Deliverable D5.5
Assessment of global strategies

Contractual delivery date:

December 2014

Actual delivery date:

January 2015

Partner responsible for the Deliverable:

EPFL-LIS

Authors:

Prof Dario Floreano

Nicolas Dousse

Gaëtan Burri

Dissemination level[1]		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

[1] Dissemination level using one of the following codes: **PU** = Public, **PP** = Restricted to other programme participants (including the Commission Services), **RE** = Restricted to a group specified by the consortium (including the Commission Services), **CO** = Confidential, only for members of the consortium (including the Commission Services)

Document Information Table

Grant agreement no.	ACPO-GA-2010-266470
Project full title	myCopter – Enabling Technologies for Personal Air Transport Systems
Deliverable number	D5.5
Deliverable title	Description and comparison of various global traffic control strategies including formation flying
Nature[2]	R
Dissemination Level	PU
Version	1.0
Work package number	WP 5
Work package leader	EPFL-LIS
Partner responsible for Deliverable	EPFL-LIS
Reviewer(s)	XXX

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 266470.

[2] Nature of the deliverable using one of the following codes: **R** = Report, **P** = Prototype, **D** = Demonstrator, **O** = Other

Revision Table

Version	Date	Modified Page/Section	Author	Comments
1.0	Jan 15	Initial issue	Nicolas Dousse	
		Flocking with individual goals	Gaëtan Burri	inserted by Nicolas Dousse
		Avoiding non-flying zones	Clément Kunz	inserted by Nicolas Dousse

Executive Summary

This report describes the research for tackling the problem of flocking for agents with individual goals as well as navigating through non-flying zones. This deliverable is part of the work package 5. In this work package, it is assumed that due to the high density, the PAVs are automatically controlled and humans are passive passengers.

Literature about collective behavior in nature always assumes that the agents tend to achieve a common goal for the group as whole. For instance, swarm of birds are migrating towards a common location, group of robots can form a sensor network to gather information about a specific location. In a commuting scenario, PAVs have their own defined destination and are egoistic agents that do not want to alter too much their trajectory to join other groups of PAVs. We propose modifications to the standard Reynolds flocking rules that enable natural creation and leaving of flocks of PAVs.

We previously assumed that PAVs were flying in straight lines between their starting position and their destination. However, in a realistic environment, flight-forbidden areas or non-flying zones such as nuclear plants, large airports or military basis are present. The navigation scheme of the PAVs should be adapted in consequence in order to avoid those non-flying zones.

Table of Contents

1. Introduction

In Deliverables 5.3, a strategy to tackle the problem of mid-air collision avoidance for Personal Aerial Vehicles (PAVs) was presented. This strategy was able to avoid collisions between PAVs while meeting user-defined level of comfort.

1.1. Non-flying zones

To maximize the efficiency of the navigation, it was previously assumed that the PAVs were travelling in straight lines from their starting position to their destination. In a realistic scenario, this assumption can no longer hold. Cities, nuclear plants, military basis are examples of places that no one is allowed to flight over. Therefore, a navigation strategy that avoids non-flying zones should be introduced.

1.2. State of the art in collective behavior

Evolution came up with a simple solution to navigate safely in dense environments. In Nature, flocks of birds[1], school of fishes[2], swarms of insects[3], human crowds[4] and bacteria[5] are examples of collective behaviors emerging from local control and local sensing. As stated in [6], it can be observed that large-scale behaviors coming from self-organization process emerge spontaneously from only local interaction and local sensing.

In flocks of birds, we can observe emerging global behaviors from local rules and local knowledge of the environment without any central entity deciding for the whole[3].

Flying in groups by decentralized control strategies can be divided in three main formation structures: the Leader-Follower (or Leader-Wingman) approach[7], [8], the Virtual leader approach[9] and the Behavior approach[10]–[12].

In the Leader-Wingman approach, one agent is defined as being the leader of the formation and all other agents are followers and are required to maintain a given distance and position with respect to the leader and neighbor agents. In such approach, each agent should be able to track precisely their own position as well as leader position and intent in order to maintain a rigid formation. This pseudo-rigidity makes collision avoidance a hard task especially in very dense environments as the formation has to react as a whole and error propagation makes rear agents react poorly to new dynamic threats.

In the Virtual leader approach, each agent tracks the same virtual leader that can either be a real agent or a virtual point. Agents are not required to maintain a relative distance to neighbor agents and thus overcome the drawback of error propagation of the Leader-Wingman approach. Nevertheless, as agents do not track neighbors, they might not be able to avoid collisions. Furthermore, a central entity should designate the leader as well as provide the trajectory of the Virtual Leader leading to communication issues as well as decision processes. The needs of central entity and communication are major drawbacks of such approaches.

Behavior approaches are inspired from Nature where flocks of birds tend to position themselves only by "sensing" local neighbors. Agents tend to position themselves with respect to local neighbors and recreate a given formation geometry. This approach doesn't suffer from single point of failure and is able to cope with dynamic changes of the environment.

[3] Note that this can also be observed in schools of fishes or swarms of insects

Swarm intelligence has been widely used in Robotics. Thanks to scalability and redundancy, low cost platforms can be used together to perform higher-level tasks such as establishing communication network[13], [14], formation flying (leader-follower approach)[15], migration[16] or sensor network[17].

However, swarm literature always assume that the individual have a common goal such as migrating, mapping or forming sensor network. These strategies should therefore be adapted in order to cope with agents with individual goals such as PAVs in a commuting scenario.

2. Real-time simulator

A simulator was developed at EPFL-LIS able to simulate large number of flying agents in real-time or faster than real-time. The simulator was written in Ada[4], a language designed for concurrent real-time systems and high-reliability applications. To obtain meaningful results, vehicle dynamics are simulated with a basic physical model, using response rates based on expected PAV dynamics as developed by the Liverpool project partners.

The source code of the simulator is easily extensible and any type of collision avoidance strategy can be implemented as well as any sensor input can be simulated. Hundreds of PAVs can be simulated in real-time on a current multi-core CPU.

In real-time mode, the Graphical User Interface (GUI) permits to either follow the trajectory of a particular flying agent (see Fig. 1) or to freely navigate in the environment to see the simulation run under different perspectives. At any time, the simulation can be paused and the user can freely navigate around to change his perspective.

The ability to go faster than real-time allows running a large number of simulations in order to explore the parameter space of the process and to have statistical meaningful results.

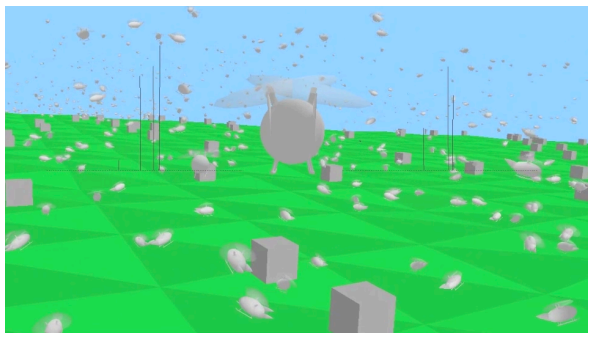

Figure 1: GUI of the Swarm simulator (very dense traffic).

2.1. Simulator data flow

A simulation step is split into five different stages (see Fig. 2) for each vehicle: three stages in the simulated real world (i.e. these stage could be implemented on a real platform) and two in the simulator layer (i.e. a layer that exists only in simulation). The output of each stage is the input of the following one. First, the neighbor selection stage selects the agents that are close enough to be in the sensor range of each other.. With this set, the simulated sensors feed the autopilot with various data (e.g. position, velocity, bearing) depending on the set of simulated sensors. With this information, the autopilot computes the command to the actuators, which

[4] http://www.adaic.org/

then creates a command to the control surfaces. The dynamics of the agent are simulated in a last stage, updating the state of the agent.

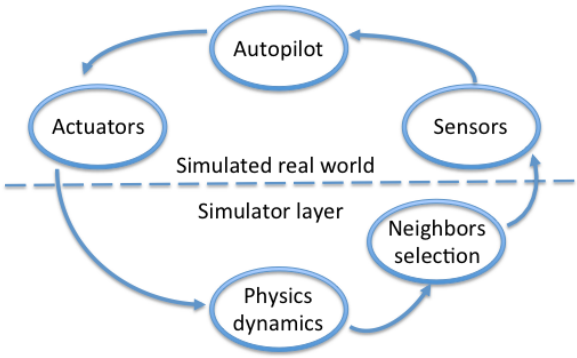

Figure 2: Simulator data flow chart.

Parallelization of the architecture of the different tasks allows taking advantage of multi-core computers to increase processing power. However computation for each agent should be synchronized at every step after the computation of the physics dynamics leading to complex synchronization events. Therefore, the simulator parallelizes each step one after the other, ensuring synchronization and efficient use of the computational power available.

3. Flocking for agents with individual goals

3.1. Modified Reynolds flocking

We propose an extension to the standard flocking in order to deal with agents with individual goals. In this Section, we recall first the seminal rules of Reynolds flocking[10] and then show how these rules can be adapted to cope with agents with individual goals.

In [10], Reynolds proposed three simple rules to explain the global behavior of a flock. These rules are based only on local sensing and interaction with local neighbors. These rules are

- *Repulsion*: if an agent is too close from a neighbor, it is pushed away;
- *Velocity matching*: the agent try to match its velocity with the nearby flock mates;
- *Flock centering*: the agent tries to reach to average position of its nearby flock mates.

Reynolds didn't propose any mathematical implementation for these rules. In [18], the authors presented a mathematical analysis as well as mathematical formulation of these rules. In this work, we base our approach on this formulation. In addition to Reynolds' rules, the authors proposed a fourth rule called *migration* that makes the agent move towards its goal.

The authors showed also that not all neighbors could be taken to perform flocking. For an agents i located as q_i, the set of neighbors, N_i, that are selected for flocking, is defined as all neighboring agent j such that

$$N_i = \{j : \|q_j - q_i\| < r\}$$

where r is the interaction range. Only the agents of this set will influence agent i.

3.1.1. Mathematical definition

The mathematical formulation tend to create a formation where all agents are equally distant to its closest neighbors, in other words

$$\|q_j - q_i\| = d \; \forall j \in N_i$$

In [18], the authors uses a σ-norm for smoothness and to overcome the fact that the Euclidean norm is not differentiable at $z = 0$. The σ-norm is defined as

$$\|z\|_\sigma = \frac{1}{\epsilon}\left[\sqrt{1 + \|z\|^2} - 1\right]$$

Therefore, the gradient is equal to

$$\sigma_\epsilon(z) = \frac{z}{1 + \epsilon\|z\|_\sigma}$$

where $\epsilon > 0$. The authors use a fixed value equal to 0.1.

In addition, the authors defined a bump function which varies smoothly between 0 and 1 and is null above 1

$$\rho_\delta(z) = \begin{cases} 1, z \in [0, \delta) \\ \frac{1}{2}\left[1 + \cos(\pi\frac{z-\delta}{1-\delta})\right], z \in [\delta, 1) \\ 0, otherwise \end{cases}$$

with $\delta \in [0,1]$. The authors use again a fixed value for this parameter equal to 0.2.

Finally, the authors defined an action function $\phi_\alpha(z)$ that vanishes for $z \geq r_\alpha$

$$\phi_\alpha(z) = \rho_\delta(z/r_\alpha)\phi(z - d_\alpha)$$

$$\phi(z) = \frac{a+b}{2}\frac{z+c}{\sqrt{1+(z+c)^2}} + \frac{a-b}{2}$$

where $0 < a \leq b, c = (b-a)/\sqrt{4ab}$. This function, when integrated, creates a smooth attractive/repulsive pairwise potential $\psi_\alpha(z)$ which has a global minimum at $z = d_\alpha = \|d\|_\sigma$ and a finite cut-off at $z = r_\alpha = \|r\|_\sigma$. It is furthermore differentiable for all z.

3.1.2. Control strategy

The control input, u, of agent $i$is composed of the sum of three terms:

$$u_i = u_i^{mp} + u_i^{vm} + u_i^{\gamma}$$

where u_i^{mp} is a gradient-based term; u_i^{vm} is the velocity consensus term and u_i^{γ} is a navigational feedback term.

Gradient-based term

The gradient-based term u_i^{mp}, or motion planning term encompasses both the *separation* (Fig. 3) and the *cohesion* (or *flock centering*) (Fig. 4) rules. The separation rule ensures that agents that are too closed from each other are getting repulsed from each other. The cohesion rule tends to steer the agent towards the center of mass of its neighbor set, N_i. u_i^{mp} is then defined as

$$u_i^{mp} = \sum_{j \in N_i} \phi_\alpha(\|q_j - q_i\|)\, n_{ij}$$

where ϕ_α is the action function previously defined, and $n_{ij} = \sigma_\epsilon(q_j - q_i)$ is a vector connecting q_i to q_j.

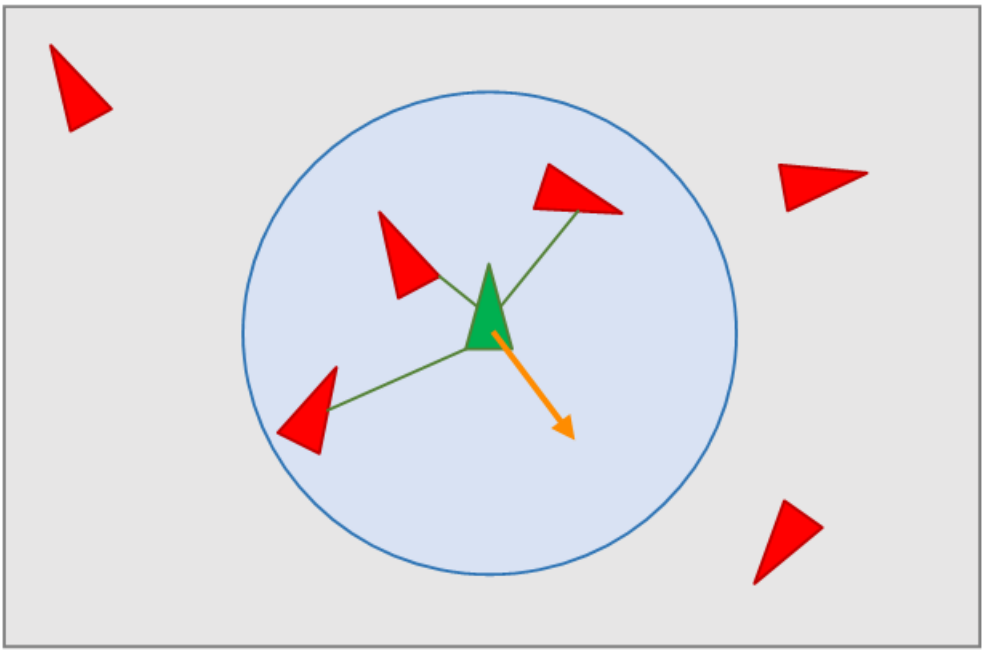

Figure 3 Separation principle

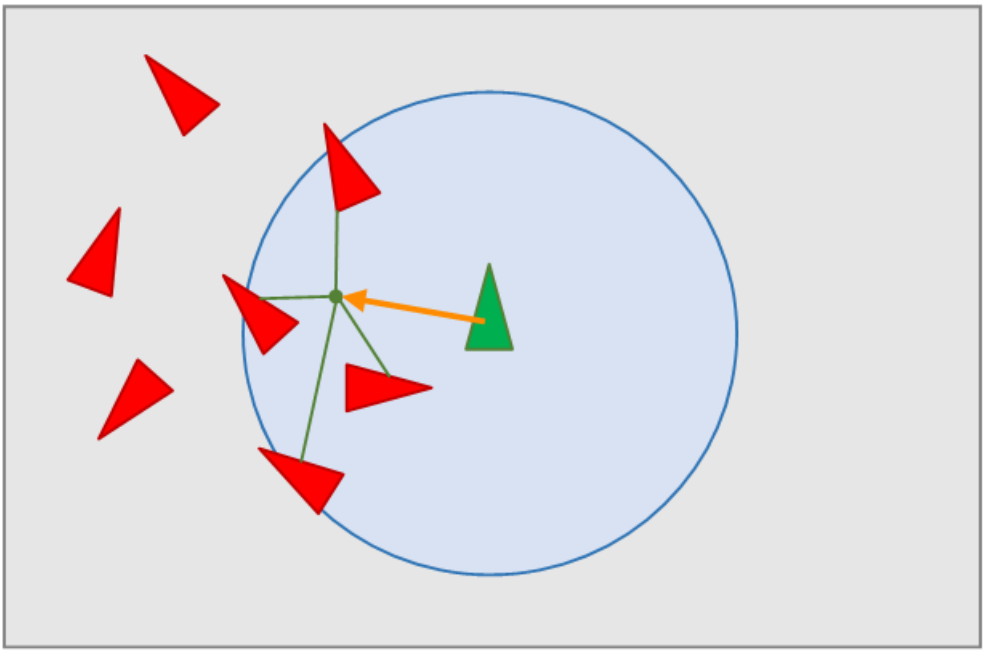

Figure 4 Cohesion principle

Velocity consensus term

The velocity consensus rule term u_i^{vm} tends to align the velocity vectors (both direction and norm) of every agent in N_i (Fig 5). This term is based on the relative velocity between the agent and its neighbor

$$u_i^{vm} = \sum_{j \in N_i} a_{ij}(p_j - p_i)$$

where $a_{ij} = \rho_\delta\left(\frac{\|q_j - q_i\|_\sigma}{r_\sigma}\right)$ is calculated by the bump function defined previously.

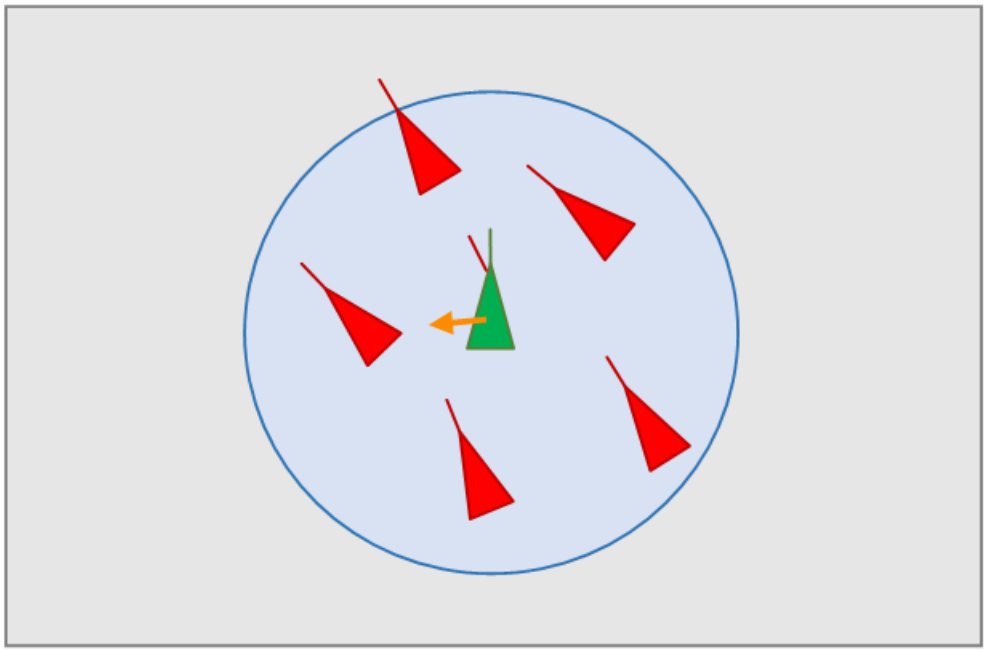

Figure 5 Velocity alignment principle

Migration term

The migration term (Fig. 6) u_i^γ makes the agent move towards its goal. This goal could be non-static. Therefore the control is expressed by

$$u_i^\gamma = -c_1(q_i - q_g) - c_2(p_i - p_g)$$

where $c_1, c_2 > 0$ are constants, q_g is the position of the goal and p_g is the velocity of the goal.

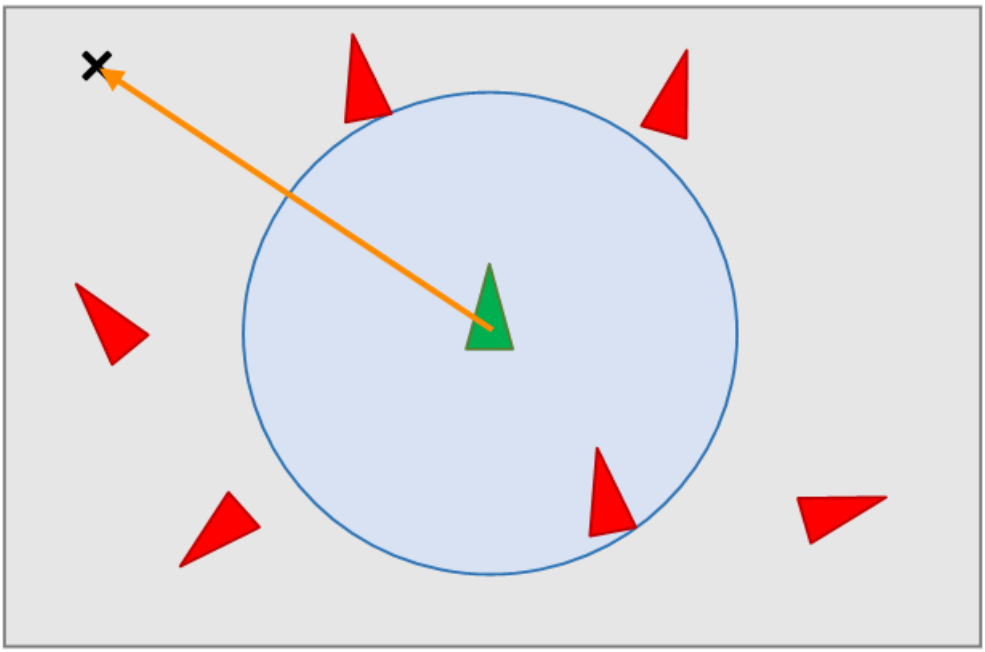

Figure 6 Migration principle

3.1.3. Modifications for agents with individual goals

The strategy extracted from [18] and presented so far as two drawbacks. First, the agents are assumed to have the same goal (or migration point). Second, the control input is equal to the derivative of the velocity. In the case of a PAV, we control directly the velocity and therefore the control scheme defined in the previous section has to be adapted.

Agents with individual goals

The first issue is solved in two steps. First, the global goal q_g, is made specific to every PAV. In addition, the goal of each PAV is static such that the dynamic of the point is null and c_2 can be set to zero. Therefore, the migration term becomes

$$u_i^\gamma = -c_1(q_i - q_{g_i}).$$

Second, there is no differentiation made on the current flight direction of neighboring PAVs and every agent will tend to get closer to neighbors even if they fly in opposite direction. This is clearly suboptimal and counter-intuitive in terms of human acceptance. One would not understand why the PAV steers towards potentially colliding neighbor.

We introduce a coefficient c_d in the attraction and in the velocity matching terms. The PAV will be more attracted to PAVs flying in the same direction than the ones flying in the opposite direction. It will also tend to align more its velocity to PAVs travelling in the same direction. However, the repulsion will remain the same no matter the similarity of flight directions. The coefficient c_d can be defined as

$$c_d = \left(\frac{< v_{opt}, v_{neighbor} >}{\|v_{opt}\|\|v_{neighbor}\|} + 1\right)/2 \in [0,1]$$

where v_{opt} is a vector pointing from the position of agent i towards the direction of its goal (or its next waypoint), and $v_{neighbor}$ a vector pointing from the position of agent i towards the direction of the considered neighbor. This means that an agent going close to the same

direction of agent's goal direction will have a coefficient close to 1, whereas an agent going in opposite direction will have a coefficient close to 0. We can furthermore be normalized this term with the number of neighbors. The gradient-based term becomes

$$u_i^{mp} = \frac{1}{card(N_i)} \sum_{j \in N_i} c_{ij} \phi_\alpha \left(\|q_j - q_i\|_\sigma \right) n_{ij}$$

which can also be expressed as

$$u_i^{mp} = \frac{1}{card(N_i)} \sum_{j \in N_i} c_{ij} a_{ij} \phi \left(\|q_j - q_i\|_\sigma - d_\alpha \right) n_{ij}$$

where $c_{ij} = \begin{cases} 1, if \ \|q_j - q_i\|_\sigma < d_\alpha \\ c_d, \ otherwise \end{cases}$.

Figure 7 shows the amplitude of this last term (in magenta) for the case with a single neighbor considering a coefficient $c_d = 1$. This term is composed of the action function ϕ_α (in green), which is itself the combination of a_{ij} and ϕ and of the norm of the vector $n_{ij} = \sigma_\epsilon(q_j - q_j)$. As it can be seen, the repulsion is stronger than the attraction in order to avoid having agents stuck together in small flock. Furthermore, this term is null at the scale of the lattice.

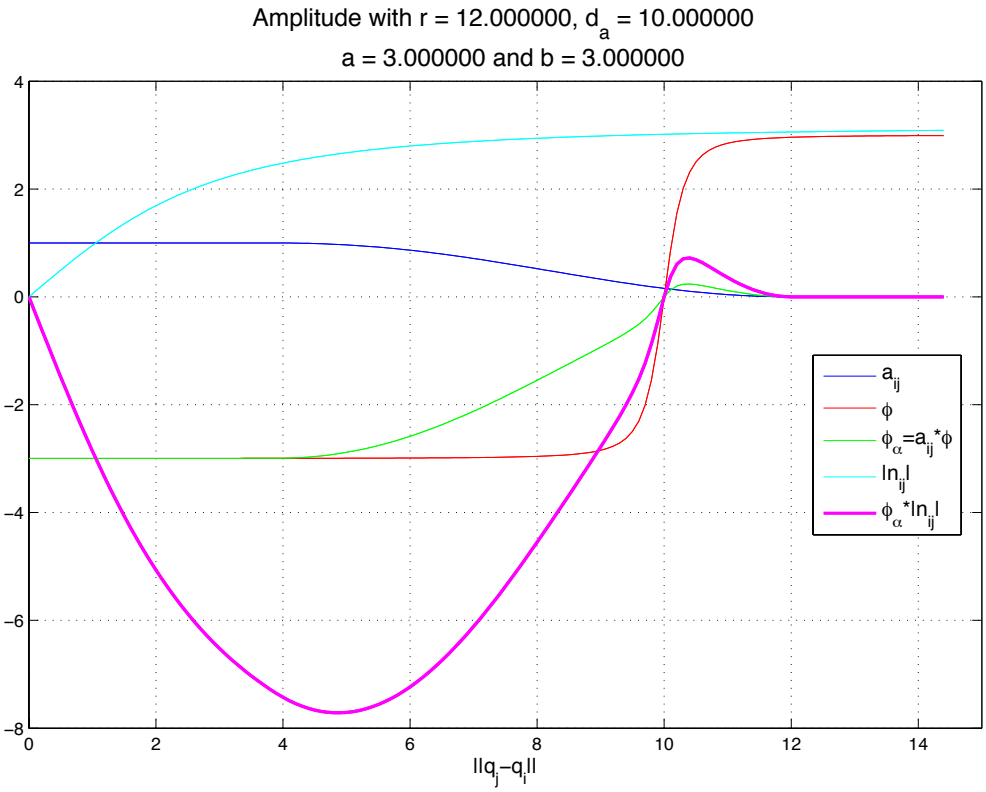

Figure 7 Amplitude of the motion planning term for a single neighbor

The coefficient c_d is also used in the velocity matching control equation. It replaces a_{ij} such that the agent tends to more align its velocity towards neighbors travelling in the same direction than its goal.

Velocity control

To transform the control in a velocity control, we use Euler interpolation i.e.

$$v_{i_c}(t) = u_i \Delta t + v_i(t-1)$$

3.2. Joining/leaving a flock

In the previous section, we presented an improvement of standard Reynolds flocking to deal with agents with individual goals. The influence of the relative flight direction tends to create naturally flock of PAVs flying in the same direction. In similar fashion, when the direction of flight of a PAV in a flock changes too much, the PAV will naturally leave the flock and continue to fly towards its goal.

Joining and leaving flocks is done in a continuous way, avoiding oscillation problem that could occur if a binary state of membership was introduced.

4. Avoiding Non-flying zones

Non-flying zones are generally known beforehand and published by the aviation authorities of the area. The might be active only at certain period of time (e.g. military shooting areas) or at all time (e.g. nuclear plants, large airports). However, these changes occur over large period of time usually many orders of magnitude above the flight time planned for PAVs. In addition, the geographical zone on which they are active is fixed and do not move over the activation period.

Therefore, it can be assumed that the location of these zones are known beforehand and won't change over the time of a flight. Non-flying zones can in principle be avoided solely by a navigation scheme as opposed to a collision avoidance scheme to avoid mid-air collision avoidance as presented in Deliverable 5.3.

Navigation strategies can be run *à priori* of the flight in order to establish a flight plan. Each PAV can compute its own flight plan by considering the non-flying zone he should encounter on the trajectory to its destination.

4.1. Convex hull

The geometry of the non-flying zone can be complex. In order to find the shortest trajectory around a non-flying zone, we need to compute the convex hull of the geometry. We apply the Graham-Scan technique to compute this convex hull.

4.1.1. Graham-Scan technique

First, the point with the minimum xy-coordinates is selected as starting point (Fig. 8).

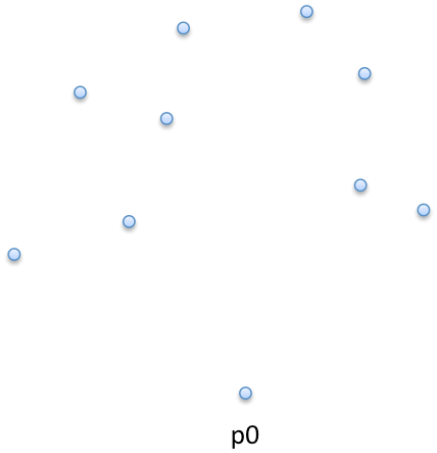

Figure 8 A set of point defining the non-flying zone, p0 is the first selected point to compute the convex hull

Then, the points are sorted by polar angle in counter clockwise order around p0 (Fig. 9).

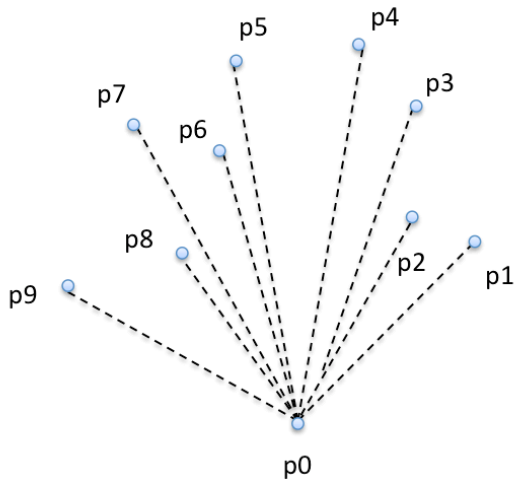

Figure 9 Ordering of the points around p0

In the next step, the angle between two consecutive edges is computed and the common point rejected if it is not a vertex of the convex hull i.e. if at this point the curve make a non-left turn (Fig 10).

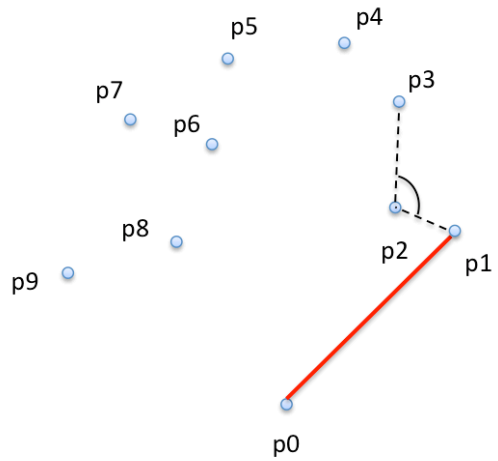

Figure 10 Selection of p1 and rejection of p2

The operation is repeated until the whole convex hull is obtained (Fig. 11).

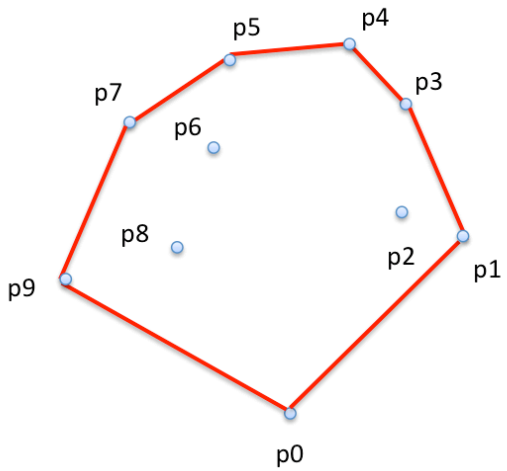

Figure 11 Convex hull

4.2. Shortest path roadmaps

Introduced first in [19] and as the name states it, the shortest path planning strategy try to search for the shortest path between a start and ending position. Many different variation of this strategy exists and the interested reader can find plenty of them in [20].

4.2.1. Working principle

When an intersection between the trajectory and a flying zone is detected (Fig. 12), the distance from all the vertices of the non-flying zone to the trajectory is computed. On each side of the trajectory, the furthest vertices can be found and the closest furthest vertex is selected as new waypoint. This allows getting around the non-flying zone following the shortest path.

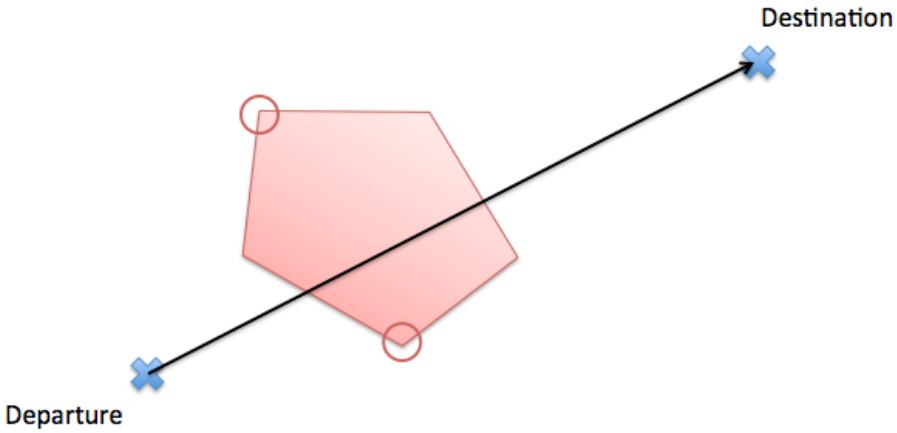

Figure 12 Furthest vertices on both side of the trajectory

The trajectory is therefore divided into segments between two waypoints (Fig. 13).

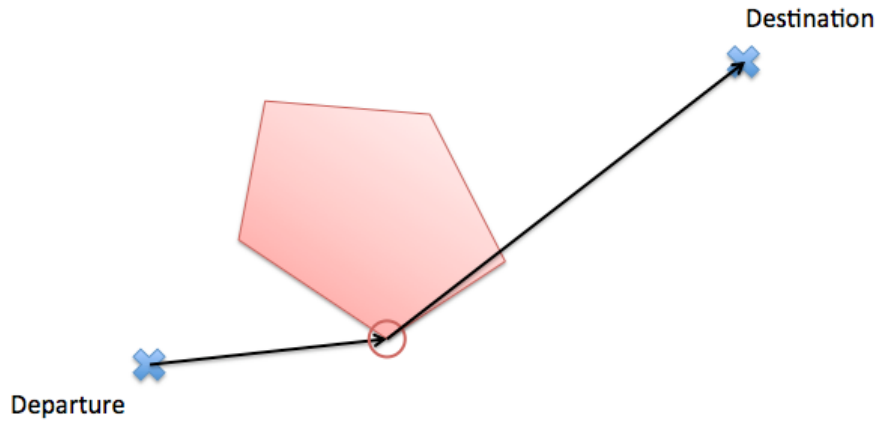

Figure 13 First vertex selected

The process is repeated recursively on each part of the new trajectory until no intersection between the trajectory and the non-flying zone is detected (Fig. 14,15).

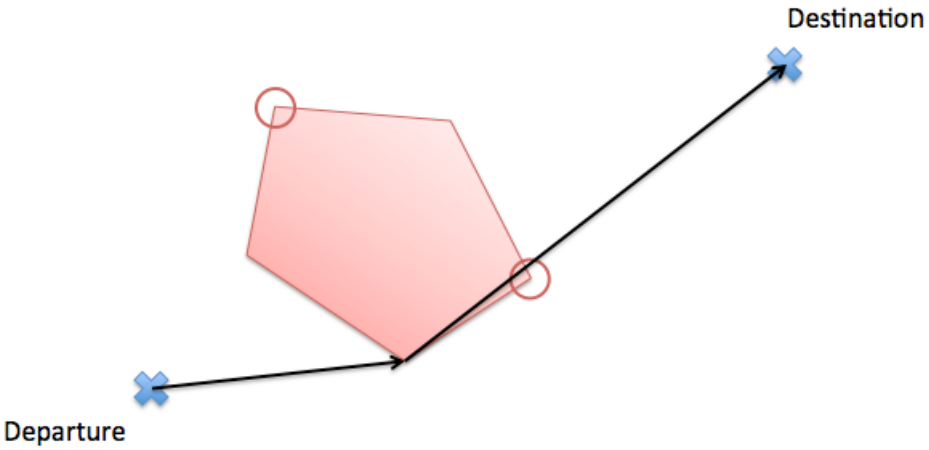

Figure 14 Selection of the second waypoint

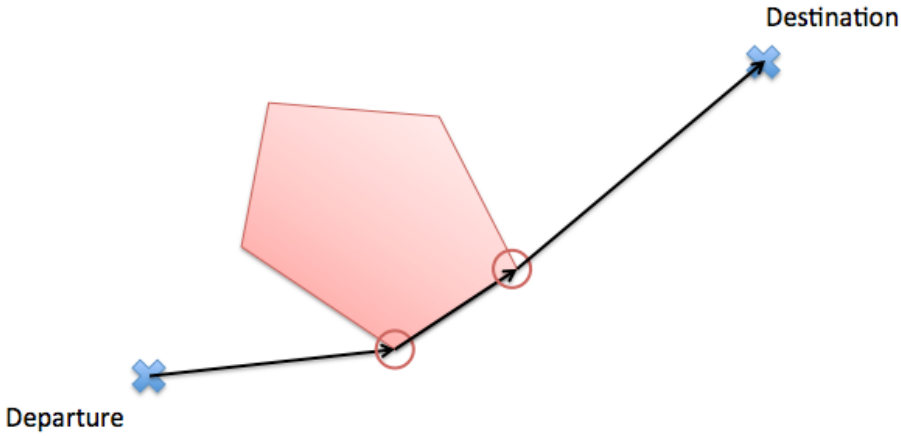

Figure 15 Final trajectory around the non-flying zone

4.3. Safety margin

The waypoints are located at the boundary of the non-flying zone. Any imprecision in the guidance process and the PAV would violate the non-flying zone. Therefore, a safety margin (Fig. 16 in green) is added around the non-flying zone. The size of the safety margin can be adapted to the performance of the guidance strategy.

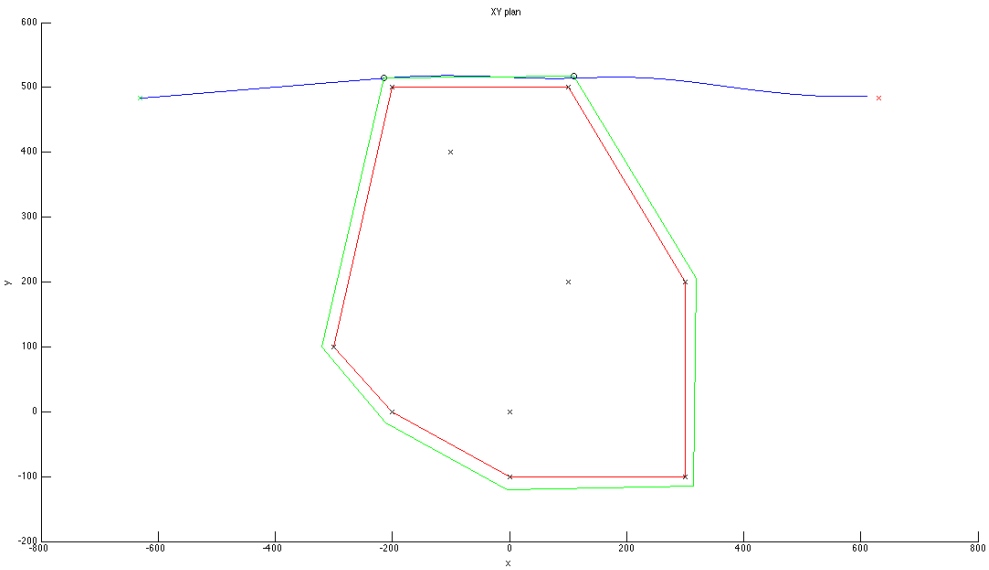

Figure 16 Flight (in blue) around a non-flying zone (red) with a safety margin (green).

4.4. Multiple non-flying zones

The extension to multiple non-flying zones is straightforward. The strategy can simply be applied successively to one zone after another (Fig. 17).

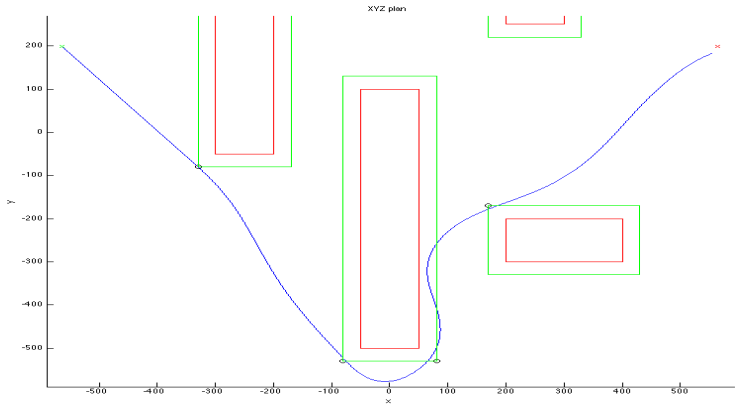

Figure 17 Trajectory with multiple non-flying zones

4.5. Flying corridor and multi-PAV simulations

In the previous Section, we showed that a single PAV could fly between a start and end position while avoiding non-flying zones. On the top of this navigation strategy, the collision avoidance strategy presented in Deliverables 5.3 ensures that mid-air collisions are avoided. Therefore, the combination of these two strategies leads to have multiple PAVs avoiding multiple non-flying zones while avoiding mid-air collisions. An example of such scenario is presented on Figure 18.

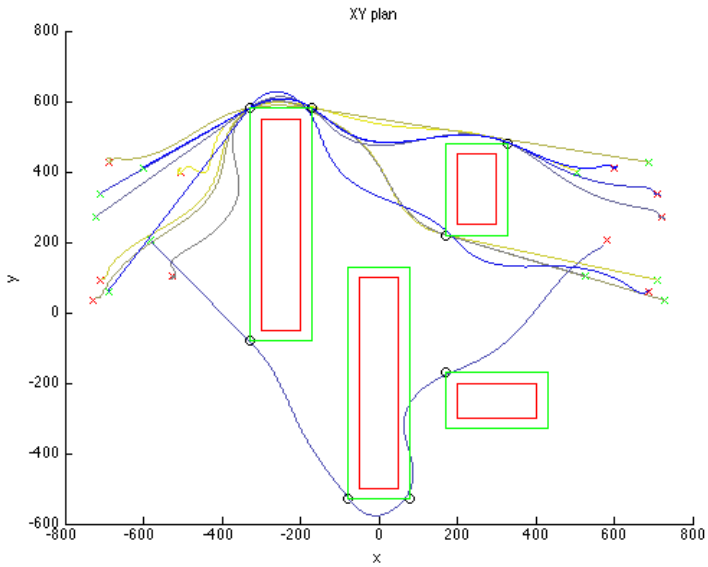

Figure 18 Simulation with multiple PAVs avoiding multiple non-flying zones

5. Conclusion

In this Deliverable, we showed how Reynolds flocking rules could be adapted in order to cope with agents with individual goals. We proposed to take into account the difference of intended flight direction to alter the influence of selected rules. This allows the PAVs to join, create and leave flocks depending on their actual flight direction.

In addition, we showed how non-flying zones could be avoided by many PAVs while avoiding collisions. Simulations of many PAVs avoiding many non-flying zones were performed.

6. Bibliography

[1] E. Sirot and F. Touzalin, “Coordination and Synchronization of Vigilance in Groups of Prey: The Role of Collective Detection and Predators’ Preference for Stragglers,” *Am. Nat.*, vol. 173, no. 1, pp. 47–59, Jan. 2009.

[2] B. L. Partridge, “The structure and function of fish schools,” *Sci. Am.*, vol. 246, no. 6, pp. 114–123, 1982.

[3] W. L. Romey and E. Galbraith, “Optimal group positioning after a predator attack: the influence of speed, sex, and satiation within mobile whirligig swarms,” *Behav. Ecol.*, vol. 19, no. 2, pp. 338–343, Apr. 2008.

[4] D. Helbing, P. Molnar, I. J. Farkas, and K. Bolay, “Self-organizing pedestrian movement,” *Environ. Plan. B*, vol. 28, no. 3, pp. 361–384, 2001.

[5] H. P. Zhang, A. Be’er, E. L. Florin, and H. L. Swinney, “Collective motion and density fluctuations in bacterial colonies,” *Proc. Natl. Acad. Sci.*, vol. 107, no. 31, pp. 13626–13630, 2010.

[6] M. Moussaid, S. Garnier, G. Theraulaz, and D. Helbing, “Collective Information Processing and Pattern Formation in Swarms, Flocks, and Crowds,” *Top. Cogn. Sci.*, vol. 1, no. 3, pp. 469–497, 2009.

[7] Y. Li and X. Chen, “Leader-formation navigation with sensor constraints,” in *Information Acquisition, 2005 IEEE International Conference on*, 2005, p. 6–pp.

[8] M. Saffarian and F. Fahimi, “A novel leader-follower framework for control of helicopter formation,” in *Aerospace Conference, 2007 IEEE*, 2007, pp. 1–6.

[9] T. Paul, T. R. Krogstad, and J. T. Gravdahl, “Modelling of UAV formation flight using 3D potential field,” *Simul. Model. Pract. Theory*, vol. 16, no. 9, pp. 1453–1462, 2008.

[10] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *ACM SIGGRAPH Computer Graphics*, 1987, vol. 21, pp. 25–34.

[11] R. Olfati-Saber, “A unified analytical look at Reynolds flocking rules,” 2003.

[12] N. R. Watson, N. W. John, and W. J. Crowther, “Simulation of unmanned air vehicle flocking,” in *Theory and Practice of Computer Graphics, 2003. Proceedings*, 2003, pp. 130–137.

[13] S. Hauert, S. Leven, J. C. Zufferey, and D. Floreano, “Communication-based Swarming for Flying Robots,” in *Proc. Intl. Conf. Robotics and Automation Workshop on Network Science and Systems*, 2010.

[14] S. Hauert, L. Winkler, J.-C. Zufferey, and D. Floreano, “Ant-based swarming with positionless micro air vehicles for communication relay,” *Swarm Intell.*, vol. 2, no. 2–4, pp. 167–188, Aug. 2008.

[15] F. Giulietti, L. Pollini, and M. Innocenti, “Autonomous formation flight,” *Control Syst. Mag. IEEE*, vol. 20, no. 6, pp. 34–44, 2000.

[16] E. W. Justh and P. S. Krishnaprasad, “A Simple Control Law for UAV Formation Flying,” Institute for Systems Research, TR 2002-38, 2002.

[17] C. M. Cianci, J. Pugh, and A. Martinoli, “Exploration of an Incremental Suite of Microscopic Models for Acoustic Event Monitoring Using a Robotic Sensor Network,” in *Proc.*

of the 2008 IEEE Int. Conf. on Robotics and Automation, Pasadena, USA, 2008, pp. 3290–3295.

[18] R. Olfati-Saber, "Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[19] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," DTIC Document, 1969.

[20] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of Discrete and Computational Geometry, Second Edition*. CRC Press, 2004.