Framework Development for Swarm Analysis and Design

Tristyn J. Noone & Norman G. Fitz-Coy

UF UNIVERSITY of FLORIDA
Duke UNIVERSITY
TEXAS The University of Texas at Austin
UC SANTA CRUZ

Current Status

- **Formalizing terms and definitions** relevant to swarm analysis and generalizing quantification metrics for the purposes of **comprehensive theoretical analysis** that includes **swarm stability and networked architecture** as considerations.
- Preliminary work done on **potential applications of networked architecture** in the arena of assured autonomy – specifically as it relates to the tracking of nearby objects.
- **Development of software elements** that facilitate these analyses.

Reintroduction

Why satellite swarms?

- To address the challenges of an ever-competitive, ever-congested space environment, satellite swarms **posit survivability of the system** in the event that individual satellites become disabled.
- Communication between member satellites of the swarm constitutes a **hierarchical networked architecture** in which some satellites are specialized to perform certain roles while others manage job assignments between them.
- Given the multiple points of view granted by satellites in a swarm, optical tracking of nearby objects becomes a viable means for **space situational awareness.**
- Use of member satellite orbital elements to **disguise** those of high-valued assets is being investigated as a potential application of satellite swarms.

Framework Development

Consists of **theoretical and practical components:**

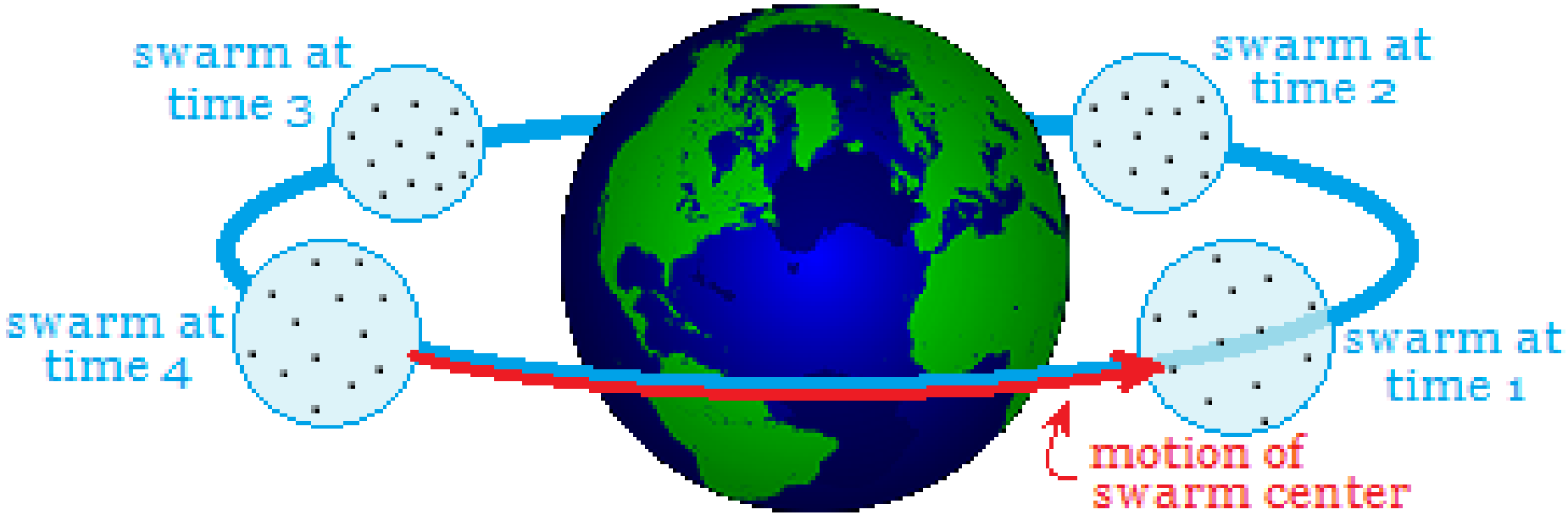
- Theoretical:
  - **Formalized definitions** for satellite swarm; geometry; etc.
  - **Assumptions** that narrow the scope of these definitions or otherwise reduce the complexity of the generalized problem.
  - Idealization of **Networked Architecture** including parameters, structure, limitations, and bounds on ability to keep swarm together.
- Testing:
  - In MATLAB®, a library of highly modular **classes and functions** to enable generalizability of testable mission scenarios.
  - A **user interface** to make effective use of these building blocks.
- Followed by stages of iteration between theoretical analysis and practical testing of the results. Goal: **Convergence towards a comprehensive theory of swarm mechanics.**

Swarm Definition

For the purposes of this discussion, a **swarm** shall be defined as a satellite formation consisting of the following components at each time t:

1. A **closed set of points** $\mathcal{R}_t \subset \mathbb{E}^3$ in space consisting of the region to be occupied by the swarm.
   - The region $\mathcal{R}_t$ is called "**the swarm envelope**" at time t.
2. A **prescribed distribution** of satellites (points $\{S_{1,t}, \dots, S_{n,t}\} \in \mathcal{R}_t$ at time t, where n is the number of satellites) with positions $\vec{r}_1(t), \dots, \vec{r}_n(t)$ relative to the center of the Earth (point C).
3. A **swarm centroid** $O_t \in \mathbb{E}^3$ whose position relative to C is given by the prescribed **swarm centroid functional** $\vec{r}_O(\vec{r}_1, \dots, \vec{r}_n)$.
   - The point O_t is also called "**the swarm center**" at time t.
   - Nominally, $\vec{r}_O(\cdot)$ is a **central tendency** (e.g., mean, median, etc.).
4. A **prescribed trajectory** $\vec{r}_O^*(t_f)$, defined for all time $t_f \geq t$, which yields the desired position of the swarm center at time t_f. A principle goal of this analysis is to **assess difficulty in maintaining** $\|\vec{r}_O(t) - \vec{r}_O^*(t_f)\| = \mathbf{0}$.

Swarm Definition



Assumptions

Examples of simplifying assumptions that can be made:

1. Swarm envelope assumptions:
   - **Avoiding the use of non-convex geometry**.
   - Taking advantage of geometry that can be well-described by **relatively few parametric variables** (e.g., a sphere).
2. Prescribed distribution of satellites:
   - Restricting satellites to **level-sets** (e.g., the surface of the swarm envelope, pre-defined Lyapunov functions).
   - **Imposing homogeneity** or even-spacing requirements between satellites.
3. Swarm centroid:
   - Using a centroid functional with **permutational symmetry** among the order of its arguments – i.e.,

$$\vec{r}_O(\vec{r}_1, \dots, \vec{r}_n) = \vec{r}_O(\vec{r}_{i_1}, \dots, \vec{r}_{i_n}) \; \forall \; (i_1 \in \{1, \dots, n\}) \cap \cdots \cap (i_n \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{n-1}\}).$$

   - Using a centroid functional that is **equivalent to identity** when there is only one argument, or when all arguments are equal in value.
   - **If $\mathcal{R}_t$ is convex**, using a centroid functional for which $\vec{r}_O(\vec{r}_1, \dots, \vec{r}_n) \to O_t \in \mathcal{R}_t$.

Networked Architecture

At present, a satellite swarm architecture is one in which:

1. Every satellite in the swarm can **communicate** with every other satellite in the swarm…
   a) either **directly** along a single line of sight with the respondent,
   b) or **indirectly** by communicating along a chain of nearby swarm satellites.
2. Every satellite in the swarm can perform **parallel calculations** for every other satellite in the swarm, provided that…
   a) the satellite being requested to perform the calculations **is available** (i.e., is not currently performing calculations of its own or for another satellite),
   b) or the satellite being requested to perform the calculations **is trustworthy**.

- Signal processing time between satellites is assumed to be constant or negligible.
- Transmission time between satellites i and j is given by the approximation

$$t_t \approx \left\| \vec{r}_j(t_0) - \vec{r}_i(t_0) \right\| / c$$

where c is the speed of light and t_0 is the time at which the signal was sent.

Networked Architecture Scenario

Three satellites (*A*, *B*, and *C*) are present in a satellite swarm. An adversarial entity electronically attacks satellite *C*, which results in the following effects:

- **All stored logs of position and velocity are deleted**, leaving the satellite without any ability to perform inertial navigation.
- The satellite is commanded to **apply a randomly directed impulse** to itself, then **stop communications with all other satellites** – including GPS and other members of the swarm – hindering efforts to reacquire it.

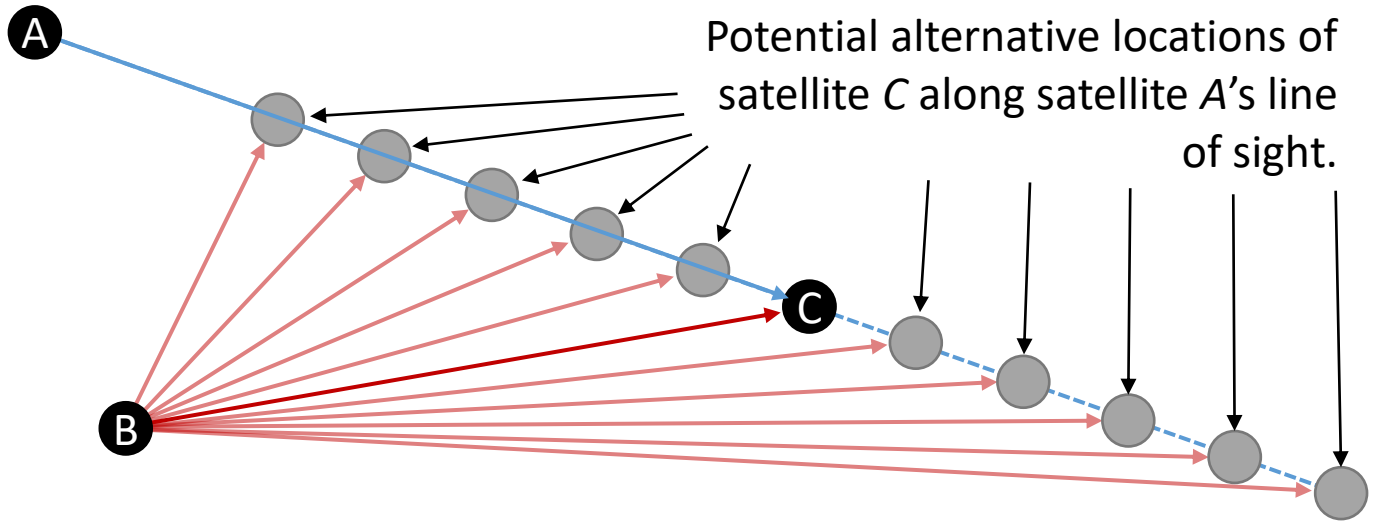
Detecting foul play, a signal is sent to satellite *C* requesting a state update and ordering a return to the formation. There is no response. **A command is sent to the swarm to acquire the orbital parameters of satellite *C*.**

Through the networked architecture, following chain of events is triggered:

1. Being the satellite in nearest proximity to its last known position, satellite *A* points its camera to the last known position of satellite *C* and scans the surrounding region in a spiral pattern until a visual acquisition is made.
2. This step can be repeated for satellite *B*, thus obtaining a position and velocity fix through parallax.

Networked Architecture Scenario

If, for any reason, satellite *B* (or, equivalently, satellite *A*) is unable to acquire an object, or there is uncertainty as to whether satellites *A* and *B* have located the same object (for instance, if the lines of sight do not pass within a certain parallel distance of one another), then **it is possible to obtain a viewing direction for satellite *B* from the data obtained by satellite *A*** (or a viewing direction for satellite *A* from data obtained by satellite *B*) by obtaining a position estimate of satellite *C*.



Networked Architecture Scenario

An algorithm designed to obtain the **range and range rate** of satellite C relative to satellite A utilizes multiple optical images taken over a span of time (e.g., 10 pictures taken over 30 seconds). Each image defines a line of sight between the satellite and object:

1. For each line of sight i, $\dot{s} = f_i(s)$ is produced which yields the range rate as a function of range such that **the eccentricity of the resulting orbit is minimized**.
2. For each line of sight i, every local minimum that occurs in $f_i(s)$ is saved as a **candidate pair** of range and range rate, provided that
   - $s > s_{\min}$, where $s_{\min} > 0$ is a **prescribed minimum range**, and
   - $e\big(s, f_i(s)\big) < 1$, where $e(\cdot,\cdot)$ produces the **eccentricity** of the resulting orbit.

Steps 1 and 2 are fast compared to the latter stages of the algorithm and may be handled directly by satellite A during the imaging process.

Networked Architecture Scenario

Once all images have been taken, every line of sight scanned, and resulting candidate pair identified, the algorithm moves forward to step 3:

3. Each candidate pair is supplied as the initial condition to a numerical optimizer which attempts to **produce an orbit that matches the relative angles and angular rates** of satellite C relative to satellite A. The degree to which these match is called "**plausibility**."
   - **Requires known directions for all other lines of sight**, thus cannot be performed concurrently during the imaging.
   - Requires numerical optimization of cost functions with term complexity that **increases with the square of the number of images**.
4. Return the most plausible resulting orbit.

Step 3 has the potential to take the greatest amount of time if there are many candidate pairs to evaluate. Furthermore, since this process cannot be run concurrently with that of collecting images, any additional time spent on this evaluation is potential response time lost. **Networked architecture provides a solution to this problem** by allowing step 3 to be delegated to other satellites in the swarm.

Stocking the Framework Toolkit

Where theory grapples with practicality:

- Identifying dynamics that satisfy swarm design constraints and assumptions invariably require some form of **nonlinear cost function optimization**.
- In order to maintain generality, our approach to handling optimization must also be as general as possible.
- Specifying the MATLAB® as our primary tool, which offers optimization tools that use first and second order methods, we sought a toolkit to produce **the gradients and hessians of input variables** under any number of arbitrary transformations comprised of MATLAB® -intrinsic functions.
- The Drv-class mimics MATLAB® 's float arrays but stores the gradients and hessians of each element "underneath." These derivatives are updated across operations (e.g., $C = A \times B$ performs the product rule).

Next Steps

- Full completion of Drv-class library.
- Simulation of outlined scenario.
- Proof of swarm stability.
- Bounds of control effort required to maintain swarm geometry.