Simulating Formations of Non-Holonomic Systems with Control Limits Along Curvilinear Coordinates

Athanasios Krontiris, Sushil Louis, and Kostas E. Bekris

Computer Science and Engineering Department
University of Nevada, Reno - Reno, NV, 89557, USA
bekris@cse.unr.edu

Abstract. Many games require a method for simulating formations of systems with non-trivial motion constraints, such as aircraft and boats. This paper describes a computationally efficient method for this objective, inspired by solutions in robotics, and describes how to guarantee the satisfaction of the physical constraints. The approach allows a human player to select almost an arbitrary geometric configuration for the formation and to control the aircraft as a single entity. The formation is fixed along curvilinear coordinates, defined by the curvature of the reference trajectory, resulting in naturally looking paths. Moreover, the approach supports dynamic formations and transitions from one desired shape to another. Experiments with a game engine confirm that the proposed method achieves the desired objectives.

Keywords: Path Planning, Flocking and Steering Behavior, Physics-based Motion, Navigation and Way-finding

1 Introduction

Many computer games and simulations involve the modeling of agents, such as airplanes and boats, which have to move in formation to reflect the behavior of their real world counterparts. For example, military games can utilize formations to move teams of airplanes before attacking a user specified target. This paper is especially motivated by training simulations for military officers, which must model realistically the maneuvers of opposing and friendly aircraft and boats. The need to model formations arises as a primary requirement during the development of such simulations.

An important characteristic regarding the motion of airplanes and boats is the presence of non-holonomic constraints as well as limits in velocity and steering angle. For instance, aircraft have to maintain a minimum velocity and cannot turn beyond a maximum curvature. These constraints complicate the motion planning process for these systems, especially when a formation has to be maintained. Traditionally, games sacrifice either the accurate modeling of the motion constraints or the accurate maintenance of a formation so as to achieve computationally inexpensive and simple solutions. This paper is motivated by the objective of simulating an aircraft formation while respecting the physical constraints of the aircraft, modeled as non-holonomic systems with limits in velocity and curvature. A low computational cost solution is proposed that builds upon motion planning solutions for formations of mobile robots [3].

In the proposed approach, the user specifies a destination for a team of aircraft. A reference point for the formation moves independently so as to achieve this goal and the aircraft move in response so as to maintain the formation. The parameters of the formation are expressed as the difference between aircraft and the reference point in curvilinear coordinates rather than the usual rectilinear coordinates, as shown in Figs. 1 and 2. The curvature of the curvilinear coordinates is the instantaneous curvature of the reference point. This approach allows the exact, geometric computation of the velocity and curvature for each follower aircraft based on the reference point's control parameters. The computation, however, may violate the velocity or curvature limits for the airplanes. This paper shows how to impose constraints on the motion of the reference agent so that the limits are always satisfied for the followers. It also provides details for the correct implementation of the scheme. Simulated results illustrate that the proposed scheme allows a human player to guide on-the-fly aircraft formations defined over curvilinear coordinates.

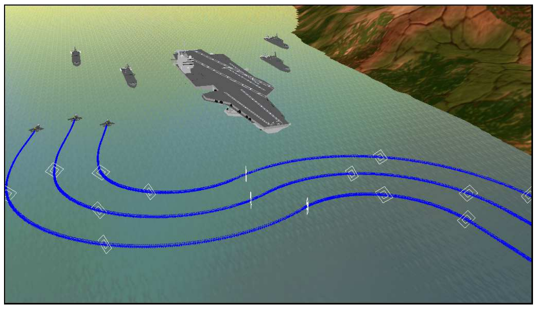


Fig. 1. Blue lines correspond to the trajectory of airplanes. Triplets of white markers point to the positions of the aircraft at the same points in time.

Background Many research challenges in computer games deal with the motion of multiple agents, such as crowd simulation [4, 12, 15, 19, 20], pedestrian formations [8, 22], shepherding and flocking behaviors [17, 26]. It has been argued that formations can help to model crowd motions such as tactical arrangements of players in team sports [24]. Kinematic formations can be defined using a general framework for motion coordination that employs generalized social potential fields [11]. Nevertheless, most work on popular game agents, such as people, employs simple motion models. It is important to explore algorithms for agents who obey complex motion constraints (e.g., non-holonomic ones), and dynamics [6].

Formations for aircraft are typically studied in aerospace engineering and robotics. Motion planning and control techniques, developed originally in these areas, are today mature enough to be used in computer games [5]. The various approaches to formation control can roughly be divided into three categories: (i) Behavior based approaches [1, 2, 10] design simple motion primitives for each agent and combine them to generate complex patterns through the interaction of several agents. The approach typically does not lend itself to a rigorous analysis but some schemes have been proven to converge [13]. (ii) Leader-follower approaches [3, 7, 9, 25] designate one or more agents as leaders and guide the formation. The remaining agents follow the leader with a predefined offset. (iii) Rigid-body type formations [16] maintain a constant distance between the agents' configurations during all motions.

Control theory in particular has been used to study aircraft formations with tools such as input-output feedback linearization for leader-follower formations [9], where the stability of the controller is typically the issue [25]. Control-theory is often integrated with graph theory to define controllers that allow the transition between different formations [7]. Graphical tools have also been used to study whether there exist non-trivial trajectories for specified formations given the agent's kinematic constraints [23]. Moreover, a formation space abstraction has been defined for permutation-invariant formations of robots that translate in the plane [14]. More recently, there have been methods for distributed task assignment so that mobile robots can achieve a desired formation [18].

This paper builds upon a motion planning method for formations of mobile robots [3], which appears appropriate for gaming applications. It allows a human player to control the formation as a single entity by defining a desired goal. In contrast to the optimization and linearization tools typically employed for formations, the technique is geometric and exact. It provides equations so that each follower tracks the reference trajectory but instead of maintaining a fixed distance to the leader, it adapts the shape during turns so as to satisfy the non-holonomic constraints. When a formation is turning, the method automatically plans for followers on the "outside" to speed up and robots on the "inside" to slow down, resulting in natural and pleasant formations to the human eye. The approach is also very fast, allowing for good scalability and real-time operation. Moreover, it supports almost any arbitrary geometric formation and is homogeneous, since each agent uses the exact same algorithm given certain parameters. Furthermore, it supports dynamic formations, which allow transitions from one shape to another.

Contribution Beyond proposing this method as a way for achieving formations for systems with non-holonomic constraints in games , this paper contributes a series of algorithmic improvements: **1)** The controls of the leader are computed on the fly based on user-specified targets. In the previous work, the knowledge of a precomputed path is utilized to maintain the formation. **2)** The approach models non-holonomic systems with a minimum velocity, which cannot move backwards, such as airplanes. Then it defines constraints in the leader's motion that allow the followers to respect such physical limits as well as guarantee formation maintenance. Note that the original controller would violate such constraints, since it can return paths for mobile robots that required them to move backwards or turn with a very high curvature. **3)** The approach extends the application of the limits for the formation's leader to the case of dynamic formations, which can be used so as to switch between user-specified static formations. **4)** The paper describes the implementation of the overall technique within a typical graphics rendering loop and provides the related implementation details.

2 Problem Setup

The model employed is that of a simple car, that is a planar, first-order, non-holonomic system, with state parameters $[x_i,\, y_i,\, \theta_i]$ (Cartesian coordinates and orientation). The controls correspond to the velocity u_i and steering angle ϕ_i, which relates the path's curvature K_i ($K_i = \tan(\phi_i)$). For system i:

$$\dot{x}_i = u_i \cos\theta_i, \qquad \dot{y}_i = u_i \sin\theta_i, \qquad \dot{\theta}_i = u_i \tan\phi_i, \tag{1}$$

where the velocity and curvature are limited similar to an aircraft:

$$|K_i| \leq K_{max}, \qquad 0 < u_{min} \leq u_i \leq u_{max} \tag{2}$$

It is also possible to consider different limits for each aircraft.

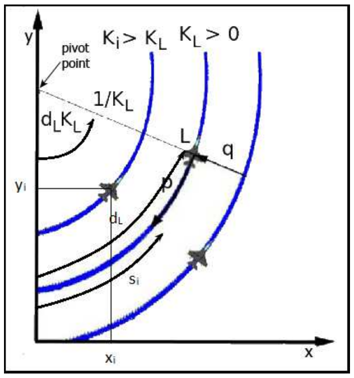


Fig. 2. An illustration of a formation along curvilinear coordinates.

The objective is to maintain a static formation defined over curvilinear coordinates, which are a coordinate system in which the coordinate lines may be curved. The formation is defined relative to a reference agent, which is directly controlled by the human player and follows controls u_L and K_L. The reference agent can correspond to a leader aircraft or to a virtual reference point. Then, the curvature of the curvilinear coordinates becomes the instantaneous curvature of the reference agent. For example, if airplane L in Fig. 2 is the leader, then the middle blue line corresponds to the "x" axis of the curvilinear system, and the line perpendicular to the airplane is the "y" axis. To maintain a static formation, each follower aircraft must maintain p_i distance along the leader's curved trajectory and q_i distance along the perpendicular direction, as in Fig.2. Finally, the variable d_L denotes the distance the reference agent has covered and $s_i = d_L + p_i$ denotes the distance that the projection of follower i has covered along the leader's trajectory.

A human player must be able to select n aircraft, which are flying independently in the game, specify the type of formation and a goal that the team must achieve. Then the aircraft must move so as to reach the vicinity of the goal while maintaining the formation along curvilinear coordinates and respecting Eq. 1 and 2. The human player should be able to change the goal on the fly and the aircraft should adapt their trajectory.

3 Aircraft Formations along Curvilinear Coordinates

The approach automatically selects a reference point, which in the resulting formation will end up not having any aircraft ahead of it. This point moves independently to reach the goal specified by the user. For an obstacle-free environment, this can be achieved with a PID controller. The steering angle ϕ_L of the leader is changed to minimize the difference between θ_L and the direction to the goal. The controller makes the leader to gradually turn towards the goal and then execute a straight-line path. As the leader approaches the goal, it reduces velocity and initiates a circular trajectory around it defined by the maximum curvature and the original position from which the controller was initiated. The leader respects the control limits of Eq. 2. Different types of planners and controllers can be substituted for the leader, especially for the case of environments with obstacles, but the current solution achieves low computational overhead for planar aircraft formation, which typically operate in obstacle-free environments. Furthermore, the approach described here can be directly applied when the player "flies" one of the airplanes or when the target for the reference point is a moving goal, such as another airplane or a target on the ground.

3.1 Equations for Maintaining a Static Formation

Consider a follower aircraft, which must retain a constant distance p_i and q_i from the leader along curvilinear coordinates as described in Section 2. Then, given Figure 2 and basic trigonometry on angle $s_i K_L(s_i)$ at the pivot point, the coordinates of the follower aircraft can be expressed as:

$$x_i = (\frac{1}{K_L(s_i)} - q_i) \cdot sin(s_i K_L(s_i)) \qquad y_i = \frac{1}{K_L(s_i)} - (\frac{1}{K_L(s_i)} - q_i) \cdot cos(s_i K_L(s_i)).$$

If q_i, p_i are constant and $K_L(s_i)$ is instantaneously constant at s_i, the first derivatives of the Cartesian coordinates are:

$$\dot{x_i} = \dot{s_i}(1 - q_i K_L(s_i)) \cdot cos(s_i K_L(s_i)) \qquad \dot{y_i} = \dot{s_i}(1 - q_i K_L(s_i)) \cdot sin(s_i K_L(s_i))$$

where $\dot{s_i} = \frac{d(d_L+p_i)}{dt} = \dot{d_L} = u_L(d_L)$. Thus:

$$u_i(s_i) = \sqrt{\dot{x}^2 + \dot{y}^2} = u_L(d_L)(1 - q_i K_L(s_i)) \tag{3}$$

Note that $u_L(d_L)$ is the current velocity of the leader at d_L but $K_L(s_i)$ is defined as the projection of the follower's position on the leader's trajectory (i.e., the curvature of the leader at distance p_i along its trajectory). To compute the follower's curvature, it is necessary to compute the second order derivatives of the Cartesian coordinates, since $K = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2+\dot{y}^2)^{3/2}}$, which results in:

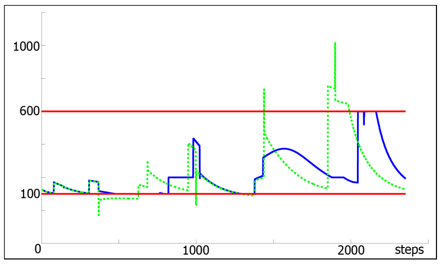
$$K_i(s_i) = \frac{K_L(s_i)}{1 - q_i K_L(s_i)} \tag{4}$$

The advantage of considering curvilinear coordinates is that it is possible to exactly compute Eqs. 3, 4 that provide the controls of the followers as functions of the reference agent's controls. The follower requires access only to the past leader curvature $K_L(s_i)$ and current velocity $u_L(d_L)$. For $K_L(s_i)$ to be known, it has to be that $p_i < 0$, which is the reason why the reference agent is selected to be ahead of every other aircraft. Otherwise, a follower requires the curvature of the reference agent into the future. When the leader's trajectory is computed in real-time, this is not available.

Using Eqs. 3 and 4 it is possible to maintain a perfect formation, such as the triangular one implied by Fig. 2, when the leader moves along a straight line. When the leader turns, the Cartesian distances between the aircraft change and the formation becomes flexible. The aircraft "outside" of the leader's turn have lower curvature and higher velocity. Aircraft "inside" of the leader's turn will have higher curvature and lower velocity.

3.2 Respecting the Control Limits

There is an important issue that arises with the above approach when the leader's trajectory is computed in real time. It is possible that the controls for the follower computed using Eqs. 3 and 4 will violate the control limits specified by Eq. 2.



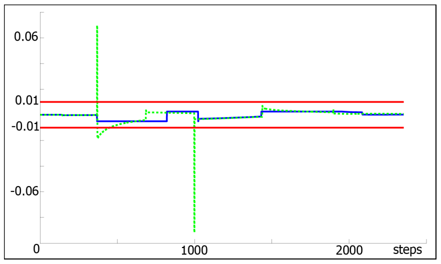


Fig. 3. (left) Velocity of the left follower in Fig. 4, (right) steering angle of same aircraft, (both) green line: the control value when the limits are ignored for the follower, blue: the control value given the proposed fix, red: the limits respected by the leader.

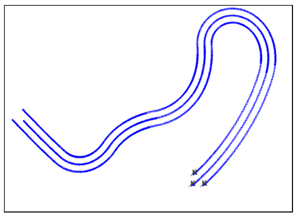


Fig. 4. A benchmark formation trajectory.

For instance, consider the trajectory displayed in Figure 4 for a triangle formation, where the two followers have parameters $p_i = -70$ and $|q_i| = 90$. This trajectory can be achieved by the proposed approach only if the control limits in Eq. 2 are not respected. Fig. 3 shows that for certain control limits respected by the leader, Eqs. 3 and 4 will result in control values for the followers that violate the same limits. The proposed idea in this paper is to compute online limits for the control parameters of the leader so as to guarantee that the result of Eqs. 3 and 4 will never violate the limits of Eq. 2. This is not as straightforward as it appears, since the followers' controls depend on the current velocity of the leader $u_L(d_L)$ and a past curvature value $K_L(s_i)$. Thus, the leader's controls become correlated over time.

Constraints for the leader's curvature Given Eqs. 4 and 2, it has to be true for a follower that:

$$\left|\frac{K_L(s_i)}{1 - q_i K_L(s_i)}\right| \leq K_{max}$$

Consider tentatively only right turns, for which $K > 0$. If the leader turns right using maximum curvature then the right side followers (for which $q_i > 0$) will have to use even greater curvature, which will violate the above constrain. Consequently, for $K > 0$ it has to be that:

$$\max\left(\frac{K_L(s_i)}{1 - q_i K_L(s_i)}\right) \leq K_{max} \quad \Rightarrow \quad \frac{K_L(s_i)}{1 - \max(q_i) K_L(s_i)} \leq K_{max}$$

Let's denote the maximum positive q_i value as: $q^+_{max} = \max(q_i)$, then:

- if $(1 - q^+_{max} K_L(s_i)) > 0$ then: $K_L(s_i) \leq \frac{K_{max}}{1+q^+_{max} K_{max}}$
- if $(1 - q^+_{max} K_L(s_i)) < 0$ then: $K_L(s_i) \geq K_{max} \cdot (1 - q^+_{max} K_L(s_i))$

But for the second case, it has been assumed that $K_L(s_i) > 0$, $K_{max} > 0$ and $(1 - q^+_{max} K_L(s_i)) < 0$. Consequently, the second equation is always true since $K_L(s_i)$ is greater than something negative. Thus, only the first case imposes a constraint on the leader's curvature. For the case that the leader turns left, the computations are symmetrical. If $q^-_{max} = \min(q_i)$ denotes the maximum negative value for q_i, the overall constraint for the leader's curvature is:

$$\frac{-K_{max}}{1-q^{-}_{max}K_{max}} \leq K_L(s_i) \leq \frac{K_{max}}{1+q^{+}_{max}K_{max}} \tag{5}$$

Constraints for the leader's velocity given past curvature values Given Eqs. 3 and 2, it has to be true for a follower that:

$$0 \ < \ u_{min} \ \leq \ u_L(d_L)(1-q_iK_L(s_i)) \ \leq \ u_{max}$$

For the lower bound, and given that $u_L(d_L) > 0$, it has to be true that:

$$\min_i(u_L(d_L)(1-q_iK_L(s_i))) \geq u_{min} \quad \Rightarrow \quad u_L(d_L)(1-\max_i(q_iK_L(s_i))) \geq u_{min}$$

Then there are two cases:

- If $(1-\max_i(q_iK_L(s_i))) > 0$ then: $u_L(d_L) \geq \frac{u_{min}}{1-\max_i(q_iK_L(s_i))}$
- If $(1-\max_i(q_iK_l(s_i))) < 0$ then: $u_L(d_L) \leq \frac{u_{min}}{1-\max_i(q_iK_L(s_i))}$

The second case implies that $u_L(d_L)$ must be less than or equal to something negative. This should never be true, however, for the aircraft. For that reason, and in order to guarantee that $(1-\max_i(q_iK_L(s_i))) > 0$, it has to be that:

$$|K_L(s_i)| < \frac{1}{\max_i(|q_i|)} \tag{6}$$

This means that a follower cannot be further away from the leader along the perpendicular direction than the pivot point defined by the maximum curvature value. Otherwise, a follower has to move backwards.

For the upper bound, and given that $u_L(d_L) > 0$, it has to be true that:

$$\max_i(u_L(d_L)(1-q_iK_L(s_i))) \leq u_{max} \quad \Rightarrow \quad u_L(d_L)(1-\min_i(q_iK_L(s_i))) \leq u_{max}$$

Due to Eq. 6, it is true that $1-\min_i(||q_iK_l(s_i)||) > 0$. Then this implies: $u_L(d_L) \leq \frac{u_{max}}{1-\min_i(q_iK_L(s_i))}$

By combining the lower bound and the upper bound, it is possible to define the constraint for the leader's velocity:

$$\frac{u_{min}}{1-\max_i(q_iK_L(s_i))} \ \leq \ u_L(d_L) \ \leq \ \frac{u_{max}}{1-\min_i(q_iK_L(s_i))} \tag{7}$$

Constraints for the leader's curvature to allow future velocities Notice that it is possible the lower bound in Eq. 7 to become greater than the upper bound. This means that given past curvatures choices at different points in time, which depend on the p_i parameters of the followers, it is possible at some point the leader not to have any valid velocity to choose from, given Eq. 7. To guarantee that this will never happen it has to be always true that:

$$\frac{u_{min}}{1-\max_i(q_iK_L(s_i))} \ < \ \frac{u_{max}}{1-\min_i(q_iK_L(s_i))}$$

The minimum upper bound is achieved for the maximum value of $1-\min_i(q_iK_L(s_i))$. But given Eq. 6: $max\{1-min_i(q_iK_L(s_i))\} = 2$, thus the upper bound in Eq. 7 is at least $\frac{u_{max}}{2}$. Thus it is sufficient to guarantee that:

$$\frac{u_{min}}{1-\max_i(|q_iK_L(s_i)|)} < \frac{u_{max}}{2} \ \Rightarrow \ \max_i(|q_iK_L(s_i)|) \leq \frac{u_{max}-2u_{min}}{u_{max}} \ \Rightarrow$$

$$|K_L(s_i)| \leq \frac{u_{max}-2u_{min}}{u_{max}\max_i(|q_i|)} \tag{8}$$

Given the above equation, it has to be that $u_{max} \geq 2u_{min}$ to guarantee the existence of a valid curvature. Overall, if the leader satisfies Eqs. 5, 6, 7 and 8, then followers that execute Eqs. 3 and 4 will always satisfy Eq. 2.

3.3 Dynamic formations

Allowing the shape of the formation to change over time might be desirable for multiple reasons. For instance, the user might want to change the formation or the team might have to fit through a small opening. Dynamic formations can also be useful for assembling a static formation from a random configuration. To achieve dynamic formations, the approach allows the coordinates $[p_i,\ q_i]$ to be functions of time or distance along the leader's trajectory. Following the derivation in the original work on formations along curvilinear coordinates [3] the controls for the followers in the case of dynamic formations are:

$$u_i = Qu_L(d_L) \tag{9}$$

$$K_i = \frac{1}{Q}(K_L(s_i) + \frac{(1 - q_i(s_i)K_L(s_i))(dq_i^2/ds_i^2) + K_L(s_i)(dq_i/ds_i)^2}{Q^2}) \tag{10}$$

where $Q = \sqrt{(\frac{dq_i}{ds_i})^2 + (1 - q_i(s_i)K_L(s_i))^2}$. Once the functions $q_i(s_i)$, $\frac{dq_i}{ds_i}(s_i)$ and $\frac{dq_i^2}{ds_i^2}(s_i)$ are defined for the above controllers, it is possible to use them together with any reference trajectory. For example, these functions can be defined so as to change the formation from a square to a line, magnify the current shape or even permute positions within the same formation [3].

The control limits must still be satisfied during the execution of dynamic formations. Note that in the case of static formations, the limits on the leader's curvature (Eqs. 5, 6 and 8) could be computed once given the value $max_i|q_i|$ of the static formation. For the dynamic formation, q_i can be dynamic and the approach has to update the constraint for the leader's curvature online by using the $q_i(s_i)$ function for the followers at that point in time. A conservative approximation for the dynamic formation is to consider the maximum q_i achieved by any of the followers during the execution of the entire dynamic formation ($max_i|q_i(s_i)|,\ \forall s_i$). Then the constraint for the leader's velocity (Eq. 7) is recomputed every iteration as in the case of a static formation but for the latest value of the $q_i(s_i)$.

4 Implementation

In order to evaluate the proposed approach, a game engine developed at the University of Nevada, Reno for educational purposes was utilized. The engine is built over the Python-Ogre platform using Python as the programming language. This engine provides the opportunity to select aircraft and direct them to a desired goal. As is often the case in game engines, the engine repetitively calls a function that executes the necessary operations for physics, AI and rendering that models a constant amount of simulation time dt. Each aircraft is modeled as a different entity and has a specific amount of time to complete its computations.

Recall that equations 3 and 4 require the follower to utilize the leader's current velocity $u_L(d_L)$ together with the leader's past curvature $K_L(s_i)$. For static formations, during simulation step dt, the follower's projection along the leader's trajectory must move the same amount as the leader does, so as to retain a constant curvilinear distance p_i. This means that the duration for which a specific curvature value followed by the leader in the past is not going to be the same as the duration that the follower will use it to compute its own curvature. This is because the two agents are moving with different velocities during the corresponding time windows. The correct duration for which a curvature is going to be followed can be computed by finding the distance for which the leader executed the same trajectory and dividing with the current velocity of the follower. It might happen that during a single simulation step dt, the follower will have to switch multiple controls if the leader is moving faster.

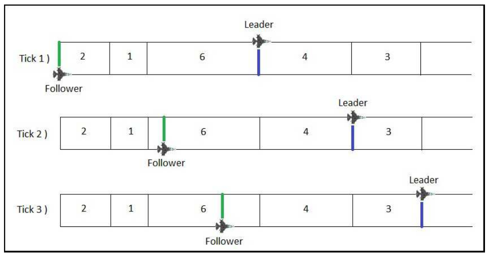


Fig. 5. A follower may be forced to switch controls during a single simulation step.

Fig. 5 illustrates this issue. Assume the leader and the follower start at the positions shown in Fig.5(top). Each box represents the time window that the leader was using a specific set of controls. The numbers are the distances that the leader covered within each time window. For instance, the leader is about to cover a distance of 4 units. In a static formation, the same distance must be followed by the follower's projection along the leader's trajectory during the same window. But in the past and for the same distance of 4 units, the leader had switched three different curvature values (for distances 2, 1 and 6). The duration for which the follower will execute the n_{th} control is $t_n = box\_distance_n / u_L$, where $u_L(t)$ is the leader's current velocity. For the consecutive step, it happens that the follower will execute a single control, since the leader has currently slowed down compared to its past velocity.

5 Results

Variety of Formations: The proposed approach is able to accommodate a large variety of geometric configurations for formations. The only limitation is due to Eq. 6, which means that in order to allow non-zero values for the leader's curvature, the max $(|q_i|)$ value has to be limited. For reasonable values of curvature constraints and size of aircraft, the limitation on $max(|q_i|)$ is such that allows followers in a considerable distance from the leader relative to the aircraft size. Fig. 6 provides examples of formations achieved.

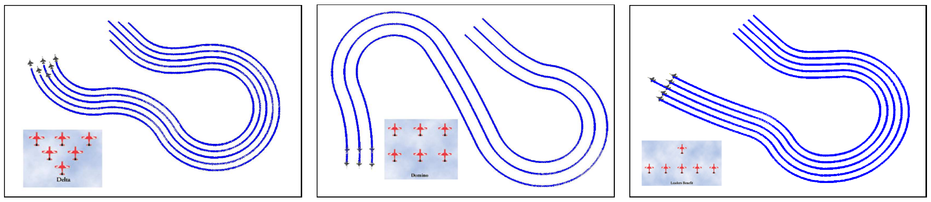


Fig. 6. Check: http://www.raaf.gov.au/roulettes/images/formations.jpg for formations tested. (left) Delta type. (middle) Domino type. (right) Leader Benefit type.

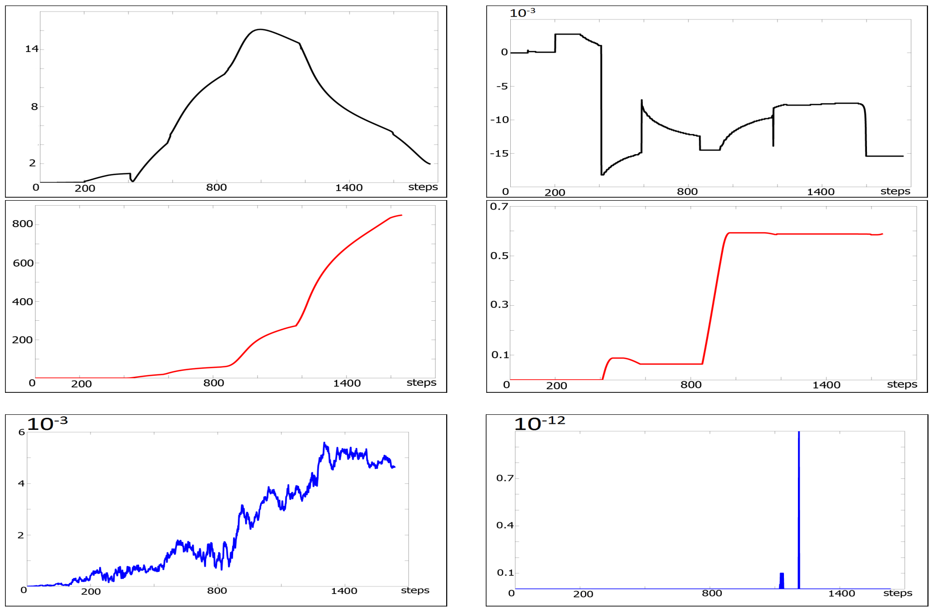


Fig. 7. These figures evaluate the errors for: (top) version 1, (middle) version 2, (bottom) version 3. (left) the error in Cartesian coordinates (unit is pixels) between the followed path and the correct path, (right) Error in orientation (unit is radians).

Errors in Formation Maintenance: The formation in Figure 4 for $p = -60$ and $|q| = 50$ was also used to evaluate the amount of error in the achieved formation. The following results considered three versions for testing formations.

- In version 1 the aircraft do not respect control limits, so they can follow any formation. The implementation employs a constant control for the follower during each simulation step using the leader's current velocity and the leader's past curvature at the beginning of the step. Thus, this step ignores the discussion in section 4.
- Version 2 properly implements the time window approach described in Section 4 but the aircraft respect control limits. The leader respects only the original limits of Eq. 2 but not the limits proposed in Section 3.
- Version 3 corresponds to the proposed approach.

Figure 7 provides the error between the desired formation and the achieved formation at each simulation step for the trajectory in Fig. 4. Similar results have been acquired for hundreds of trajectories. If the steps in Section 4 are not followed or the limits on the leader's controls are not respected, the formation cannot be followed and the error increases considerably. For the proposed approach, the error remains well below 10^{-2} pixels for many minutes of simulation.

Effects of Control Limits: Figure 8 studies how constrained a team of aircraft becomes when it has to respect the control limits specified in this paper. The left image shows the velocity limits for the leader for the trajectory in Fig. 4 and shows that still a large portion of the velocity parameters is valid but care

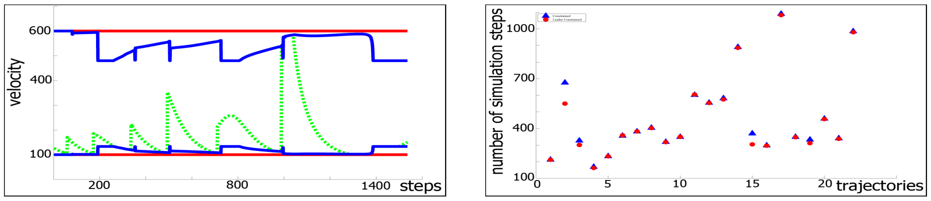


Fig. 8. (left) Red lines: general velocity limits for the aircraft, Blue: velocity limits for the leader that guarantee formation maintenance, Green: actual velocity of the leader (right) # of steps to reach a destination for aircraft formations that don't respect control limits (red dots) and aircraft that do (blue triangles).

has to be taken for the extreme values. The right image shows how much time it takes for the triangular formation to reach various random targets if it is not constrained or constrained. Sometimes, the more physically realistic, constrained formation has to follow a longer trajectory to reach a target due to the introduction of the constraints for the leader. But the difference is not significant and does not justify not respecting the control limits during the simulation process.
Dynamic Formations: Figure 9 shows the extension of the approach to the case of dynamic formations.

6 Discussion

This paper has outlined a framework for simulating planar formations along curvilinear coordinates for systems with non-holonomic constraints. A previous method for mobile robots [3] has been extended to handle systems with curvature limits and minimum velocity. The implementation employs a PID controller for computing the controls of the reference agent but allows for the user to directly control one of the aircraft or to assign a mobile target to the formation. Future work includes integration of the formation algorithm with path planners and reactive methods (e.g., [21]) to get formations that avoid static (e.g., mountains) or dynamic obstacles (e.g., missiles or other aircraft). A 3D version is of significant interest for aircraft formations, as well as an extension to higher-order aircraft to allow for continuous or smooth velocity.

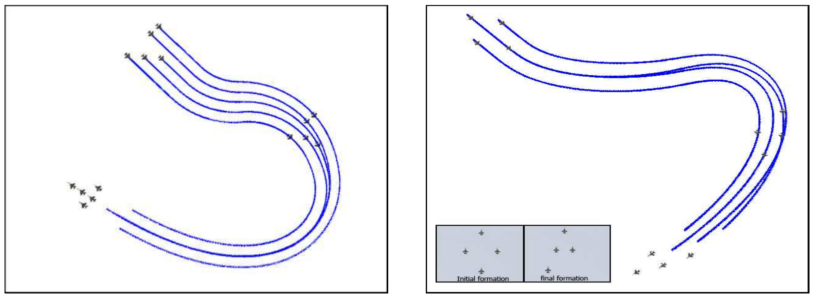


Fig. 9. Dynamic formations along curved trajectories for five and four airplanes.

References

1. Balch, T., Arkin, R.: Behavior-based formation control for multi-robot teams. IEEE Transactions on Robotics and Automation 14(6), 926–939 (Dec 1998)
2. Balch, T., Hybinette, M.: Social potentials for scalable multi-robot formations. In: IEEE International Conference on Robotics and Automation. pp. 73–80 (Apr 2000)
3. Barfoot, T.D., Clark, C.M.: Motion planning for formations of mobile robots. Robotics and Autonomous Systems 46(2), 65–78 (February 2004)
4. van den Berg, J., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal n-body collision avoidance. In: International Symposium on Robotics Research (ISRR) (Sep 2009)
5. Bottesi, G., Laumond, J.P., Fleury, S.: A motion planning based video game. Tech. rep., LAAS CNRS (Oct 2004)
6. Brogan, D.C., Hodgins, J.K.: Group behaviors for systems with significant dynamics. Autonomous Robots 4, 137–153 (1997)
7. Desai, J., Ostrowski, J., Kumar, V.J.: Modeling and control of formations of nonholonomic mobile robots. Transactions on Robotics and Aut. 17(6), 905–908 (2001)
8. Ennis, C., Peters, C., O'Sullivan, C.: Perceptual evaluation of position and orientation context rules for pedestrian formations. In: Symposium on Applied Perception in Graphics and Visualization (APGV'08). pp. 75–82 (2008)
9. Fierro, R., Belta, C., Desai, K., Kumar, V.J.: On controlling aircraft formations. In: IEEE Conf. on Decision and Control. pp. 1065–1070. Orlando, FL (Dec 2001)
10. Fredslund, J., Mataric, M.J.: A general, local algorithm for robot formations. IEEE Transactions on Robotics and Automation 18(5), 873–846 (Oct 2002)
11. Gayle, R., Moss, W., Lin, M.C., Manocha, D.: Multi-robot coordination using generalized social potential fields. In: ICRA. pp. 3695–3702. Kobe, Japan (2009)
12. Geraerts, R., Kamphuis, A., Karamouzas, I., Overmars, M.: Using the corridor map method for path planning for a large number of characters. In: 1st Intern. Workshop on Motion in Games (MIG). Utrecht, The Netherlands (June 2008)
13. Jadbabaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile agents using nearest neighbor rules. IEEE Trans. on Automatic Control 8(6), 988–1001 (2003)
14. Kloder, S., Hutchinson, S.: Path planning for permutation-invariant multirobot formations. IEEE Transactions on Robotics 22(4), 650–665 (2006)
15. Lau, M., Kuffner, J.: Behavior planning for character animation. In: ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA) (2005)
16. Lewis, M., Tan, K.H.: High-precision formation control of mobile robotis using virtual structures. Autonomous Robots 4(4), 387–403 (Oct 1997)
17. Lien, J.M., Rodriguez, S., Malric, J.P., Amato, N.: Shepherding behaviors with multiple shepherds. In: International Conf. on Robotics and Automation (2005)
18. Michael, N., Zavlanos, M.M., Kumar, V., Pappas, G.J.: Distributed multi-robot task assignment and formation control. In: ICRA. pp. 128–133 (May 2008)
19. Patil, S., van den Berg, J., Curtis, S., Lin, M., Manocha, D.: Directing crowd simulations using navigation fields. IEEE Transactions on Visualization and Computer Graphics (TVCG) (2010)
20. Pelechano, N., Allbeck, J., Badler, N.: Controlling individual agents in high-density crowd simulation. In: ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA). vol. 3, pp. 99–108. San Diego, CA (August 3-4 2007)
21. Snape, J., Manocha, D.: Navigating multiple simple-airplanes in 3d workspace. In: IEEE International Conference on Robotics and Automation (ICRA) (2010)
22. Stylianou, S., Chrysanthou, Y.: Crowd self organization, streaming and short path smoothing. In: Computer Graphics, Visualization and Computer Vision (2006)
23. Tabuada, P., Pappas, G.J., Lima, P.: Motion feasibility of multi-agent formations. IEEE Transactions on Robotics 21(3), 387–392 (2005)
24. Takahashi, S., Yoshida, K., Kwon, T., Lee, K.H., Lee, J., Shin, S.Y.: Spectral-based group formation control. Comp. Graph. Forum: Eurographics 28, 639–648 (2009)
25. Tanner, H.G., Pappas, G.J., Kumar, V.J.: Leader-to-formation stability. In: IEEE Transactions on Robotics and Automation (2004)
26. Vo, C., Harrison, J.F., Lien, J.M.: Behavior-based motion planning for group control. In: Intern. Conf. on Intelligent Robots and Systems. St. Louis, MO (2009)