Distributed Control for Low Computational Cost Satellite Swarm Obstacle Avoidance

Jonathan Green
gjonatha@ethz.ch

Tasman de Pury
tdepury@student.ethz.ch

***Abstract*— We propose an online, low computational cost, strategy for distributed control of satellites in a swarm is proposed. The efficacy of this approach is discussed analytically and through simulation. A range of different methods for dynamically forming the graph structure of the problem are compared.**

I. INTRODUCTION

The rapid reduction of the cost associated with placing objects in orbit has caused a marked increase in the accessibility of space [1]. As a result, picosatellites (defined as satellites with a mass of <1kg [2]) have become a popular and affordable option for deployment of hardware in space. Their small size, relative simplicity, and low cost of launch means that they are well suited for use in "swarms", configurations in which many individual satellites operate in formation.

Ensuring that satellite swarms are capable of avoiding other objects in orbit is essential to both maintain their function and to prevent the creation of unnecessary space junk. However, for picosatellite swarms, this is complicated by the difficulty associated with tracking small orbital objects (namely the satellites themselves) with conventional ground-based methods. On-board, online collision avoidance algorithms are a potential solution to this, however such methods often have a high computational cost that is at odds with the limited capability of picosatellites.

In this work we propose the use of a distributed control strategy that allows for each individual satellite to locally run a low computational cost controller that nevertheless evolves complex collision avoidance behaviour at the swarm scale. We explore different formulations of this strategy to best avoid collisions while maintaining the benefits of a local, online controller.

II. MODELLING

A. Satellite Swarm

In this work we consider space missions in which a set of n satellites $S = \{s_1, s_2, \dots, s_n\}$ are desired to maintain a given formation (referred to henceforth as a "swarm") for the duration of the mission. We restrict the scope to static swarm configurations, where the desired formation is constant in time. Additionally, we consider a set of m obstacles $O = \{o_{n+1}, o_{n+2}, \dots, o_{n+m}\}$. These obstacles are moving objects that will compromise the mission if they pass too close to any of the satellites in the swarm. In practice, this may take the form of space junk that will destroy satellites in the case of a collision, or other spacecraft that may interfere with communications if they pass too close to any of the swarm members.

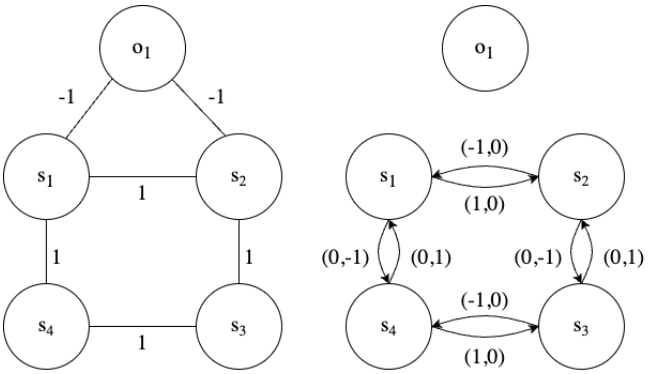

Fig. 1. An example graph of a possible system showing the augmented adjacency matrix (left), and formation matrix (right). Note that in this system, only satellites s_1 and s_2 have detected obstacle o_1

The satellites have the ability to communicate with one another, and have the ability to detect obstacles under certain conditions (see Section III-C). To capture this information, we model the swarm as a graph G. The communication links between satellites and detection of obstacles are encoded in an augmented adjacency matrix A of G, where

$$A_{ij} = \begin{cases} 1 & s_i \text{ communicates with } s_j \\ -1 & s_i \text{ detects } o_j \\ 0 & \text{otherwise} \end{cases}$$

Additionally, the desired formation of the swarm is defined with formation matrix

$$F = \begin{bmatrix} \boldsymbol{r_{11}} & \cdots & \boldsymbol{r_{1n}} & \\ \vdots & \ddots & \vdots & 0_{n\times m} \\ \boldsymbol{r_{n1}} & \cdots & \boldsymbol{r_{nn}} & \\ & 0_{m\times n} & & 0_{m\times m} \end{bmatrix}$$

where $\boldsymbol{r_{ij}} \in \mathbb{R}^2$ is a vector describing the desired position of satellite j relative to satellite i. Note that this matrix is zero for entries that correspond to obstacles, as they are not a desired part of the formation.

B. Dynamics

We describe the state of the ith satellite with

$$\boldsymbol{x_i} = \begin{bmatrix} \boldsymbol{p_i} \\ \boldsymbol{\dot{p}_i} \end{bmatrix}$$

where $\boldsymbol{p_i} \in \mathbb{R}^2$ denotes the satellite's cartesian position. Each satellite in the swarm is modelled as a point mass under Newtonian motion. Given the low drag environment of space, we neglect any damping terms. We model only the translational (not rotational) dynamics of each satellite. Additionally, for ease of visualisation we restrict the system to a two dimensional plane. Finally, we define the satellites as being actuated with two linearly independent thrusters, which can produce a resultant force vector in any direction. Under these assumptions, the dynamics of each satellite s_i can be described with a simple double integrator, following

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}}_{\boldsymbol{i}} \\ \boldsymbol{u_i} \end{bmatrix}$$

Where $\boldsymbol{u_i} \in \mathbb{R}^2$ denotes the satellite's thrust vector.

C. Problem Statement

Given a swarm defined by a graph G with a desired formation F and a set of obstacles O with unknown behaviour, we seek to develop a controller that:

1) Steers the system asymptotically towards F when no obstacles are present
2) Steers satellites away from obstacles when they are present
3) Is stabilising regardless of the presence of obstacles
4) Has low computational complexity

To evaluate the impact of different approaches for obstacle detection, we define the following controller performance metric along a trajectory of length $t \in [0, T]$:

$$\frac{1}{T}\int_0^T \sum_{i=0}^{n} \frac{1}{\mathcal{N}_{G(t)}(i)} \sum_{j \in \mathcal{N}_{G(t)}(i)} \|(\boldsymbol{p_j}(t) - \boldsymbol{p_i}(t)) - \boldsymbol{r_{ij}}\|_2^2 \, dt$$

Note that the above metric is normalised over both time and the number of edges. Hence the value is the average formation discrepancy per edge, squared.

III. Analysis

A. Controller

We propose the following controller:

$$\boldsymbol{u_i} = -k_p \boldsymbol{p} - k_{d,i} \boldsymbol{v_i} + \sum_{j=1}^{n} A_{ij}[\gamma_p g_j(\boldsymbol{p_\Delta})(\boldsymbol{p_\Delta} - F_{ij}) + \gamma_d h_j(\dot{\boldsymbol{p}}_{\boldsymbol{\Delta}})\dot{\boldsymbol{p}}_{\boldsymbol{\Delta}}]$$

where $\boldsymbol{p_\Delta} = \boldsymbol{p_j} - \boldsymbol{p_i}$ and $\dot{\boldsymbol{p}}_{\boldsymbol{\Delta}} = \dot{\boldsymbol{p}}_{\boldsymbol{j}} - \dot{\boldsymbol{p}}_{\boldsymbol{i}}$ are differences in positions and velocities respectively. Additionally, $k_p, k_d, \gamma_p, \gamma_d > 0$. The functions $g_j : \mathbb{R}^2 \to \mathbb{R}$ and $g_j : \mathbb{R}^2 \to \mathbb{R}$ are used to switch the controller between formation forming and obstacle avoidance, and are defined as

$$g_j(\boldsymbol{p_\Delta}) = \begin{cases} 1 & j \leq n \\ \frac{1}{\|\boldsymbol{p_\Delta}\|_2^2} & \text{otherwise} \end{cases}$$

and

$$h_j(\dot{\boldsymbol{p}}_{\boldsymbol{\Delta}}) = \begin{cases} 1 & j \leq n \\ \epsilon & \text{otherwise} \end{cases}$$

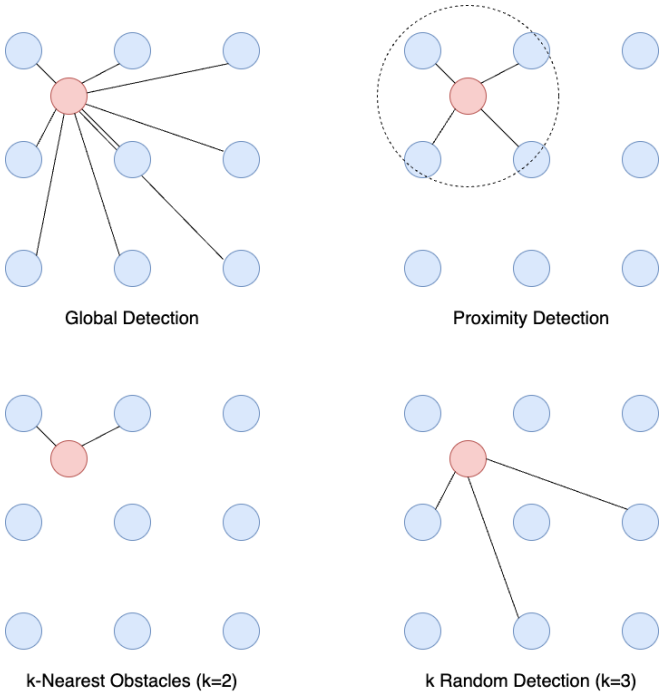

Fig. 2. The different detection methods visualised on an example graph. The blue circles are satellites, and the red circle is an obstacle. A black line indicates that the obstacle has been detected

where $\epsilon \to 0$. Note that both $g_j(\cdot)$ and $h_j(\cdot)$ are strictly positive. This can be expressed as a second order Laplacian flow by writing the control law in matrix form as

$$\ddot{\boldsymbol{p}} + (k_d I_{n+m} + \gamma_d h_j(\dot{\boldsymbol{p}}_{\boldsymbol{\Delta}})L)\dot{\boldsymbol{p}} + (k_p I_{n+m} + \gamma_p g_j(\boldsymbol{p_\Delta})L)\boldsymbol{p} - R = 0_{(n+m)\times 1}$$

where $R \in \mathbb{R}^{(n+m)\times 1}$ is defined as the main diagonal of AF^T.

B. Stability

The controller described in section III-A will stabilise the system and converge to the prescribed formation when: the satellite communication network is constructed such that G is connected and the obstacles are tracking a constant velocity. *Proof:* First note that the functions $g_j(\cdot)$ and $h_j(\cdot)$ tend to zero as $t \to \infty$ under constant obstacle velocity tracking. In the control law this is the equivalent of:

$$A = \begin{pmatrix} A_{1:n\times 1:n} & 0_{n\times m} \\ 0_{m\times n} & 0_{m\times m} \end{pmatrix}$$

Perform a change of variables to express the system in error coordinates $\boldsymbol{p_{e,i}} = \boldsymbol{p_i} - F_i$ for $i = 1, \ldots, n+m$. Then, in the absence of control influences from obstacles the closed-loop dynamics become the canonical form [3][Sec 8.1.1] for second order laplacian flow as they can be expressed as

$$\ddot{\boldsymbol{p}}_{\boldsymbol{e}} + (k_d I_{n+m} + \gamma_d L)\dot{\boldsymbol{p}}_{\boldsymbol{e}} + (k_p I_{n+m} + \gamma_p L)\boldsymbol{p_e} = 0_{(n+m)\times 1} \quad (1)$$

From [3][Thm 8.7], it can be seen that a second order Laplacian flow of form (1) will converge in the sense that

$$\lim_{t\to\infty} \|\boldsymbol{p_{e,i}}(t) - \boldsymbol{p_{e,j}}(t)\|_2 = \lim_{t\to\infty} \|\dot{\boldsymbol{p}}_{\boldsymbol{e,i}}(t) - \dot{\boldsymbol{p}}_{\boldsymbol{e,j}}(t)\|_2 = 0$$

if the Laplacian L of G is connected and if

$$k_p + \gamma_p > 0 \text{ and } k_d + \gamma_d > 0 \tag{2}$$

Condition (2) holds since $k_p, k_d\gamma_p, \gamma_d > 0$.

Hence, if G is connected, with the controller in section III-A the system will always converge to zero error, meaning the system will converge to the desired formation. ■

C. Dynamic Graph Construction

It can be seen from section III-B that the system will always converge to an equilibrium if G is connected. If we construct the underlying satellite communication network to be connected, then for a given obstacle o_i, the system will still converge for any possible set of satellites that detect it. This guarantee is because we only include obstacles in the system when they are detected, and detection by definition will add an edge that connects the existing network of G to o_i. We present four possible cases for obstacle detection to analyse, motivated by the real-world context of satellite swarms. They are:

1) **Global Detection**: all satellites always detect all obstacles. This may occur when every satellite in the swarm is highly capable of detecting obstacles.
2) **Proximity Detection**: a satellite will only detect an obstacle if it is closer than some fixed distance, i.e. satellite s_i will detect obstacle o_j iff $\|\boldsymbol{p_i} - \boldsymbol{p_j}\|_2 < \delta$ for some $\delta > 0$. This may occur when every satellite has obstacle detection capabilities, but the performance of their detector is limited.
3) ***k*-Nearest Obstacles**: the k closest satellites to an obstacle will detect it. This may occur when the swarm is distributing its compute across all the satellites, and only has the computational capacity to detect a limited number of obstacles at any given time.
4) ***k* Random Detection**: a random subset $S_D \subseteq S$ of size $k \leq n$ will detect the obstacle. This may occur when only a portion of the swarm have the ability to detect obstacles, and these detection-capable satellites are distributed throughout the swarm due to how they were deployed or as a result of their mission.

These strategies are illustrated in figure 2. They are, in effect, update rules for the augmented adjacency matrix of the graph G, as they determine how this matrix will change over time and as a function of state. This dynamic graph construction allows us to explore different object detection methods under the proposed controller.

IV. Simulation

The efficacy of the different detection methods described in section III-C were evaluated in simulation. For each strategy, the following test was run:

1) The system is initialised stationary and in the desired formation. A minimally connected grid formation was chosen as this was deemed a realistic goal formation for real world coverage problems.

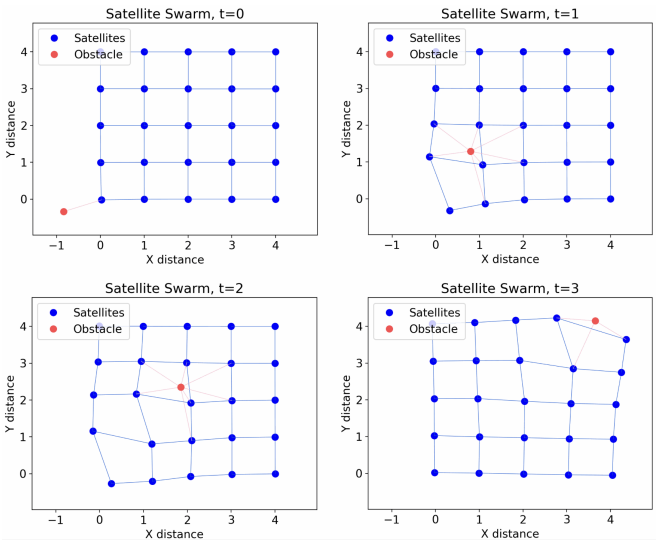

Fig. 3. An example of a simulation using k-Nearest Obstacle for dynamic graph formation. The blue lines indicate communication connections, and the red lines indicate detection. Note that the swarm successfully deforms around the obstacle to prevent a collision (t=1, t=2) before reforming to the target configuration as the obstacle departs (t=3)

2) The system is simulated with a single obstacle following a randomly initialised trajectory that passes through the swarm.
3) The performance metric described in II-C is computed for the simulation.
4) The above steps are repeated 100 times and the resulting performance metric averaged.

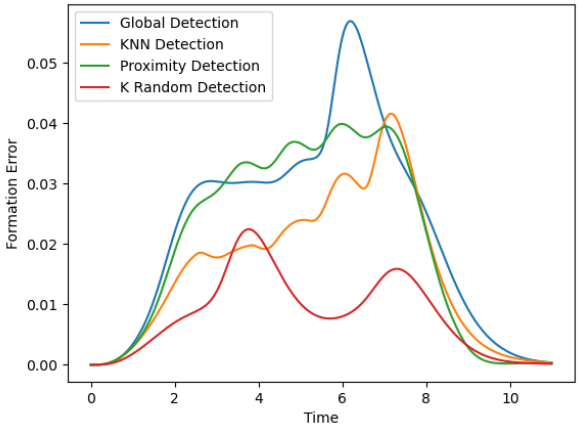

Fig. 4. A plot of the formation error as the obstacle passes through the swarm as shown in Fig. 3.

See Figure 3 for an example of a single simulation. The satellites are uniformly simulated with a unitary mass. The controller gains and simulation parameters used are recorded in table I. The code used for simulation can be found at the following repository: link.

Parameter	Value
k_p	1
k_d	0.5
γ_p	1
γ_d	1
Number of satellites	25
Number of obstacles	1

TABLE I

CONTROLLER GAINS USED IN SIMULATION

V. RESULTS

The results of the simulations are shown in Table II.

Graph Update Rule	Update Rule Parameter	Average Error
Global	NA	0.021
Proximity	$d < 1.5$	0.015
k-Nearest Obstacle	K=3	0.017
k Random Obstacle	K=7 (/25)	0.007

TABLE II

AVERAGE FORMATION ERROR DURING THE SIMULATIONS DESCRIBED IN SECTION IV

A. Performance Comparison

Quantitatively the k Random detection strategy performs better than the other options. Its low average error indicates that under this graph update rule, the control law was able to maintain the swarm close to the desired formation while avoiding obstacles. This performance is likely due to the average further separation between the k chosen satellites and the single obstacle as opposed to k-Nearest Obstacle or Proximity detection where fewer satellites have connections but are much closer. The $\frac{1}{\|\boldsymbol{p_j}-\boldsymbol{p_i}\|}$ term in the control law compounds this effect.

Qualitatively the k Random function causes more absolute movement of the whole formation whereas the localised detectors tend to create a "corridor" through the formation without shifting it in the global frame. This corridor could be a desired outcome in certain applications, and may be beneficial to swarms in which minimising total actuation effort is more important than maintaining formation.

Proximity and k-Nearest Obstacle had similar results to each other, both numerically (their average performance metric differed by only $\sim$13%), and qualitatively (their behaviour was observed to be similar in simulation). This can be explained by the fact that there is a large range of k values and detection distances for which the two strategies will have an object detected by the same subset of satellites.

As predicted in section III-B, the desired formation was reached regardless of the graph update rule used. This is indicated by the formation errors tending towards 0 in Fig 4. Additionally, while formation error increased in the presence of an obstacle, the swarm was successfully able to return to formation.

B. Computational Intensity

The controller described in section III-A and used in simulation has a computational time complexity of $O(n+m)$. Given that the simulations described in section IV considered a relatively small swarm of $n + m = 26$ this was not beneficial, however the linear time complexity of this controller means that it scales extremely well to larger swarms. This provides an advantage over many other collision avoidance methods which have, at best, polynomial time complexity. For example, those that rely on optimal control strategies (such as [4]) are limited by the complexity of solving the optimisation problem [6], and those that utilise probabilistic methods (such as [5]) require costly matrix inversion [7].

VI. LIMITATIONS AND OUTLOOK

While the proposed controller is capable of object avoidance, it does not do so flawlessly. In particular, while it will steer the system away from collisions, it does not provide any guarantees that collisions will be avoided. Additionally, the high level of coupling across the swarm means that often more satellites will move away from an obstacle than is necessary to avoid a collision. Finally, the controller does not have any capability to optimise a particular performance metric, such as minimising actuation effort or maximising a given measure of safety.

That being said, the success of this approach in demonstrating complex behaviour with a relatively simple control law motivates further exploration in this direction. Specifically, future works could consider applying a similar strategy to more complex dynamics (such as accurate orbital dynamics for a satellite), or adapt the existing strategy to swarm-based domains outside of space systems.

VII. CONCLUSION

The proposed distributed control strategy for picosatellite swarms demonstrates an effective approach for local, low computational cost collision avoidance. Through dynamic graph construction the system successfully steers satellites away from obstacles while maintaining the desired formation, and this behaviour has been demonstrated with a range of different methods for obstacle detection. Despite some limitations surrounding the performance of this approach, the results indicate promising directions for future research. Expanding this strategy could further enhance the capabilities of picosatellite swarms in space missions.

REFERENCES

[1] "Global Nanosatellite Market Anticipated to Reach $6.35 Billion by 2021, Reports BIS Research", Business Insider 2017 (accessed 28/07/2024), https://markets.businessinsider.com/news/stocks/global-nanosatellite-market-anticipated-to-reach-6-35-billion-by-2021-reports-bis-research-1002112400

[2] The Annual Compendium of Commercial Space Transportation: 2018 Federal Aviation Administration, 2018

[3] F. Bullo, Lectures on Network Systems, ed. 1.4, Kindle Direct Publishing, 2019, ISBN 978- 1986425643, with contributions by J. Cortés, F. Dörfler, and S. Martínez

[4] Siyuan Li, Dong Ye, Yan Xiao, Zhaowei Sun, Robust distributed model predictive control for satellite cluster reconfiguration with collision avoidance, Aerospace Science and Technology, Volume 130, 2022, 107917, ISSN 1270-9638, https://doi.org/10.1016/j.ast.2022.107917.

[5] M. E. Campbell and B. Udrea, ”Collision avoidance in satellite clusters,” Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301), Anchorage, AK, USA, 2002, pp. 1686-1692 vol.2, doi: 10.1109/ACC.2002.1023266.
[6] Potra, Florian A., and Stephen J. Wright. ”Interior-point methods.” Journal of computational and applied mathematics 124.1-2 (2000): 281-302.
[7] Strassen, Volker. ”Gaussian elimination is not optimal.” Numerische mathematik 13.4 (1969): 354-356.