Research on Leader-Follower-Based Sub-team Formation Control and Path Planning Algorithm Postprint

Authors: Chen Yi, Gao Yan, Wang Changbo

Date: 2018-05-20T00:00:00+00:00

Abstract

In battlefield environments, tactical unit formations are difficult to maintain effectively when confronted with complex static or dynamic obstacles. To address this issue, an improved formation control method based on the Leader-Follower algorithm is proposed. During the Leader pathfinding phase, a two-stage path search method is applied within the battlefield navigation mesh: first, the A* algorithm is employed to identify a path composed of triangular corridors and utilizable terrain features; subsequently, an improved Funnel algorithm smooths the path while considering formation scale constraints. In the Follower tracking phase, morphing techniques are utilized to generate smooth intermediate constrained formation sequences under complex obstacle constraints, which are then combined with the proposed formation spring model to locally correct and control each Follower's velocity at every time step. To resolve collision avoidance problems when facing dynamic obstacles, a method based on the relative velocity obstacle approach is developed, incorporating velocity cooperative control to prevent formation failure during the avoidance process. Finally, experimental results demonstrate the effectiveness of the proposed method.

Full Text

Preamble

Title: Research on Team Formation Control and Pathfinding Algorithm Based on Leader-Follower

Authors: Chen Yi, Gao Yan†, Wang Chang-Bo
(School of Computer Science & Software Engineering, East China Normal University, Shanghai 200062, China)

Abstract: In battlefield environments, tactical unit formations are difficult to maintain when facing complex static or dynamic obstacles. To address this problem, this paper proposes an improved formation control method based on the Leader-Follower algorithm. During the Leader pathfinding stage, we apply a two-stage path search method in the battlefield navigation mesh. First, the A* algorithm is used to find a path composed of triangular channels and traversable terrain features, followed by an improved Funnel algorithm that smooths the path under constraints considering formation scale. During the Follower tracking stage, morphing technology is employed to generate smooth intermediate constrained formation sequences under complex obstacle constraints, combined with a proposed formation spring model to locally correct and control each Follower' s velocity at every moment. To solve collision avoidance problems with dynamic obstacles, we adopt the Reciprocal Velocity Obstacle (RVO) method with velocity cooperative control to prevent formation failure during the avoidance process. Finally, experiments demonstrate the effectiveness of the proposed method.

Keywords: formation control; unit pathfinding; navigation mesh; reciprocal velocity obstacle; formation spring; morphing

0 Introduction

In combat simulation, tactical unit formation control is a critical problem. While navigating in three-dimensional battlefield environments, units must simultaneously maintain specific geometric formations. Currently, common formation control methods include behavior-based methods, artificial potential field methods, graph-based methods, and Leader-Follower methods. Behavior-based methods require modeling agent behaviors, where agents select actions such as moving forward, obstacle avoidance, or formation maintenance by perceiving the environment[1], but suffer from complexity issues. Artificial potential field methods introduce physical potential fields to represent constraints between agents and the environment through potential functions, controlling agent positions via these fields[2], but suffer from local minima problems[3]. Graph-based methods use control graphs to describe formation shapes and apply control theory to adjust formation postures[4], but are complex to implement and generally limited to simulation research[5]. The Leader-Follower method selects one agent as the formation Leader and the remaining agents as Followers, controlling their distance and angle relative to the Leader to form the formation[6]. This transforms the complex formation control problem into the Leader' s pathfinding and Followers' tracking tasks, resulting in relatively simple controller design that aligns well with actual combat scenarios, making it suitable for tactical unit formation control.

This paper builds upon the Leader-Follower approach, employing a two-stage path search method in battlefield navigation meshes composed of irregular tri-

angular networks (TIN). First, we incorporate traversability costs of linear terrain features into the cost function and use the A* algorithm to find paths composed of triangular channels and traversable terrain features. Next, we apply an improved Funnel algorithm that considers formation scale constraints to smooth the path, generating smooth movement sequences. Using morphing technology[7], we generate smooth formation transitions from source to target formations. Combined with terrain obstacle constraints, we correct and produce corresponding intermediate constrained formation sequences. We then employ the proposed formation spring model to locally correct and control each Follower' s velocity at every moment, while also introducing a feedback mechanism to dynamically adjust the Leader' s speed to prevent Followers from falling behind. For dynamic obstacle avoidance during movement, we incorporate velocity cooperative control based on the Reciprocal Velocity Obstacle method to maintain formation consistency. Finally, simulation experiments verify the effectiveness of our approach.

---

1 Algorithm Principle

The proposed improved formation control algorithm based on Leader-Follower divides formation control into two stages: the Leader pathfinding stage and the Follower tracking stage. The overall algorithm flow is shown in [Figure 1: see original paper].

Leader Pathfinding Stage: The Leader' s pathfinding in 3D scenes requires constructing a corresponding battlefield navigation mesh and applying the two-stage path search algorithm on this mesh.

Follower Tracking Stage: This stage uses morphing technology to generate intermediate constrained formation sequences, controls each Follower' s velocity through the formation spring model, and dynamically adjusts the Leader' s speed based on Follower tracking conditions.

Dynamic Obstacle Avoidance Stage: This stage calculates the RVO for each Follower, sets a unified avoidance velocity for Followers, and applies the RVO method for Leader avoidance.

2.1 Building Navigation Mesh

Navigation mesh, as the most commonly used 3D scene partitioning method, represents three-dimensional space as a network of irregular triangles. The construction of navigation mesh consists of five main steps[8]:

a) Voxelize the 3D scene, abstracting the original scene geometry into solid height fields.

b) Generate navigation regions by detecting the upper surfaces of solid height fields and partitioning them into connected regions, while eliminating non-walkable regions by checking if they are too close to obstacles or lack sufficient space.

c) Detect the contours of navigation regions and represent them as polygons, smoothing peripheral edges as much as possible.

d) Subdivide contour polygons into convex polygons.

e) Perform Delaunay triangulation on convex polygons and add height information.

The constructed battlefield navigation mesh is shown in [Figure 2: see original paper].

2.2 Two-Stage Path Search

In complex 3D scenes, using a single path search algorithm often fails to produce paths that are both optimal and smooth. Particularly in battlefield environments with various traversable terrain features such as trenches and cover, soldiers perform tactical actions like climbing and vaulting to reach destinations. Therefore, we propose a two-stage path search method. First, we use the A* algorithm in the navigation mesh to find a path composed of triangular channels and traversable terrain features, considering connectivity and traversal costs from the tactical terrain library. Then, we apply a formation-constrained Funnel algorithm to smooth the triangular mesh path.

2.2.1 A* Path Search Algorithm

The A* algorithm is a classic graph path search algorithm that can search for triangular paths from start to goal based on triangle connectivity in the battlefield navigation mesh. To enable soldiers to traverse terrain features through tactical actions during pathfinding, we pre-establish connectivity information between triangles and terrain features using a tactical terrain library. The tactical terrain library stores position information of linear terrain features (e.g., trenches, cover) and the time costs for performing tactical actions to traverse them.

In the A* search space, besides considering connectivity between triangles, we must also incorporate connectivity information between triangles and traversable terrain features, enabling A* to find paths that include terrain features. To avoid unnecessary tactical actions in local situations, the cost estimation function h(n) for state n to the goal state must consider both

distance estimation and additional costs for traversing linear terrain features like trenches and cover. Assuming the time for a soldier to perform an action is t_action and the maximum walking speed is v_max, the corresponding cost is:

$$\text{cost}_{\text{action}} = t_{\text{action}} \cdot v_{\max}$$

Therefore, the cost estimation function is:

$$h(n) = \alpha \cdot \text{dist}(n) + \beta \cdot \text{cost}(n)$$

where is the distance estimation scaling parameter, dist(n) is the Euclidean distance from node n to the goal, and is the terrain feature cost scaling parameter.

2.2.2 Improved Funnel Algorithm

The path obtained from the A* algorithm consists of triangular channels and terrain features. To acquire a smooth path in the 3D scene, we first calculate the tactical action execution positions for terrain features along the path, then compute the shortest path within each triangular channel.

The Funnel algorithm is an efficient method for finding shortest paths inside polygons, with time complexity O(n), where n represents the number of triangles after polygon triangulation. The algorithm's core idea is to identify corner points in the polygon to form a natural path by establishing a "funnel" (shown as the shaded area in Figure 3: see original paper). During triangle traversal, the funnel' s opening angle is continuously narrowed (Figure 3: see original paper). When the funnel closes, a path corner point is found, and the funnel position is reset (Figure 3: see original paper) until pathfinding completes.

Direct application of the Funnel algorithm to unit pathfinding can cause soldiers to get stuck at corners since it does not consider agent size. Therefore, we must calculate formation scale at each corner to ensure sufficient space for turning. For path W_a produced by the A* algorithm, we use an improved Funnel algorithm that satisfies formation scale constraints to generate the final path. The algorithm is described as follows:

1) Traverse path W_a, identify terrain features within it, and split W_a into local triangular channel array C using terrain features as dividers. Initialize final path W.

2) For each pair of adjacent channels with neighboring triangles p_i and p_j in C, find the two closest edges e_i and e_j between p_i and p_j. Check if they intersect with line e_st formed by the start and end points in R. If intersecting, use the intersection point as the action execution

point; otherwise, select the endpoint of e_i and e_j with the shortest distance to e_st as the action execution point.

3) For polygon channel C_i with start point s_i and end point t_i, add s_i to R. Connect s_i with the left and right endpoints n_l and n_r of the adjacent edges in s_i' s polygon, called the left line and right line.

4) Find the left endpoint of C_i' s next adjacent edge. If it lies within the region bounded by the left and right lines, update n_l to this endpoint; otherwise, keep n_l unchanged. Apply the same method to judge the right endpoint.

5) Repeat step 4) until the next left and right endpoints both lie outside the region bounded by the left and right lines. Stop the loop, where n_l (or n_r) is a corner point. Repeat step 4) until all corner points are found and saved to array T.

6) Calculate the maximum circumscribed circle radius r of the current formation. Traverse the corner point array, calculate the tangent points of adjacent circles with radius r, add them to W, and finally add t_i to R, as shown in [Figure 4: see original paper].

Repeat step 3) until all channels have been processed, and output R as the final generated path T.

---

3 Follower Tracking Stage

To maintain formation during combat simulation, the Leader-Follower method controls the distance l and azimuth angle between Followers and the Leader to form different topological network structures[9]. When unit formations need to change during combat, the common approach is to switch among preset formations[10]. However, this fixes the relative positions between Leader and Followers within a single formation, making it difficult to adapt to battlefield terrain environments and resulting in unrealistic behavior.

3.1 Formation Spring Model

In real life, springs exhibit contraction and extension when encountering obstacles to conform to them. Similarly, unit formations require the ability to dynamically conform to battlefield environments. Additionally, spring force is proportional to its length change, and when unit members are farther from their target positions, their tendency to approach (i.e., approach speed) should also be greater. This shows that unit members' movement during formation

maintenance fundamentally matches spring characteristics. Therefore, we propose constructing formation springs between Leader and Followers. Formation springs dynamically expand and contract under environmental obstacle pressure or other external forces, enabling Followers to dynamically change their relative positions to the Leader while maintaining formation as much as possible, while simultaneously controlling Follower velocities to reflect realistic situations.

As shown in [Figure 5: see original paper], A represents a unit composed of three members. If after time , the formation springs S1 and S2 of A_ are affected by external forces and change length by Δl_1 and Δl_2 respectively, forming new formation springs, then when a Follower' s formation spring length changes by Δl, the corresponding spring force F_S according to Hooke' s law is:

$$F_S = k \cdot \Delta l$$

where k is the formation spring stiffness coefficient, representing the degree to which velocity is affected by spring deformation. In this paper, k is a constant with value in (0, 50]. The Follower' s corresponding acceleration a_F is:

$$a_F = \frac{F_S}{m}$$

where m is the Follower' s mass (default m=1). The Follower' s relative velocity to the Leader at time t (t [0,]) is:

$$v_F(t) = \int_0^t a_F \, dt = \int_0^t k \cdot \Delta l \, dt$$

To make Followers move according to the formation, besides speed, we must also calculate their velocity direction. At time , if the formation spring S_i length is l_i and the angle between S_i and the Leader' s movement direction is _i, and the Leader moves at velocity v_o with unit vector n_o = (x_o, y_o, z_o), then the expected relative position between Follower and Leader is:

$$P_i(t) = l_i \cdot n_i$$

where n_i is the direction vector obtained by rotating n_o by _i. The Follower' s velocity direction at time t (t [0,]) is:

$$\psi_i(t) = \frac{P_i(t) - E_i(t)}{|P_i(t) - E_i(t)|}$$

where P_i(t) represents the Follower' s relative position to the Leader in the formation at time t.

Since formation springs change in real-time according to the scene, this may cause Followers to fall behind. To solve this problem, we introduce a feedback mechanism to dynamically adjust the Leader' s speed to ensure formation stability. Let d_max be the maximum distance between Followers' current positions and their expected positions, and d_threshold be the allowable distance deviation threshold between Followers and their expected positions. When d_max > d_threshold, meaning Followers are too far from their expected positions, we adjust the Leader' s speed as:

$$v'_{\text{leader}} = \frac{v_{\text{leader}}}{\mu \cdot d_{\max}}$$

where is the speed adjustment parameter, [1, +∞).

3.2 Formation Control

In battlefield environments, units employ various combat formations, with basic forms including triangular, echelon, line, and column formations. The maximum spacing between soldiers is 20-30 meters. This section describes formation control methods.

3.2.1 Formation Transformation Control Without Obstacle Constraints We use morphing gradient technology to produce smooth formation transitions. Morphing refers to smoothly transforming a source geometric shape to a target shape while minimizing transformation energy. Assuming the current formation is Formation A at position C1, and the target formation is Formation B at position C2, the formation transformation problem corresponds to a 2D polygon morphing problem. We use a 2D shape morphing algorithm[7] to generate intermediate constrained formation sequences; more intermediate sequences result in smoother transformations. The algorithm steps are:

1) Calculate vector sequences from each vertex to the center point for both source and target formations: S = {S_i | i = 0, 1, ···, n} and D = {D_j | j = 0, 1, ···, m}.

2) Establish mapping relationships from source formation vertices to target formation vertices, considering both the area swept and the scaling amount during S_i to D_j transformation. The cost function for S_i to D_j is Cost(S_i, D_j) = w_1 × K(i, j) + w_2 × L(i, j), where 0 w_1, w_2 1 and w_1 + w_2 = 1. The area calculation function is K(i, j) = (1/2) × ||S_i|| × ||D_j|| × sin(_ij), where _ij is the angle between S_i and D_j. The scaling calculation function is L(i, j) = | ||S_i|| - ||D_j|| |. A smaller cost function value indicates better transformation effect. We

use dynamic programming to calculate the optimal mapping.

3) According to preset time steps, generate intermediate constrained formation sequences from source to target formation based on vertex mapping.

3.2.2 Formation Transformation Control With Obstacle Constraints Battlefield environments contain various obstacle types that may prevent the default target formation from fitting at the target location, requiring target formation shape correction. We use a greedy algorithm to detect the optimal target formation geometry at the target position:

1) Initialize rotation angle to 0 and scaling factor S to 1.

2) Check if each vertex of the target formation shape collides with obstacles. If no collision, jump to step 5); otherwise, continue to step 3).

3) Increase by 3°, rotate the target geometry by . If $< 360°$, jump to step 2); otherwise, continue to step 4).

4) Multiply S by a scaling coefficient (set to 0.98 in this paper), scale the target geometry by factor S, and jump to step 2).

5) The resulting geometry is the optimal target formation shape.

The algorithm essentially performs repeated rotation and scaling operations on the default target formation to find the first corrected formation that fits at the target location. After obtaining the corrected target formation, we call the morphing technique from Section 3.2.1 between the original formation and the corrected target formation to generate intermediate constrained formation sequences under obstacle constraints.

Note that the above optimal target formation detection algorithm only ensures formation validity at the target position; intermediate formation sequences may still collide with obstacles. We provide two strategies to address this. One applies the same greedy algorithm to each intermediate formation to detect a corrected geometry that satisfies constraints, then controls agent movement through the formation spring model. The other allows automatic formation switching when the target geometry compression ratio becomes too large (generally in narrow passages or complex obstacle environments), automatically switching to a line formation to avoid excessive formation compression.

3.2.3 Formation Movement After generating and correcting intermediate constrained formation sequences, the Leader begins moving along the path. The formation springs between Followers and the Leader expand or contract to the

positions corresponding to the intermediate constrained formation within the allotted time, generating velocities at each moment according to equations (5) and (7). Simultaneously, to prevent Followers from getting stuck by obstacles during movement, the unit must continuously detect whether current formation springs are compressed by obstacles at each moment, and if so, locally update Follower velocity directions to avoid obstacles.

---

4 Dynamic Obstacle Avoidance

In combat simulation, 3D scenes inevitably contain static and dynamic obstacles. With the introduction of formation springs and navigation mesh, static obstacle avoidance can already produce smooth paths, and unit formations can dynamically switch according to obstacles. For dynamic obstacles, we adopt a formation-maintaining Reciprocal Velocity Obstacle method[11].

Assume two agents A and B exist. The velocity obstacle VO^ _{A|B} represents the set of relative velocities that cause A and B to collide after time . If agents are represented as circles with centers P and radii R, the velocity obstacle is:

$$\mathrm{VO}^{\tau}_{A|B} = \{v | \exists t \in [0, \tau], (P_A - P_B) - (v_A - v_B) \cdot t < R_A + R_B\}$$

The corresponding Reciprocal Velocity Obstacle RVO^ _{A|B} is:

$$\mathrm{RVO}^{\tau}_{A|B} = \{v | 2v - v_A \in \mathrm{VO}^{\tau}_{A|B}\}$$

In typical scenarios, any velocity outside the relative velocity obstacle range suffices for dynamic collision avoidance. However, direct application to unit collision avoidance causes formation dispersal because Followers within the unit may choose inconsistent avoidance directions, as shown in [Figure 6: see original paper].

To solve this problem, we add unit member velocity cooperative control to the RVO method by establishing a unified avoidance velocity v_avoid for Followers. For each Follower, we calculate the RVO^ with nearby dynamic obstacles, ensuring v_avoid RVO^ while keeping v_avoid as close as possible to the original v_Leader to minimize avoidance magnitude. The avoidance velocity calculation process is shown in [Figure 7: see original paper], where the left side shows the encounter between Unit 1 and Unit 2, and the right side shows the RVOs for Unit 2 members. The avoidance velocity v_avoid cannot lie within the gray dashed polygon while remaining as close as possible to the original velocity.

In some cases, collisions are unavoidable, such as when two units meet in a narrow passage where neither can find a feasible avoidance velocity. When such situations occur, we choose one conflicting party to wait in place, allowing the

other to proceed, and resume the waiting unit' s movement after the conflict is resolved.

---

5 Experimental Results

We implemented a combat simulation system based on Unreal Engine 4 using the improved unit formation control algorithm. We conducted simulation experiments on unit formation transformation control in complex obstacle environments and unit dynamic collision avoidance, achieving 60 frames per second on a PC (i7-7700HQ, 16 GB RAM, GTX 1060).

Formation Transformation Results: Consider a four-person squad initially arranged in a triangular formation. When ordered to maneuver to a target position and transform into a square formation in a complex obstacle scene, the movement effect is shown in [Figure 8: see original paper]. When the unit reaches Area A, it determines that the terrain cannot accommodate the intermediate formation calculated by morphing, so the unit automatically switches to a line column formation. After leaving Area A, the unit transforms to a square formation in Area B and continues moving. When encountering the narrow Area C, the formation compresses to fit appropriately and finally reaches the target position. The figure shows that intermediate formations generated by morphing, combined with formation spring control of each Follower' s velocity at every moment, produce smooth and reasonable unit control.

We also implemented other formation control methods for comparison: a Leader-Follower method with tangent obstacle avoidance and an artificial potential field-based formation control method[10], applied to the same scene as [Figure 8: see original paper]. The results are shown in [Figure 9: see original paper]. The boxed area in Figure 9: see original paper shows that the artificial potential field method disperses the formation when avoiding obstacles. The boxed area in Figure 9: see original paper shows that the formation switches to a single-file column due to obstacle constraints, while our method maintains a square formation by shrinking it using the correction algorithm from Section 3.2.2.

Dynamic Obstacle Avoidance Results: When two units meet, the local collision avoidance effect is shown in [Figure 10: see original paper]. Figure 10: see original paper shows the initial encounter. Figure 10: see original paper and (c) demonstrate that units maintain their triangular formations during local collision avoidance without dispersal.

---

6 Conclusion

This paper proposes an improved formation control method based on the Leader-Follower algorithm that enables units to maintain formations effectively in com-

plex battlefield environments. The algorithm divides unit control into Leader pathfinding and Follower tracking stages. During Leader pathfinding, the A* algorithm and improved Funnel algorithm produce smooth paths considering formation scale constraints. During Follower tracking, morphing technology generates intermediate constrained formation sequences, and the formation spring model between Leader and Followers locally controls and corrects each Follower' s velocity at every moment. A feedback mechanism dynamically adjusts the Leader' s speed to prevent Followers from falling behind, while the Reciprocal Velocity Obstacle method with velocity cooperative control ensures formation consistency during collision avoidance. Finally, simulation experiments verify the algorithm' s feasibility.

References

[1] Gao Qiang, Lü Donghao, Pang Yi. Research on multi-robot formation keeping and transformation simulation[J]. Automation & Instrumentation, 2012, 27(10): 4-7, 52.
[2] Zhai Hongsheng, Wang Jiaxin. A new method for robot dynamic path planning based on artificial potential field[J]. Journal of Chongqing University of Posts and Telecommunications: Natural Science Edition, 2015, 27(6): 814-818.
[3] Nair R R, Behera L, Kumar V, et al. Multisatellite formation control for remote sensing applications using artificial potential field and adaptive fuzzy sliding mode control[J]. IEEE Systems Journal, 2015, 9(2): 508-518.
[4] Li Lei, Li Xiaomin, Zheng Zhonggui, et al. Distributed formation control method for quadrotor UAVs based on consensus theory[J]. Electronics Optics & Control, 2015, 22(10): 19-23, 29.
[5] Wang Jia. Control and stability analysis of multi-agent systems[D]. Nanjing: Nanjing University of Science and Technology, 2013.
[6] Huang Jue, Qin Long, Yin Quanjun, et al. Research on CGF formation and obstacle avoidance behavior[J]. Application Research of Computers, 2012, 29(12): 4466-4468.
[7] Zhao Jianming. Research on group formation control method based on geometric constraint mechanism[D]. Hefei: Hefei University of Technology, 2013.
[8] Hale D H. A growth-based approach to the automatic generation of navigation meshes[D]. Charlotte: The University of North Carolina at Charlotte, 2011.
[9] Yuan Xuemin. Research on quadrotor UAV formation control system based on Leader-Follower theory[D]. Nanjing: Nanjing University of Posts and Telecommunications, 2016.
[10] Huang Jue. Research on CGF formation control method[D]. Changsha: National University of Defense Technology, 2012.
[11] Van Den Berg J, Lin M, Manocha D. Reciprocal velocity obstacles for

real-time multi-agent navigation[C]// Proc of IEEE International Conference on Robotics and Automation. 2008: 1928-1935.

Note: Figure translations are in progress. See original paper for figures.