Cluster Keeping Algorithms for the Satellite Swarm Sensor Network Project

Eviatar Edelman and Pini Gurfil
Technion – Israel Institute of Technology, Haifa 32000, Israel
eviatar@technion.ac.il; pgurfil@technion.ac.il

Abstract
This paper develops cluster control algorithms for the Satellite Swarm Sensor Network (S3Net) project, whose main aim is to enable fractionation of space-based remote sensing, imaging, and observation satellites. A methodological development of orbit control algorithms is provided, supporting the various use cases of the mission. Emphasis is given on outlining the algorithms structure, information flow, and software implementation. The methodology presented herein enables operation of multiple satellites in coordination, to enable fractionation of space sensors and augmentation of data provided therefrom.

1. INTRODUCTION
In the field of earth observation from space, modern approaches show a trend of moving away from the classical single satellite missions, where one satellite includes a complete set of sensors and instruments, towards fractionated and distributed sensor missions, where multiple satellites, possibly carrying different types of sensors, act in a formation or swarm. Such missions promise an increase of imaging quality, an increase of service quality and, in many cases, a decrease of deployment costs for satellites that can become smaller and less complex due to mission requirements. Need for incremental deployment can be caused by funding limitations or issues. Although a combination of funds from different interest groups is possible, it can be difficult to achieve. A higher degree of independence for service providers can be enabled through lower cost satellites that allow integration into satellite swarms.

Besides the reduction of costs, observation quality can increase through the combination of different measurement sources. Incremental deployment of different measurement sources can lead to synergy effects for the imaging/observation quality, e.g. elimination of radar or optical shadow effects by observing the same region with multiple satellites from different angles. Incremental deployment of different measurement sources can lead to improved quality of service, e.g. single pass interferometry with 2 SAR satellites can result in the reduction time for image delivery from multiple days (due to the delay of the second pass) down to hours (e.g. Tandem-X).

Still, there are major technical challenges such as interfacing, communication within the swarm constellations and to ground stations, synchronization, formation flying, precision thrusting and pointing as well as the realization of sensor fusion approaches. Furthermore, only little research has been conducted in terms of effectiveness of future developments in the field of fractionated and distributed sensors for earth observation from space respecting the feasibility of sensor fusion approaches and
also formation flying approaches needed for the distributed sensor mission scenarios. These approaches are strongly dependent on communication and on-board payload-processing capabilities and needs, on synchronization techniques between swarm satellites, as well as strong limitations on fuel needed for precision thrusting and pointing strategies of satellites.

Applications such as Hyperspectral Imaging, High-Resolution Optical Imaging, and Synthetic Aperture Radar (SAR) offer an abundant amount of information about conditions, objects and materials on earth and in the atmosphere. Sensor fusion approaches are a promising tool for enabling significant advances in performance of these applications in terms of increases of observation quality/accuracy and therefore an increase of the confidence level about retrieved information regarding conditions, materials and objects.

The Satellite Swarm Sensor Network (S3Net) project has been established to work in parallel on key enablers required to develop the efficient use of fractionated satellite swarms. One of the objectives of S3Net is to achieve breakthrough progress in formation flight algorithms, by proposing cluster control methods, which will permit sharing resources across the cluster network with real-time guarantees, including distributed data processing, optimal resource allocation and distributed optimal estimation under time delays; semi-autonomous long-duration maintenance of a cluster network, including addition/incremental deployment and removal of spacecraft modules to/from the cluster network under collision avoidance constraints; and autonomously reconfiguring the cluster to retain critical functionality in the face of network degradation, component failures, or space debris damage.

In this paper, we present the cluster keeping methodology of S3Net, designed for both optical and SAR use cases. In each case, the unique constraints on formation geometry and maintenance, as well as the related computational and implementation considerations, are analyzed, and a corresponding cluster control algorithm is designed and tested in a high-fidelity simulation.

2. FORMATION FLYING REQUIREMENTS

Table 1 summarizes the formation flying requirements of the different use cases in S3Net. In all cases the semimajor axis (SMA) is about 7070 km, and the orbit is near-circular and Sun-synchronous. The local time of the ascending node (LTAN) is different; this is related to the launcher, and has a limited effect on the formation flying algorithms. The different use cases are based on satellites with the same mass range and similar electrical power. From the formation flying perspective, the main differences stem from the need to have along-track and cross-track formations. The along-track clusters such as the multispectral and ocean currents monitoring can use the same control algorithm. The along-track baseline and allowed tolerances can be used as an input for this control algorithm. The two cross-track formations cannot be grouped due to the large difference in the required cross-track baseline. In this case, two different control algorithms are needed. Overall, this paper presents three control algorithms that satisfy the requirements in Table 1.
<table>
<thead>
<tr>
<th>Use case #</th>
<th>Optical</th>
<th>Radar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Panchromatic</td>
<td>Multispectral</td>
</tr>
<tr>
<td>Number of Satellites</td>
<td>3</td>
<td>3-5</td>
</tr>
<tr>
<td>Orbit Type</td>
<td>Near circular</td>
<td>Sun-synchronous low Earth orbit</td>
</tr>
<tr>
<td>Orbit altitude</td>
<td>693 km</td>
<td></td>
</tr>
<tr>
<td>Orbit Inclination</td>
<td>98.16°</td>
<td></td>
</tr>
<tr>
<td>Light Conditions</td>
<td>Observation in light LTAN 10 : 30 ± 30 min</td>
<td>Dawn-dusk LTAN 06 : 00 ± 30 min</td>
</tr>
<tr>
<td>Formation Configuration</td>
<td>Cross-track cluster</td>
<td>Along-track cluster</td>
</tr>
<tr>
<td>relative position</td>
<td>17.5 km ±10%</td>
<td>7 km ±25%</td>
</tr>
<tr>
<td>Satellite dry Mass</td>
<td>100 kg</td>
<td>200 kg</td>
</tr>
<tr>
<td>Satellite power</td>
<td>250-300 W</td>
<td>110 W (Payload power consumption)</td>
</tr>
<tr>
<td>Satellite size</td>
<td>40 × 40 × 40 cm</td>
<td>85 cm × 120 cm × 157 (height) cm (stowed, estimated envelope)</td>
</tr>
<tr>
<td>Active time (%)</td>
<td>50%</td>
<td>maximum 10%</td>
</tr>
</tbody>
</table>

Table 1: Formation flying requirements summary

3. ORBIT AND ATTITUDE DETERMINATION ALGORITHM STRUCTURE

Soon after the separation from the launcher, the satellites will initiate their cluster-control and maintain it throughout the mission. On each satellite, the cluster control will be carried out by the Orbit Control Module (OCM), with both hardware and software components. The OCM software is in charge of a number of tasks, such as orbit determination and mean orbital elements estimation of the host and other satellites from GPS measurements; determination of the relative positions and velocities of all the satellites in the cluster; management of the orbit and cluster control; and delivery of the relevant inputs and commands to the attitude determination and control system (ADCS). The structure and the interfaces of the OCM are shown in the block diagram of Fig. 1. The OCM is composed of three main sub-modules: orbit sub-module, inter-sat sub-module and the control sub-module.

3.1 ORBIT SUB-MODULE

The orbit sub-module is responsible for determination of position, velocity and osculating orbital elements from GPS measurements, determination of mean orbital elements, and performance of various auxiliary calculations (e.g. the sun vector).

The Orbital Kalman Filter (OKF) algorithm estimates the absolute orbit by the online processing of the respective GPS data. The GPS fixes, obtained by the host's onboard GPS receiver, include position, velocity and time (PVT) in the conventional ECEF format, and GPS time tags. The Extended Kalman Filter (EKF) algorithm consists of two parts, as shown in Fig. 2: time update and measurement update.
At the time update stage, the satellite state vector and state covariance matrix are propagated to the time of the latest PVT set, to calculate their a-priori estimates. The orbit propagation is carried out by the numerical integration of the differential equations

\[
\ddot{r} = \mathbf{P}_g + \mathbf{P}_{ng} + \mathbf{P}_{emp} + \mathbf{P}_{cont}
\]

where \( \mathbf{r} \) is the position vector of the satellite center of mass referred to the J2000 ECI frame; \( \mathbf{P}_g \) is the acceleration vector due to the gravitational forces acting on the satellite; \( \mathbf{P}_{ng} \) is the acceleration vector due to the non-gravitational (surface) forces; \( \mathbf{P}_{emp} \) denotes empirical accelerations, accounting for the unmodeled perturbations; and \( \mathbf{P}_{cont} \) is the control acceleration vector created by the onboard actuators.

The gravitational accelerations include the effects of the geopotential; gravity of the Sun, Moon, and major planets; solid Earth tides; ocean tides; and general relativity. The non-gravitational accelerations taken into account are atmospheric drag and solar radiation pressure. In order to calculate the non-gravitational accelerations precisely, the satellite cross-sections with respect to the relevant directions are to be determined using the information from ADCS, and an analytic mapping algorithm [1] that calculates the satellite's projected cross-sectional area in real time. To start the orbit propagation, the
first available PVT data set should be taken as the initial state vector. The backup option to calculate the initial conditions and propagate the orbit is represented by the SGP4 orbit predictor [2], assuming the availability of the relevant TLE files. At the measurement update stage, both the satellite state vector and state covariance are corrected by the latest GPS fix, to obtain their a-posteriori estimates. The OKF output includes time as well as estimated position and velocity in the ECI frame, and osculating orbital elements. The basic models employed in the orbit propagation are mostly those recommended by [3].

<table>
<thead>
<tr>
<th>Basic models employed in the orbit propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td>coordinate transformations</td>
</tr>
<tr>
<td>gravity field</td>
</tr>
<tr>
<td>ephemerides of the Solar system major bodies</td>
</tr>
<tr>
<td>solid Earth tides</td>
</tr>
<tr>
<td>ocean tides</td>
</tr>
<tr>
<td>atmospheric density</td>
</tr>
</tbody>
</table>

One option is to use the Zhong-Gurfil [4] mean element estimator (MEE), which is a formulation that transforms raw GPS measurements into mean orbital elements using a spherical-simplex square-root unscented Kalman filter. Despite the superior performance of this algorithm, the OCM computational load can be reduced by using a first-order mapping based on the Brouwer–Lyddane theory [5, 6]. The implementation of this algorithm can be found in [7]. The input is the estimated orbital elements from the OKF, and the output is the mean orbital elements.

The auxiliary algorithms include the Sun Vector (SV) and Earth Magnetic Vector (MGV) algorithms involved in the ADCS operation. The SV algorithm calculates the topocentric Sun vector, referred to the ECI frame, by using the satellites geocentric state vector as estimated by the OKF, and the data from the JPL planetary and lunar ephemeris as inputs. The MGV algorithm calculates the geomagnetic field vector in the ECI frame at a given point along the satellites orbit, using as inputs the satellites in-orbit position and the international geomagnetic reference field (IGRF) model [8], represented by the Gaussian coefficients and their time derivatives. The block diagrams of both algorithms are shown in Figs. 3(a) and 3(b), respectively.
3.2 INTER-SAT SUB-MODULE

The inter-sat sub-module processes data obtained from the inter-satellite communication. The inter-sat sub-module uses the same OKF and MEE, but the inputs for the OKF are estimated PVT, obtained from the inter-satellite link. The inter-sat sub-module OKF uses the drag cross-section area and the sun cross-section area of the other satellites, as received via inter-satellite communication. In case of missing data, the inter-sat sub-module will keep propagating the other satellite’s PVT using the previous step output as the current step input. This is shown in Fig. 4, where the host satellite is marked by “A”, and the two others by “B” and “C”. If the time since the last inter-satellite update crosses a predetermined threshold, then this data will be replaced by the most updated TLE data.

3.3 CONTROL SUB-MODULE

The control sub-module determines thruster commands to control the distance among the satellites and manages the active cluster control law, i.e. switching between the Cluster Control Algorithm CCA, Differential Drag Algorithm DDA, and contingencies. Since even initially close satellites, without active control, will gradually drift apart because of the differential accelerations, the cluster control algorithm [9, 10, 11] has two tasks to perform: cluster establishment at the initial phase of the mission, and cluster keeping throughout the whole mission, which determines the satellite allowed relative drift within pre-specified upper and lower limits.
The cooperative cluster-keeping controller, based on the control-Lyapunov function, steer the mean orbital elements as defined by the cluster orbital dynamics, to their desired values. The closed-loop control law operates in an on-off mode, turning on the relevant thruster, once a certain inter-satellite distance is about to exceed the limits, and turning it off otherwise. The block diagram of the CCA algorithm is shown in Fig. 5(a). The inputs to CCA are position and mean orbital elements – from the OKF and MEE algorithms, and relative position and velocity vectors with respect to the other satellites of the cluster – from the inter-sat sub-module. The outputs from CCA include commands to the ADCS and commands to the propulsion module.

Being a backup for the CCA algorithm, the DDA [12, 13] is aimed at keeping the relative distances between the satellites inside the prescribed limits by a proper variation of their effective cross-sectional areas perpendicular to their respective velocity vectors, in order to change their atmospheric drag accelerations, and thus cause the required changes in their semi-major axes and in-orbit positions. The block-diagram of the DDA algorithm is shown in Fig.5(b). The inputs to the DDA are the mean orbital elements of all the satellites from the MEE algorithm, and the output is the command to the ADCS actuators of the host satellite.

Loose cross-track clusters can be controlled by using the Earth geopotential to achieve the required cross-track separation with a limited amount of $\Delta V$. Tight cross-track clusters can be controlled by using eccentricity/inclination vector separation.

![Block diagram of the cluster control algorithm](image1)

![Block diagram of the differential drag algorithm](image2)

(a) Block diagram of the cluster control algorithm  (b) Block diagram of the differential drag algorithm

Figure 5: Cluster control algorithms

4. TESTING THE ORBIT CONTROL MODULE

The problem of simulating the orbital dynamics of a single satellite has well-known solutions [14]. Different methods can be applied to match the different design phases, from commercial-off-the-shelf software to specific mission end-to-end simulations. Commercial software such as GMAT\(^1\), STK\(^2\) or FreeFlyer\(^3\) can be used in the early design stage of the mission when rapid change of the code is expected. These programs offer well-tested environmental models and propagators, but might not be suited to testing the operational code. We have developed a simulation in the C language, which offers the ability to test

\(^1\) https://gmat.gsfc.nasa.gov/
\(^2\) https://www.agi.com/
\(^3\) https://ai-solutions.com/freefly
the operational code by embedding it into the physical modules of the simulation. A diagram of the simulation is shown in Fig. 6. The simulation includes both orbit and attitude modules, and the output of these modules is being used for the different sub-modules. The simulation can provide the sensor data, battery status, dynamic communication link budget and additional information. Based on the required accuracy and the required outputs, the simulation can include different perturbation models and simulate different sub-systems.

Figure 6: A block diagram of an orbit and attitude simulation

A challenge arises when testing a mission with multiple satellites that perform distributed closed-loop control using data coming from other satellites. To evaluate the performance of these algorithms, it is important to simulate the interface between the different agents in the formation in a realistic way. This problem was discussed during the TechSat-21 mission [15]. The proposed solution was to create a simulation environment that included computers for simulating the ground station and the space environment, and the cluster of satellites was supposed to be replaced by multiple flight processors. Some prototypes were developed, but this mission was canceled in the early 2000s. A similar solution was discussed in [16], where the on-board computer (OBC) was replaced by a virtual machine that emulates the OBC. Based on previous experience and by utilizing modern technologies we solved this challenge by developing a testing methodology that includes three phases. An overview of these three phases can be found in Fig. 7.

In the first phase, we tested the entire cluster on a single PC. In this phase, the simulation and the satellites share the same memory, and the inter-satellite communication effect is neglected. This step utilizes a regular workstation that can provide the required computational power needed to operate the simulation and multiple satellites together. In this phase, algorithms are easily tested and can be change by demand.

In the second phase, we created a multi-thread simulation that includes one thread for the simulation and multiple threads for the satellites. Each thread has its own memory, and a shared memory is used to share data between the threads. A synchronization mechanism was developed to avoid simultaneous access to the shared memory. In this phase, we use parallel computing capabilities that are built in the Linux operating system. Debugging the code in this phase is less convenient, but the multi-thread simulation is as close as it gets to the operational scenario without using actual flight hardware. In the third step, we used the same simulation to feed three identical OBCs.
Each OBC contains a fully-functional copy of the flight software. The OBCs use inter-satellite communication transceivers to share data, and simulation data are sent from the external simulation using UART. The simulation gets as an input the control acceleration created by the control algorithms, and uses it to update the satellites state. In this phase, timing and synchronization become a major issue, because the OBCs use a real-time operating system. Without a proper synchronization mechanism, this configuration cannot work. This issue was discussed in [14, chapter 7] and several different approaches were offered.

Two different times source are active on the OBC. The first one is the OBC real-time clock that counts a “tick” every 25 msec. The counting starts at the moment the satellites OBC starts running. This counter can be reset to zero to allow time synchronization with GPS. The second time source is the GPS clock. This clock provides universal time, and this time is synchronized among all 3 satellites. It is possible to use the 1 pulse per second (PPS) provided by the GPS for synchronization. The PPS signal is accurate enough [17] for this kind of an application.

---

**Figure 7: Testing methodology for distributed control algorithms**

---

5. **CONCLUSIONS**

A methodological design enabling cluster flight and cluster keeping of a swarm of cooperative satellites has been presented. This design methodology supports all use cases relevant for fractionated sensors. Implementation issues include implementation on a new simulator, capable of supporting development of software modules and synchronization of time among each satellite.

6. **ACKNOWLEDGMENTS**

This work was supported by the European Commission Horizon 2020 Program within the framework of the S3Net project under grant agreement 687351.
7. REFERENCES


