, group movement (convoy) and parking.

For a vehicle to navigate between two points, it is necessary to obtain the data regarding its surroundings and define the vehicle's entire route (itinerary). Once its itinerary is defined, it still may be blocked by dynamic obstacle or static obstacles (such as trees, rocks, broken vehicles, among others) not mapped by the path-finding algorithm. When planning a path for games or simulations it must be calculated accordingly to the environment, making possible to detour every main static object.

The vehicle can act alone or in a group movement with other entities. Also, there's the need to sense the surroundings in order to create a suitable representation for reasoning about the present scenario. To achieve that, was used a spatial hashing approach, which greatly improves the performance for proximity queries. Having a path defined by a path-finding algorithm, the vehicle must arrive at its destination managing every aspect of the vehicle, like turning, speed, direction, among others.

The 3D environment's representation has some impact over the planning behavior of the autonomous behavior. Given that, was used the 3D engine Unity, along with its physics libraries to support the world representation and its easy to use terrain and physics tools. The A* path-finding algorithm was used to generate the best path given a start and end points considering only the scene's static objects, allowing the controller to manage in an intelligent manner the displacement and behavior of one or more organized distributed vehicles in the simulated scenario. This is a complex task given the number of situations that need to be taken into account.

The main problem that must be solved is the movement management of one or more vehicles in a pre-calculated route considering the most basic requirements such as physical principals and follow a path using reasoning and inter-vehicle communication. Each solution proposed in this paper has its own separate implementation allowing the controller to stack multiple behaviors to perform complex movements.

II. Related Works

Planning and reasoning over a 3D environment has been subject of many works in Computer Graphics, AI and games. When planning for an agent's movement in a virtual environment, the AI must take into account all obstacles for reasoning to find at least one correct path. Since our primary concern is not graphics quality, we assume the vehicles are already modeled and the physics engine works properly.

A. Steering Behavior

The Steering Behavior technique [6] allows autonomous objects to move in a realistic way using a simple management of forces. This resultant force is calculated based on the objects surrounding the agent and the selected behavior (e.g. path following, flee, wander, seek, etc) dictates the movement direction. This approach has no performance impact since it uses simplistic local forces in neighbors that guide the agent away from the obstacle until it can return to its original path, instead of recalculate the path in real time [7], for instance. Apparently simple, it is capable of reproducing complex movement patterns and its main field of application is to reproduce the movement of a group of independent individuals.

The original implementation [6] shows that is possible to reproduce local behaviors by simply applying the resultant forces in a response vector and summing with the instantaneous velocity of the agent. Thereby, the original implementation does not controls properly the movement of agents with larger sizes. With that in mind it made clear the deficiency of this solution for richer environment and the necessity of a different approach. In section III is presented the differences between this technique and the one proposed in the present paper.

B. Real Time Path Planning

Sud et al. [7] proposed a solution for real time path-finding for multiple agents in a dynamic environment. It is introduced a new type of data called Multi-Agent Navigation Graph (MaNG) and its data is computed using discrete Voronoi diagrams. Voronoi diagrams have been extensively used for path planning computing in static environment [8], [9] and is being extended for dynamic environments.

The Voronoi diagrams codifies the connectivities of space resulting in the path with the widest space for the agent to pass. To reproduce this calculations in a dynamic scene, it is processed the Voronoi diagram of each agent treating the others as obstacles making it very costly when increasing the number of agents.

It treats complex movements given several vehicles moving in a same region, simultaneous movement of multiple agents and coordinate planning. It is not appropriated to our problem since it is not needed the best result and the cost of this approach increases exponentially with the size of the terrain. There are other solutions that does not use such computational power and offers satisfactory solution [6].

C. Information Flow and Cooperative Vehicle Formation Control

Existing approaches in the cooperative vehicle formation control are usually divided in two categories. The first is a "leader-follower" approaches, which the vehicle convoy follows a leader and mimics its steps. The second approach is "virtual leaders" [10], [11], [12], which the vehicle formation is clustered in one or more fictitious vehicles where the aggregated trajectory act as leader.

Fax et al. [1] solves the single task vehicle's group cooperation problems using channels of communications to coordinate its actions. The intention is to consider a highest number of possible connections between vehicles to understand how the information flow topology affects the stability and the system performance. The communication model uses a topology that is similar to a graph. Gathering the graph theory, the control theory and dynamic system theory, it was studied the interaction of the communication network and the vehicle dynamics to compose strategies of information exchange.

III. Proposed Solution

Our solution allows a group of vehicles to move between one position to another. To start a movement (Fig. 1), the vehicles and a destination point must be selected and sent to the Movement Manager, which has two tasks. The first task is to invoke the path-finding algorithm to calculate a rough route (itinerary) and the second is to initialize the vehicles controller by sending the vehicle's convoy order and the calculated itinerary. The path-finding algorithm uses the position of the first vehicle as origin and the destination point and takes into account only the objects mapped in the navigation graph.

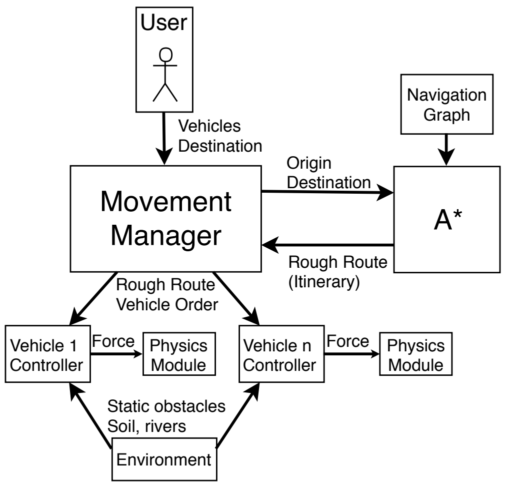

Figure 1. Solution overview. Basic steps to start a movement.

The vehicle behavior (controller) handles the decision making for one vehicle in a 3D virtual environment and is divided in three modules (Fig. 2): sensor, physics and behavior. The sensing module simply detects objects in the vicinity of the vehicle allowing better decision making. The physics module implements the interface used by the behavior and has the most basic physical function to standardize the application of forces in the model and stabilize the vehicle's movement, so all the vehicles use the same implementation. The behavior module dictates the taken decision by the vehicle in a given situation.

The vehicle's order is chosen by the user and the controllers manage themselves so they can navigate over the calculated itinerary. The environment is sensed by the behavior in order to modify most of the vehicle's properties based on the terrain surface and obstacles. The communication between vehicles happens when a vehicle needs to avoid colliding with another (Fig. 5) by acting based both on the

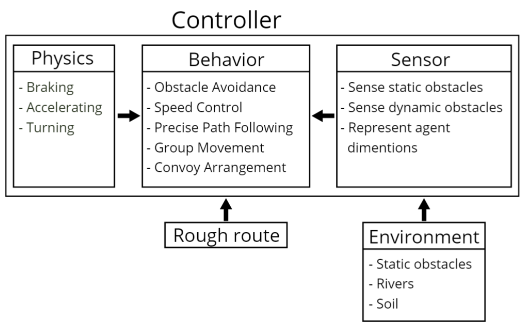

Figure 2. Visual description of the controller's modules and their functions.

scenario and their current state. Static obstacles are also detected so they can be avoided.

A. Physics module

To achieve a physically accurate simulation, it must be used a sophisticated physics engine and the same physical pattern for all vehicles. They use the following implementations for better visual and physical consistency. This module implements the vehicle's interactions with the Unity's physics API along with its colliders by modeling the application of forces on the objects.

1) Forces - Acceleration and Brake: Forces are essential for every physical simulation. The wheels are simulated using the WheelCollider component that handles the wheel's physical simulation allowing the application of force. The force's magnitude is proportional to its acceleration, vehicle's mass, friction forces between the ground and the tire, engine power and relative to the center of mass so the vehicle stays stable while moving. Accelerating is performed by applying a force directly on the tire in the direction of the movement.

Braking is performed by applying a force directly on the tire in the reverse direction of the movement. This module implements two kinds of brake. The first is a progressive brake, where the force is increased over time to simulate a slow brake and is used when the safe distance from any obstruction ahead of the agent is met. The second is a hard brake, where the maximum brake force is applied making the tire stop spinning instantaneously and is used when exists the danger of eminent collision.

2) Wheel Direction: It is defined by the angle between the current direction and the desired direction. The function receives as parameter one point in the 3D space and calculates how much the wheel's front axis must rotate so the agent faces the desired direction. Since an angle alone can not define the entire movement because it does not contain information about the rotation axis sign, it is computed the signed based on the parameters. Using both informations, it is applied a gradual rotation on the vehicle's wheels so the vehicle can face the target.

To assure the stability of any real life vehicles in turns is a complex task and is achieved by a set of specialized vehicle's parts called stabilizer bars and are implemented for better physical accuracy. These bars connect each wheel of the same axis and have the purpose to force each side of the vehicle to stabilize the heights of the wheels counteracting the centrifugal forces of the vehicle rounding the curve. When one wheel is pushed up, the bar transfers part of that force to the other wheel, so its suspension is also compressed. That limits the rotation of the car body in that axis, making it less likely to roll over.

B. Sensing Module

Sensors are specialized colliders that collect information about the objects in the vicinity of the vehicle. The implementation presented in this paper for this module uses a circle to represent static or dynamic obstacle and a rectangle (Vision) in front of the vehicle (Fig. 3) to detect and verify the most eminent obstacle for detour.

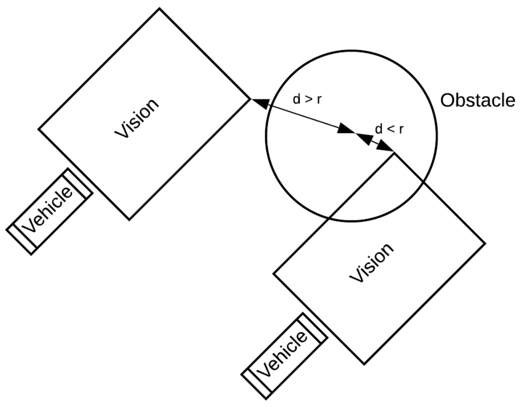

Figure 3. Collision test using sensors.

Smaller agent's dimension can be simplified with only a test ray but vehicles have bigger dimensions thus the need to adjust the vision with a collider positioned in its front and proportional to its size (Fig. 4). These configurations allow the vehicle to act with higher precision and physically accuration.

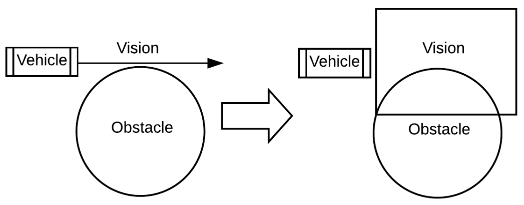

Figure 4. Difference between sensing the environment with a ray of vision and a rectangle.

To map a very large number of obstacles and allow real time access in the vehicle's surroundings is implemented the Spatial Hash Algorithm [13]. It uses a data structure that subdivide the 3D space in nodes representing position ranges and maps in each node the respective objects within the range. The implementation uses multiple hashs, each representing sets of trees, rocks and other kinds of static obstructions.

C. Behavior Module

In a given situation the vehicle must behave as it is being controlled by a human being. In this context, was selected the situations found in a movement defined by a set of points that required proper thinking of the driver and for each one was developed a solution. It is important to emphasize that the simulator's problem is to move a group of vehicles in a convoy or individually, with equal paths or not, but aways following its itinerary.

1) Collision Avoidance: The first issue was to avoid multiple obstacles that were not mapped by the path-finding. The proposed solution uses a variation of the steering behavior algorithm so it can work in vehicles. It was explained earlier that the steering behavior works with the sum of forces, but vehicles can not rotate instantaneously and theirs dimensions must be also simulated.

A vehicle's rotation must be gradual and must be applied on its wheels, not on its body. To handle this restriction, the blockage detection must be modified based on the vehicle's properties such as current speed, dimensions and maximum wheel rotation to allow the maneuver.

It is also important to analyze dynamic objects and predict their movement so the agent can avoid collisions. Since dynamic objects are neither mapped on the path-finding data nor on the static objects, multiple vehicles can still collide with each other. This problem appears when two different convoys cross each other's paths in "X" shape, for example. Since vehicles should not collide in any simulation we propose a communication system that makes one vehicle to stop while the other pass by allowing the crossing without any major issues.

While moving, each vehicle is a dynamic obstacle and is represented with a sphere sensor with its radius being twice the vehicle's biggest dimension. It is compared the direction the vehicle is heading to the direction of all other vehicles that are not in the convoy and inside the sensor. If it is detected that moving in the current direction will occur a collision the vehicle that first detected sends a message forcing the other to stop, and lock himself using a simple lock which prevents deadlock (both stopping). The vehicle that receives the message stops its movement resulting in the whole convoy to stop while registering them as static obstacle. That makes other agents to treat them as a static objects in the collision deviation logic.

The vehicle that made the other to stop continues the movement (Fig. 5) and once it detects that the first has already passed, it sends another message that restarts the other vehicle's movement.

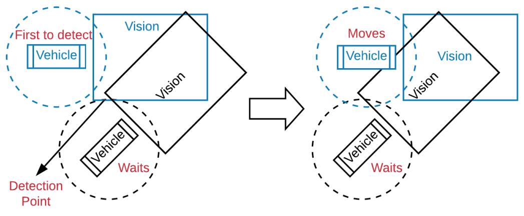

Figure 5. A vehicle predicted a collision and sent a message to the other forcing it to stop.

The obstacle deviation is not a path-finding algorithm. It only takes over the agent modifying its direction until the obstacle is avoided to finally return to its original path. There are some cases where the deviation does not generate a suitable result, it usually occurs in objects with corners such as "L" or "T" shaped obstructions [6].

2) Speed control: The speed control is calculated based on two limits: the vehicle's engine maximum speed and the user desired speed. The vehicle adjusts its speed respecting both limits and takes into account the ground surface.

When following another vehicle in a convoy the speed must be proportional to the distance of the next vehicle in formation. Also, must be maintained a minimum safe distance between vehicles allowing it to brake safely. The safe distance is calculated every frame using the current speed and the brake force every frame.

3) Parking: Parking is performed when the head vehicle (Fig. 6) in the formation reaches the last itinerary point or when the minimum distance of a parked vehicle is hit. The speed is reduced overtime by the distance from the agent and its destination, that being another parked vehicle, object or a final point.

4) Precise path following: Given a pre-computed path from a path-finding algorithm, the vehicle must follow its path precisely, passing through all the waypoints. When it arrives at one of the waypoints, the vehicles must verify the existence of another in the path and if it is the last, it has arrived in its final destination and must park.

To rotate the vehicle towards its destination the wheel's rotation angle must be calculated. The angle from the current direction to the desired direction is calculated with the following formula:

$$\theta = arccos(\frac{\vec{v1} \cdot \vec{v2}}{|\vec{v1}||\vec{v2}|})$$

In the formula above, $\vec{v1}$ is the current agent's direction vector and $\vec{v2}$ is representing the desired direction vector.

5) Group movement: The communication between vehicles for moving in convoy is of extreme importance for exchanging reports of the present situation of each vehicle

[1]. The group movement exists based on the constant communication from the head to tail vehicles (Fig. 6). They share information about current speed, destination, itinerary progress and status (eg.: waiting, broken, parked). Each agent act based on its arrangement in the convoy and on the gathered information.

To start a group movement, it must be selected the desired vehicles and the destination. Then it is computed by the path-finding algorithm the detailed route that is shared between all the vehicles. When the itinerary is defined, it is also designated the convoy vehicle's arrangement by the order that they are disposed on the list of vehicles selected by the user.

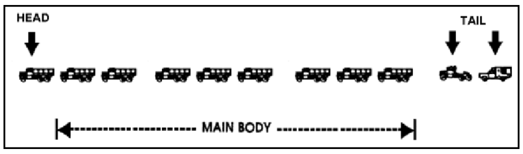

Figure 6. Convoy arrangement.

6) Assuring convoy arrangement: To assure the vehicle's arrangement, the comparison between the progress of two subsequent vehicles is made. When comparing the relative position of each, it enables the possibility to set a vehicle to wait for its successor until the right order is achieved to continue the movement.

The verification of the right vehicle arrangement can be made comparing the progress on the itinerary of both vehicles as emphasizes the formula:

$$progress = \frac{distFromLastPoint}{subPathLength}$$

A sub path is a set of two sequential points of the itinerary. To calculate the progress of each vehicle the distance from the first point (distFromLastPoint) and the sub path length is needed. The vehicle's action choice takes into account this progress comparison.

IV. Results

This work was based on the needs of a reliable and stable vehicle controller in an environment shared with other static and dynamic objects. The result is a system that controls the movement of generic vehicles in a virtual environment, allowing the simulation of any model of automobile.

When a vehicle is following its path, obstacles may stand between the automobile and its destination. In these cases the vehicle takes a detour from the original path to avoid such objects (Fig. 7). The detour takes place when a obstacles is sensed by the sensors and is detected a possible collision. The strength of the detour is proportional to the distance from the obstacle to the vehicle.

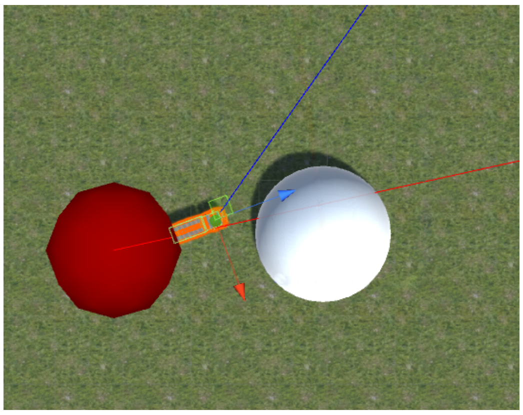

Figure 7. This image illustrate a vehicle turning away from an obstacle represented by the white shpere. The blue line represents resultant force. The red spheres and lines represent the rough path calculated by the path-finding.

The messages exchange allows vehicles to avoid colliding with another by making one wait for the passage of the other (Fig. 8).

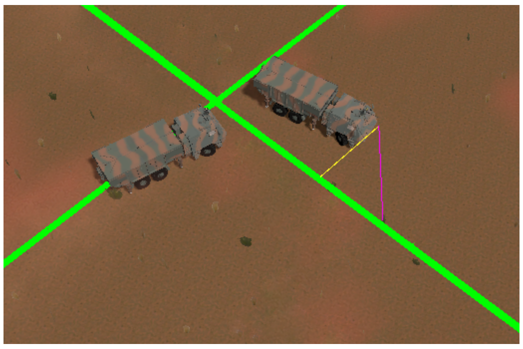

Figure 8. This image illustrate a stopped (waiting) vehicle while the other is following its path. The green lines are the rough path. The yellow line is the vehicle's projected position over the itinerary and the magenta line represents the target position over the itinerary.

The controller supports the displacement of a group of vehicles. These movements uses the same path to every vehicle and assure a minimum spacing between them (Fig. 9). In case that the vehicles are in a different order from the user's input, the convoy will reorganize itself to match the correct order.

A movement ends when the vehicles achieve their final position (itinerary's last point, parked vehicle or object) and park (Fig. 10).

V. Acknowledgments

We thank the Brazilian Army for the financial support through the SIS-ASTROS project.

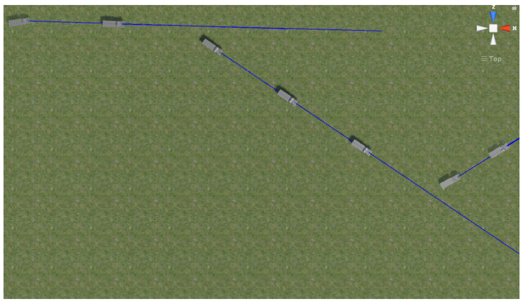

Figure 9. Vehicle group movement following the same path.

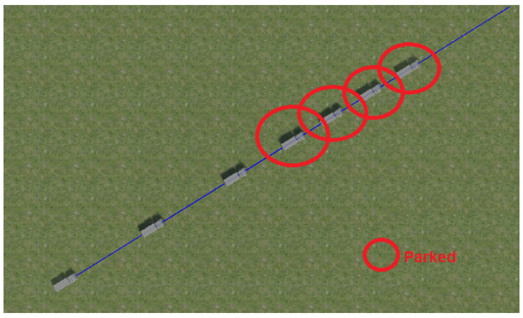

Figure 10. Vehicles parking in the correct convoy order.

VI. Conclusion

The objective of this paper was to provide a solution for autonomous vehicle in a virtual environment. Our solution provides an approximation of humanoid agent controllers for vehicles.

The goal of elaborating a generic vehicle controller was achieved once it's possible to reproduce the behavior of vehicles using this approach. It can be used in any virtual environment where there are the existence of vehicles and the need to reproduce its movement in a robust way. The controller offers vehicles decision making in any virtual environment including speed control, path following, group movement and obstacle deviation. The vehicle's physics is implemented using a robust physics engine and includes functionalities such as acceleration and braking.

Given a precomputed path by any path-finding algorithm, the controller assures that a moving agent achieves its destination. For that, the controller senses the immediate situation of the surroundings and acts based upon it.

The obtained results showed satisfactory feedback when exhaustive tests were executed in a complex virtual 3D environment, the controller found and reached its destination in every execution. This work shows a solution for vehicles group movement. Multiple increments are possible such as a more precise planning when avoiding obstacles, roads behavior, highway, built-in path-finding algorithm, among others.

References

[1] J. A. Fax and R. M. Murray, “Information flow and cooperative control of vehicle formations,” *IEEE transactions on automatic control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[2] H. Wang, J. K. Kearney, J. Cremer, and P. Willemsen, “Steering behaviors for autonomous vehicles in virtual environments,” in *null*. IEEE, 2005, pp. 155–162.

[3] A. Botea, M. Müller, and J. Schaeffer, “Near optimal hierarchical path-finding,” *Journal of game development*, vol. 1, no. 1, pp. 7–28, 2004.

[4] A. M. Bloch, D. E. Chang, N. E. Leonard, and J. E. Marsden, “Controlled lagrangians and the stabilization of mechanical systems. ii. potential shaping,” *IEEE Transactions on Automatic Control*, vol. 46, no. 10, pp. 1556–1571, 2001.

[5] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, “A survey of spacecraft formation flying guidance and control (part ii): Control,” 2004.

[6] C. W. Reynolds, “Steering behaviors for autonomous characters,” in *Game developers conference*, vol. 1999. Citeseer, 1999, pp. 763–782.

[7] A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha, “Real-time path planning for virtual agents in dynamic environments,” in *ACM SIGGRAPH 2008 classes*. ACM, 2008, p. 55.

[8] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[9] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[10] T. R. Smith, H. Hanßmann, and N. E. Leonard, “Orientation control of multiple underwater vehicles with symmetry-breaking potentials,” in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 5. IEEE, 2001, pp. 4598–4603.

[11] N. E. Leonard and E. Fiorelli, “Virtual leaders, artificial potentials and coordinated control of groups,” in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 3. IEEE, 2001, pp. 2968–2973.

[12] M. Egerstedt, X. Hu, and A. Stotsky, “Control of mobile platforms using a virtual vehicle approach,” *IEEE transactions on automatic control*, vol. 46, no. 11, pp. 1777–1782, 2001.

[13] C. T. Pozzer, C. A. de Lara Pahins, and I. Heldal, “A hash table construction algorithm for spatial hashing based on linear memory,” in *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*. ACM, 2014, p. 35.