Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Guidance of Flocks of Vehicles Using Virtual Signposts

Y. Rochefort [*] H. Piet-Lahanier [*] S. Bertrand [*] D. Beauvois [**] D. Dumur [**]

[*] *ONERA - DCPS, 91761 Palaiseau Cedex, France (e-mail: yohan.rochefort, helene.piet-lahanier, sylvain.bertrand at onera.fr).*
[**] *SUPELEC Systems Sciences E3S, Control Department, 91192 Gif-Sur-Yvette Cedex, France (e-mail: dominique.beauvois, didier.dumur at supelec.fr).*

Abstract: In this paper, we present how virtual signposts can be used to guide a flock of autonomous vehicles along a desired path and to a desired location. Signposts can also be used to make the flock avoid obstacles with the interesting feature of indicating the best direction. These virtual signposts allow to define specific desired path without virtual leader constraints. Simply reaching a final location is also possible and both approaches can be mixed if necessary. Stranded vehicles can be put back on the track with carefully placed signposts.

Keywords: Multi-vehicle systems, Autonomous mobile robots, Obstacle avoidance, Distributed control

1. INTRODUCTION

The guidance of flocks of vehicles has driven a large amount of interest since the works of Reynolds (1987). Various civil uses of flocks have been cited: search and rescue, grid of mobile sensors, foraging, parallel transportation of high amounts of vehicles and/or payloads. Military uses can be recognition, surveillance, combat mission.

The control design for a flock of unmanned vehicles with a given mission must fulfil several requirements. First, it must ensure flocking of the fleet while avoiding collisions among vehicles. Second, it must ensure obstacle avoidance. Finally, it must guide the flock along a desired path up to the final location.

Several solutions have been proposed to accomplish flocking and collision avoidance between vehicles. The most widely employed one consists in assigning a potential field to each vehicle: McInnes (1996), Leonard and Fiorelli (2001), Tanner et al. (2003a), Tanner et al. (2003b), Olfati-Saber (2004). Other methods include gyroscopic forces (Chang et al. (2003)), virtual structure (Ren and Beard (2003a)), predictive control (Wang et al. (2007)).

In order to keep the control as simple as possible, the solution proposed here is based on the approach of Vicsek et al. (1995) and Jadbabaie et al. (2002), namely the nearest neighbour rule, with adaptations. One adaptation is the control of the velocity module and direction of the vehicles instead of only their direction. This allows late vehicles to catch up with the flock, or inversely vehicles ahead to wait for the flock. In the original work, the contribution of a vehicle to other vehicles control input is its current direction. It has been extended to take into account its velocity vector and position. This extended contribution allows to regulate the distance between vehicles and is therefore necessary to obtain flocking.

The resulting control law consists in very simple rules that are intended to be decentralized on each vehicle. This solution is close to the work of Inada and Takanobu (2010) but uses progressive transitions between the attraction, imitation, and repulsion areas (instead of a clean border) and the contributions of all vehicles (instead of a limited number of vehicles selected among those within a given radius, with a preferred direction).

The second main issue addressed in multi-vehicles coordination is obstacle avoidance. This differs from avoiding collisions between vehicles because, while vehicles of the flock are likely to share their position and speed, vehicles will have to detect obstacles without their assistance. Moreover, obstacles may present threat different than that of collision which could be harder to detect (like losing communication or GPS signal, being shot down by a hidden installation, or taken in turbulent air flow). Proposed solutions to obstacle avoidance include potential fields (Tanner (2004)), virtual vehicles (Olfati-Saber (2004)), gyroscopic forces (Chang et al. (2003)).

Beside flocking and obstacle avoidance, we aim at driving the flock to a final location. To do that, several guidance methods have been developed, based on different ideas. In Ogren and Leonard (2003), Jadbabaie et al. (2002) the flock follows a leader, which knows the task. Olfati-Saber (2004) uses virtual leaders instead. In Wang et al. (2007), the flock path to a location is calculated using modified Grossberg Neurons Network. Tanner (2004) uses potential fields to bring the flock to the desired location.

The novelty presented in this paper is the use of virtual agents, called (virtual) signposts to jointly guide the flock of vehicles and avoid obstacles. The basic utility of a signpost is to point a direction that vehicles must follow. This can be used to achieve different objectives. Signposts can represent the final location of the flock, attracting it. They can also mark out a desired path allowing complicated trajectory to be defined simply. Finally, they can represent obstacles, thus ensuring avoidance and, when the information is available, point toward a preferred direction for avoidance. Signposts can be set before the mission takes place and/or defined during the flight as new information on the environment becomes available.

In the next section, we expose the full problem we want to address, considering the navigation of unmanned vehicles in two dimensions only. Section 3 is devoted to a brief explanation of the interaction model we use. Section 4 explains the main properties of virtual signposts. Numerical simulations are presented in section 5. Finally we conclude this paper in section 6 with comments and future work.

2. PROBLEM STATEMENT

We consider a system composed of N agents, moving on a plane. This can represent either a group of ground vehicles or a flock of aerial vehicles flying at a constant altitude. For simplicity, we use a simple discrete integrator as kinematic model for our agents:

$$r_i(k+1) = r_i(k) + \Delta t.u_i(k) \tag{1}$$

where $r_i(k)$ is the position vector of agent i at step k, $u_i(k)$ is the control input of agent i (i.e. a velocity vector reference) computed with information available at step k. Δt is the time interval between two updates. In the following, we will omit the index k to lighten the expressions. Relative positions between agents i and j ($j \neq i$) are noted $r_{ij} = r_i - r_j$.

The pursued objective is to guide the N agents in the environment to accomplish a given mission. The mission requires the agents to form and travel as a flock when possible. The desired movements of the flock can be described as a predefined path to be followed and/or a final location to be reached. The flock must avoid obstacles that are present in their path. Additionally, each agent computes its own control input. For that purpose, the positions and velocity vectors of the other agents are available. This means that agents must communicate this information to each other and that their positions must be known in a common frame.

In any case we do not look for a precise flock geometry, we want the agents to travel as a flock, fish-like (i.e. there is no isolated agent, nor collision) and at a distance of around D from their closest neighbours (in terms of euclidean norm).

3. INTERACTION MODEL

Our model of interaction between agents is based on the three Reynolds (1987) rules: 1. Collision avoidance, 2. Velocity matching, 3. Flock centering. Our implementation is inspired, with large adaptations, by the nearest neighbour rule from Vicsek et al. (1995).

To present the model, we explain how agent i control input is computed. This computation is divided in three steps: computation of control input; computation of each agent contribution; determination of each agent influence.

3.1 Control input computation

The computation of agent i control input is a simple weighted mean of the contributions of all agents (including agent i itself). The contribution of agent j in agent i control input is associated to the movement that agent i should do to form a flock with agent j (see section 3.2). The weighting factors are set to reflect the importance of agent j contribution (see section 3.3).

The control input of agent i is given by (2), (3), and (4).

$$u_i = v_i. \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix} \tag{2}$$

$$v_i = \sum_{j=1}^{N} \beta_{ij}.v_{ij} \Bigg/ \sum_{j=1}^{N} \beta_{ij} \tag{3}$$

$$\theta_i = \arctan\left(\sum_{j=1}^{N} \beta_{ij}.\sin(\theta_{ij}) \Bigg/ \sum_{j=1}^{N} \beta_{ij}.\cos(\theta_{ij}) \right) \tag{4}$$

where v_i and θ_i are the module and direction of u_i, the control input of agent i; v_{ij} and θ_{ij} are the module and direction of agent j contribution to u_i, β_{ij} is the weight of this contribution.

3.2 Agent contribution

The contribution of agent j in agent i ($j \neq i$) control input is the velocity vector that agent i should adopt to produce a behaviour that will allow flocking. This contribution is determined by considering that agents i and j are the only agents. Depending on the distance between the two agents, three basic behaviours (corresponding to the three Reynolds rules) are possible: 1. repulsion when the two agents are too close (i.e. $\|r_{ij}\| < D$); 2. imitation when the agents are at a desired distance (i.e. $\|r_{ij}\| = D$); 3. attraction when the agents are too far (i.e. $\|r_{ij}\| > D$).

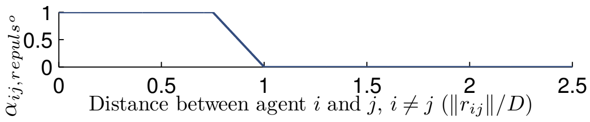
To produce these behaviours, agent i can control both module and direction of its velocity vector. Table 1 shows how the module and direction of agents' velocity vector must be defined, depending on the desired behavior. It details also how these components change with the relative position of the agents (as an example, agent i must accelerate if agent j is too far ahead, but slow down if it is too far behind).

To achieve smooth movements of agents, contributions are not limited to these three basic behaviours. Instead, contributions are computed by a weighted mean of the three basic behaviours, with weight of each basic behaviour reflecting the need for the behaviour. For example, when two agents are too close to each other, there is no need for

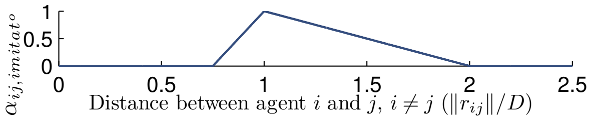
Table 1. Basic behaviours contribution

desired behaviour (b)	agent j is ... agent i	agent i velocity vector	
		module ($v_{ij,b}$)	direction ($\theta_{ij,b}$)
repulsion	ahead	slowest [i]	steer away [iii]
	behind	fastest [ii]	
imitation	whichever	same as j	same as j
attraction	ahead	fastest [ii]	steer toward [iv]
	behind	slowest [i]	

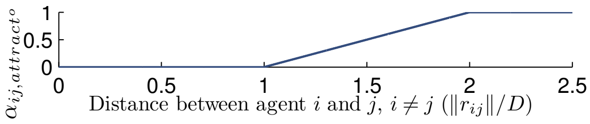
[i]slowest allowed | [iii]repulsion direction is $\arg(r_{ij})$
[ii]fastest allowed | [iv]attraction direction is $\arg(r_{ij}) + \pi$
$\arg(r_{ij})$ stands for the direction of vector r_{ij}



(a) Repulsion function



(b) Imitation function



(c) Attraction function

Fig. 1. Spacing functions used in our model

attraction. The need for repulsion and imitation depends on how close the agents are. The closer they are, the more repulsion and the less imitation are needed. Figure 1 shows the spacing functions α that give the weight of each basic behaviour according to the distance between agents in our model. These functions put a stronger emphasis on repulsion effect by having a shorter repulsion/imitation transition comparing to the imitation/attraction one.

By convention, the contribution of an agent to its own control input is plain imitation: it would make no sense if an agent attracts or repulses itself. Consequently, spacing functions are set to: $\alpha_{ii,attra} = \alpha_{ii,repul} = 0$; $\alpha_{ii,imita} = 1$.

The two components of agent j contribution to agent i control input (u_i) are computed separately by (5) and (6).

$$v_{ij} = \sum_{b\in B} \alpha_{ij,b}.v_{ij,b} \Big/ \sum_{b\in B} \alpha_{ij,b} \quad (5)$$

$$\theta_{ij} = \arctan\left(\frac{\sum_{b\in B} \alpha_{ij,b}.\sin(\theta_{ij,b})}{\sum_{b\in B} \alpha_{ij,b}.\cos(\theta_{ij,b})}\right) \quad (6)$$

where v_{ij} and θ_{ij} are the module and direction of agent j contribution to u_i; B is the set of basic behaviours :{repulsion, imitation, attraction}; $v_{ij,b}$ and $\theta_{ij,b}$, ($b \in B$), are the velocity module and direction associated with the behaviour b as defined in table 1; $\alpha_{ij,b}$ is the weight of this basic behaviour in the computation of the contribution.

3.3 Influence of agents

The contribution of an agent j in agent i control input is computed as if agent i and j were alone. However there are many other agents in the flock to consider. If all agents were given the same importance in agent i control input, then the contributions of an agent too close (repulsing) and an agent too far (attracting) could (partly or fully) compensate each other: agent i would not even try to avoid collision. To give more importance to collision avoidance, nearby agents are given the most importance and remote agents are ignored. Agents that are in between (close, but not *too* close) are not ignored (there would be no flocking behaviour) but are given less importance.

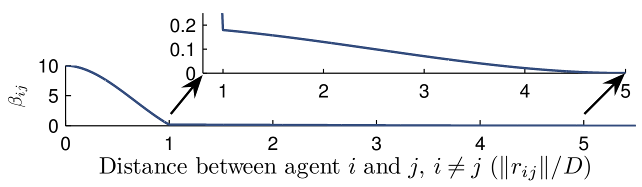
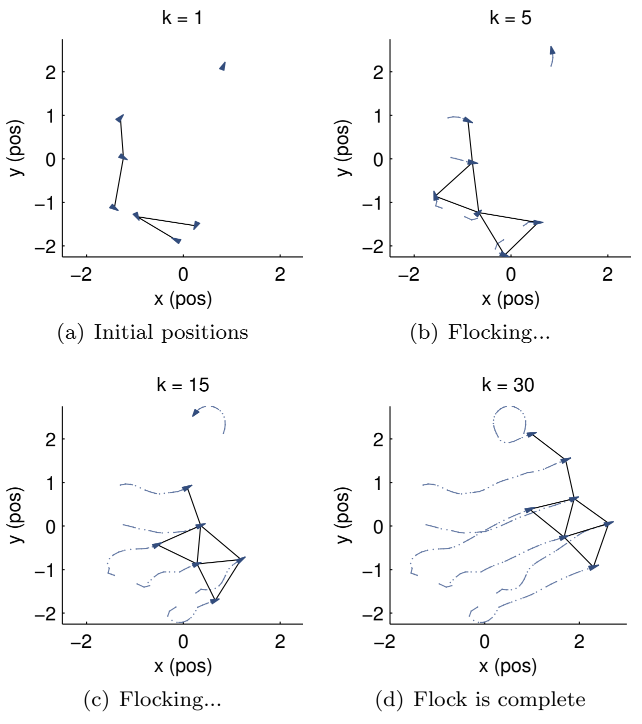


Fig. 2. Influence function used in our model

We define the influence of agent j as the scalar β_{ij} that reflects the importance to give to agent j contribution in the computation of agent i control input. In our work, this scalar depends only on the distance between the two agents. Figure 2 shows the influence function that is used in our model. It is based on the sum of two third degree polynomial functions: one for the 'pic' and one for the base influence. By convention, the influence of an agent on its own control input is 1, this provides a reference point to define other influence functions. Other methods of influence determination could be added, like the distinction ahead/behind.

Figure 3 shows the successful flocking behaviour of 7 agents following the presented interaction model. The desired distance D between agents is used as the unit of length. The simulated time update is used as the unit of time. Initial positions of agents are set randomly in a square of side 5 centred around the origin. Their initial velocity direction is set randomly between $[-\pi\ \pi]$ and initial velocity module is 0.1. The path followed by each agent is shown by a dashed line. A link is drawn between two agents when they are at an acceptable distance one from the other (i.e. inside a neighbourhood of D).



(a) Initial positions (b) Flocking... (c) Flocking... (d) Flock is complete

Fig. 3. Demonstration of flocking

4. VIRTUAL SIGNPOSTS OVERVIEW

The guidance of the flock and the avoidance of obstacles are assured by virtual agents, also called signposts. Regular agents consider signposts as if they were regular agents, i.e. they are added to the list of agents when computing control input of regular agents with equations 3 and 4 (which are left unchanged). Equations 5 and 6 also apply on the signposts.

For these purposes, signposts have the exact same properties as regular agents : position, velocity vector, influence and spacing functions (although these are differently shaped). Note that signposts' velocity vectors are used to influence agents and do not represent their actual motion.

4.1 Management and placement of signposts

As they do not represent real agents, the management of signposts is a task in itself. The simplest way is to fix the list of signposts and their properties before the beginning of the mission. A copy of the list is transmitted to each agent to be used for navigation. This method however does not allow new information to be transmitted to agent while the mission progresses.

To allow new information (changes in the mission, agents' discoveries) to be integrated, agents can use a dynamical list of signposts. This list will be updated with new information when available.

This list can either be maintained by a particular agent (in or outside the flock) which transmits it to the others, or managed in a collective manner. In the second case, each agent keeps a local copy of the list and collaborates with the others to update it. The work of Ren and Beard (2003b) could be a starting point to implement this.

The use of a dynamical list will increase the need for communication between agents, but it will allow them to share information about their discoveries.

4.2 Various uses of signposts

Implementation of signposts can be done with different objectives: attract agents to the final location, guide agents through a path or avoid obstacles. The common point of all types of signposts is to show a direction to follow with a specific velocity module.

Final location – A signpost can be used as a final location. Agents are only attracted by this signpost (i.e. repulsion and imitation functions are null, attraction function is constant equal to 1). Influence of this signpost must be designed with caution, to attract the flock without interfering with other agents contributions (real or not). We use a constant influence function with a value of 1.

Path guidance – Signposts can be used to guide the flock along a desired path. Agents only imitate these signposts (i.e. repulsion and attraction functions are null, imitation is constant equal to 1). Influence of these signposts must be local to put the flock on the right path and "pass it" to the next signpost. Their influence must be high enough to influence the flock and prime over a "final location" signpost. We propose a polynomial based function as in figure 4.

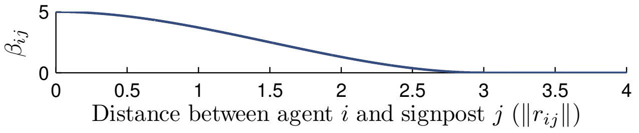


Fig. 4. Influence function used for path guidance

Obstacle avoidance – Signposts can be used to point out the presence of obstacles. Agents are not attracted by these signposts, but repulsed if they become too close. The operator designing the signposts can choose whether or not to use imitation when the flock meets obstacles. While not necessary, it offers an interesting feature: the possibility to tell agents in which direction they should go while avoiding the obstacle and if they should slow down or not. The influence function of these signposts is designed to give to obstacle avoidance a importance greater than anything else. We propose to use the same function as in figure 2 with an amplitude of 100. The horizontal extend must reflect the size of the obstacle to avoid.

5. NUMERICAL SIMULATION

In this section, we present simulations that illustrate the various possibilities offered by virtual signposts. The properties of the signposts employed in these simulations are those proposed in section 4.2. The simulations were conducted for a system of 7 agents. The desired distance D between agents is used as the unit of length. The simulated time update is used as the unit of time.

Initially, agents are randomly placed in a square of side 5 units and centred around the origin. Initial velocity directions of agents are also randomly chosen, their initial velocity module is fixed to 0.1. Agents can take any direction and speed in one iteration, with allowed speed being in [0.05 0.2] in our simulations.

5.1 Final location

To show the effect of a "final location" signpost, we initially let the agents move with no external reference. Then, at step $k = 50$ we add a signpost indicating the final location. Figure 5(a) shows the flock of agents before the signpost addition. Figure 5(b) shows how the flock has manoeuvred to take the direction of that new signpost. Except for the addition of the signpost, nothing has been changed between the two snapshots.

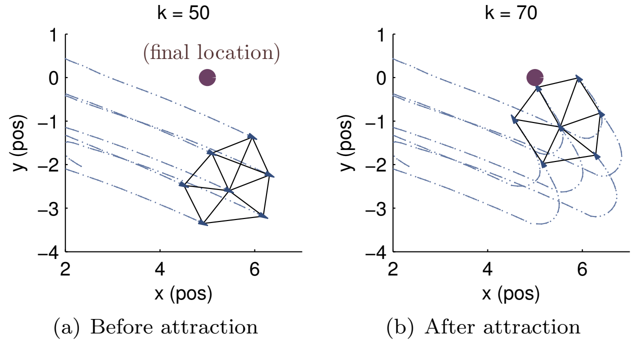


Fig. 5. Attract the flock to the final location (circle)

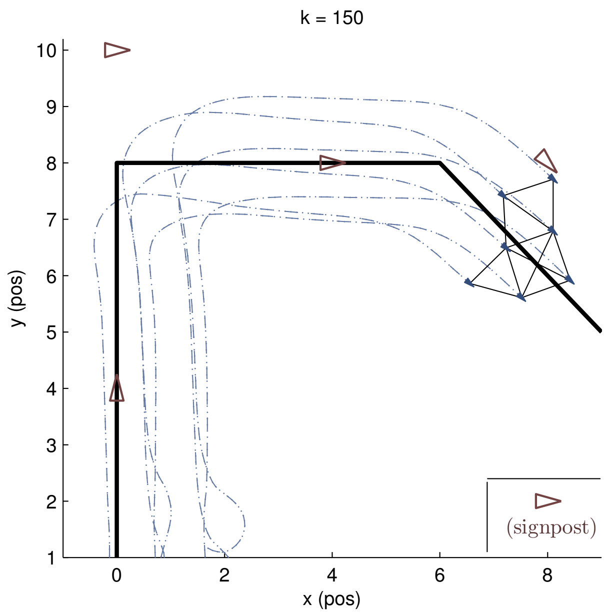


Fig. 6. Guide the flock along a path

5.2 Path guidance

In figure 6, signposts are used to guide the flock along the path represented by the solid straight line. Signposts have an influence radius of 3. No final location exists here.

Two signposts are shifted from the path. Without these shifts, the flock would have turned too early because of the early high influence of the signposts. A good knowledge of the structure and dynamics of the flock is necessary to tune the influence areas. This tuning can be done by moving the signposts or changing the influence functions.

5.3 Obstacle avoidance

Figure 7 shows the ability of the flock to split in order to avoid an obstacle and rejoin once it is passed. The circle represented is not the border of the obstacle (which may be unknown) but the beginning of the repulsion area. It is then normal that agents enter this area. A "final location" signpost is used in this simulation at coordinates [20 0].

When the flock must not split, the use of virtual signposts offers an interesting feature. Using the direction of its velocity vector, the signpost can inform the flock on the best side to proceed to avoid the obstacle. This way, agents do not need to be involved in high level decision making concerning the direction to take. Figure 8 illustrates the behaviour of the flock in this circumstance. The obstacle to avoid and the objective are the same as in figure 7. The position and speed of the agents are also the same before the manoeuvre. Only the direction of the velocity vector of the signpost has changed to allow the set of agents to remain a flock.

Figure 9 shows a complete scenario where a flock of 25 agents is confronted to all types of signposts.

6. CONCLUSION

In this paper, we proposed a new way to guide flocks of vehicles and ensure obstacle avoidance in the form of

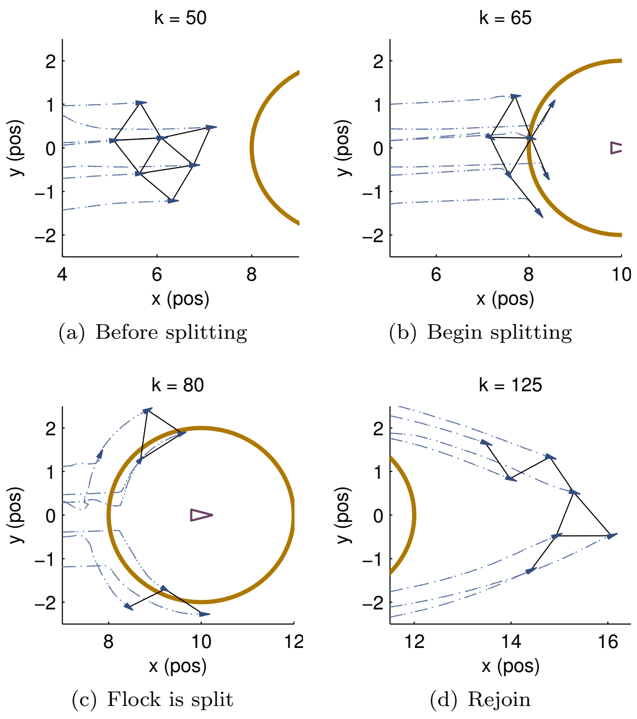


Fig. 7. Avoidance with splitting and rejoin manoeuvre

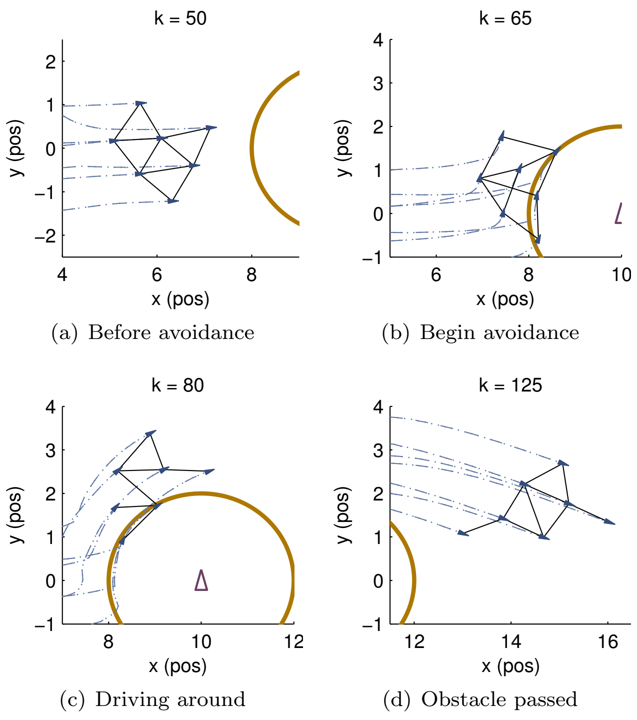


Fig. 8. Avoidance without splitting

virtual signposts. These signposts interact with agents in the same way as an agent of the flock but do not move. They are used to point agents to the right direction, allowing the flock to 1. reach a final location indicated by a signpost; 2. follow a path marked out with signposts; 3. avoid obstacles represented by signposts. The most interesting feature of signposts is that they allow to inform the flock about a preferred side to avoid an obstacle. This is achieved by assigning a desired avoidance direction to

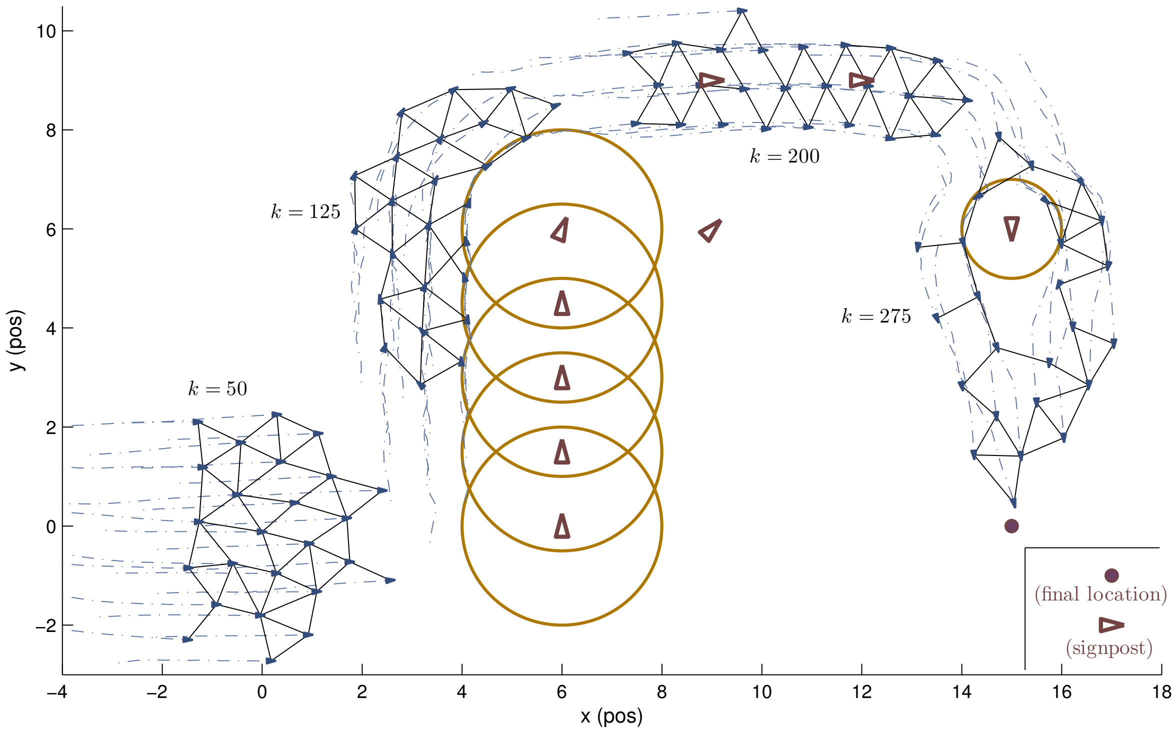


Fig. 9. Example of a full scenario with $N = 25$ agents at different steps

the velocity vector of the signpost. This feature is useful in cases where the flock must not split.

The use of signposts seems worth investigating further. Specially, the possibility to define dynamically signposts as new information becomes available, can be used by the agents to communicate simply their findings. A task to be addressed to use signposts in the physical world is the management and diffusion of the different signposts properties. Finally, the presented study was in two dimensions only, future work will address the case of vehicles moving in a 3 dimensional space. The robustness to noise, loss of communication and loss of agents will also be studied.

REFERENCES

Chang, D., Shadden, S., Marsden, J., and Olfati-Saber, R. (2003). Collision avoidance for multiple agent systems. In *Proceedings of the 42nd Conference on Decision and Control*, volume 1, 539–543.

Inada, Y. and Takanobu, H. (2010). Flight-formation control of air vehicles based on collective motion control of organisms. In *Proceedings of the 18th IFAC Symposium on Automatic Control in Aerospace, Nara, Japan.*

Jadbabaie, A., Lin, J., and Morse, A. (2002). Coordination of groups of mobile autonomous agents using nearest neighbor rules. In *Proceedings of the 41st Conference on Decision and Control*, volume 3, 2953–2958.

Leonard, N. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th Conference on Decision and Control*, volume 3, 2968–2973.

McInnes, C.R. (1996). Potential function methods for autonomous spacecraft guidance and control. *Advances in the Astronautical Sciences*, 2093–2109.

Ogren, P. and Leonard, N. (2003). Obstacle avoidance in formation. In *Proceedings of the International Conference on Robotics and Automation, 2003*, volume 2, 2492–2497.

Olfati-Saber, R. (2004). Flocking for multi-agent dynamic systems: Algorithms and theory. Technical report, California Institute of Technology.

Ren, W. and Beard, R.W. (2003a). Decentralized scheme for spacecraft formation flying via the virtual structure approach. *AIAA Journal of Guidance, Control, and Dynamics*, 27, 73–82.

Ren, W. and Beard, R.W. (2003b). Dynamic consensus seeking in distributed multi-agent coordinated control. Technical report, BYU MAGICC Lab.

Reynolds, C.W. (1987). Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, 25–34.

Tanner, H. (2004). Flocking with obstacle avoidance in switching networks of interconnected vehicles. In *Proceedings of the International Conference on Robotics and Automation, 2004*, volume 3, 3006–3011.

Tanner, H., Jadbabaie, A., and Pappas, G. (2003a). Stable flocking of mobile agents, part i: Fixed topology. In *Proceedings of the 42nd Conference on Decision and Control*, volume 2, 2010–2015.

Tanner, H., Jadbabaie, A., and Pappas, G. (2003b). Stable flocking of mobile agents part ii: Dynamic topology. In *Proceedings of the 42nd Conference on Decision and Control*, volume 2, 2016–2021.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6), 1226–1229.

Wang, X., Yadav, V., and Balakrishnan, S.N. (2007). Cooperative uav formation flying with obstacle/collision avoidance. *IEEE Transactions on Control Systems Technology*, 15(4), 672–679.