16th International Conference on Space Operations, Cape Town, South Africa - 3 - 5 May 2021.

SpaceOps-2021,7,x1297

SENSOR FUSION KALMAN FILTERING FOR STABILITY AND CONTROL OF SATELLITE SWARMS

Dr. Rahul Rughani[1] and David A. Barnhart[2]

[1] *Department of Astronautical Engineering, University of Southern California Information Sciences Institute and Space Engineering Research Center, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292,* rughani@usc.edu
[2] *Department of Astronautical Engineering, University of Southern California Information Sciences Institute and Space Engineering Research Center, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292,* barnhart@isi.edu

Abstract

With the emergence of the space servicing sector, along with the return of manned missions beyond low earth orbit, there is a need for quick, efficient, and most of all, safe Rendezvous and Proximity Operations (RPO). More than that, the next big step forward may be manufacturing in space, which will involve large swarms of spacecraft cooperating in close proximity to each other, all subjected to the same laws of orbital mechanics. Currently, there is a lack of knowledge about how to safely operate a swarm of spacecraft in close quarters in a dynamically changing environment (i.e., a "space construction site"), without creating a high risk of collision and potential debris creation. Satellite swarms are groups of spacecraft that operate cooperatively to perform a common task or goal in orbit. Examples of this are in-space assembly of structures, manufacturing, cooperative sensing platforms, on-orbit servicing, asteroid mining, etc. Swarms would be operating in relative-motion orbits, working around a common point in space, within a few kilometers of each other. Such swarms would share information between each member of the swarm, allowing them to have a shared situational awareness of the entire mission. Although not all members of the swarm may have the same functionality, i.e. some may be communication nodes, others may have robotic arms, and yet others may have scanning cameras and LIDARs, it is expected that they will all have propulsive capabilities to allow them to maneuver in relative orbits around each other and/or a common target. Once a set of orbits are created for a swarm to perform is operations, primarily done by taking into account the individual tasks of the various members of the swarm while keeping collision avoidance in mind, these orbits need to be maintained. Unlike a single spacecraft operating alone, a spacecraft swarm cannot rely on inertial navigation methods or large navigational systems such as GPS, as these methods cannot provide the high-fidelity data required to determine the relative position of the members of the swarm with respect to each other. Instead, a system of relative positions sensors, along with data from RF communications between the spacecraft can be used. Upon receiving this wealth of data from various sensor inputs, each of varying resolution and quality, this data can be parsed in real-time by the on-board system on each spacecraft in the swarm, and then fed into a sensor fusion Kalman filter, also aboard the spacecraft, to maintain the relative motion orbit of the swarm in the face of gravitational perturbations

Keywords: Satellite, Swarm, Sensor Fusion, Rendezvous, Kalman Filter

Nomenclature

CLIENT .... Satellite or Platform to be Serviced

SERVICER .... Satellite or Platform that provides Service

SWARM .... Group of two or more spacecraft cooperating towards a common task or goal

Acronyms/Abbreviations

C-W .... Clohessy-Wiltshire

GEO .... Geostationary Earth Orbit

GNC .... Guidance Navigation and Control

LEO .... Low Earth Orbit

LVLH .... Local Vertical Local Horizontal

OOS .... On Orbit Servicing

RPO .... Rendezvous and Proximity Operations

SFKF .... Sensor Fusion Kalman Filter

1 Introduction

Given the recent advancements in satellite servicing technologies and space robotics [1–7], the collective capabilities of the space industry as a whole are moving towards in-space manufacturing. The ability to manufacture or assembly anything in space using robotics is a crucial technology for deep space exploration and the eventual colonization of Mars and beyond, as current platforms are limited to the volume and mass constraints of a single rocket fairing. Although the existing On-Orbit Servicing (OOS) methodology employs the use of large, monolithic robotic spacecraft; the method of OOS and in-space manufacturing proposed in this research study considers taking the leap from using one spacecraft to a swarm of dozens, potentially even hundreds, of small robotic spacecraft. These *swarms* of spacecraft would operate like a colony of bees, each individual member of the swarm performing its own dedicated task, culminating in the successful execution of a large and complex operation [8].

To control a large number of spacecraft cooperating in close proximity, each member of the swarm must be able to maneuver on its own, as well have the capability to sense the position and velocity of other nearby members of the swarm to communicate and avoid collisions if anything goes wrong [9]. To streamline this process, for a given swarm function a set of optimized trajectories needs to be determined such that there are no predicted collisions between the spacecraft for a prescribed amount of time, barring any malfunctions, and the ΔV required to insert to the swarm is minimized.

There has been some prior research into the use of safety metrics and collision avoidance for RPO in Earth orbit [10]; however, no previous research has sought to combine the use of all of these for swarm operations. For example, Gaylor, Brent, and Barbee [11] outline a framework for safe rendezvous algorithms using safety ellipses and linear optimization techniques, but it is applied only to one-on-one RPO, and is not suited for swarm operations. Izzo and Pettazzi [12] analyze the use of equilibrium potential functions to determine optimal orbits for satellites in the swarm to avoid collisions. Lopez and McInnes [13] employ virtual vector fields to control the final approach trajectory during RPO, but only for one-on-one satellite rendezvous. Slater, Byram, and Williams [14] create probability functions to determine the collision risk of members of satellite formations with foreign objects or drifting members of the formation. However, they do not set up the formation (swarm) in a way that reduces the probability of these collisions in the first place. Ross, King, and Fahroo [15] investigate optimization techniques for formations and swarms, though they concentrate on propellant optimization rather than safety and collision avoidance optimization.

This paper will cover the use of Sensor Fusion and Kalman Filtering to accurately determine the relative states of all spacecraft in a swarm, in real-time, accounting for deviations due to errors and perturbations.

2 Swarm Architecture

The goal of swarm Rendezvous and Proximity Operations (RPO) is to enable cooperative on-orbit construction and assembly projects, allowing for the creation of large structures in space while potentially using in-situ resources. These trajectory generation and verification methods are designed to minimize the overall risk of collisions between any spacecraft in the swarm, or with any object that is in close proximity to the swarm. The methodological solutions presented in this paper use Genetic Algorithms to evolve a set of initial conditions into a viable and efficient solution, while also applying a Sensor Fusion Unscented Kalman Filter to predict the relative positions of the spacecraft for real-time collision detection and avoidance. These algorithms result in a set of trajectories for each spacecraft that enable it to achieve its mission goals within its ΔV budget, while also leveraging the combined sensor data of the entire swarm to accurately determine the relative positions between each spacecraft to a higher precision than GPS data alone.

2.1 Trajectory Generation

Genetic Algorithms (GAs) can be used to converge on an optimal set or family of trajectories for each member of the swarm [8]. These trajectories will be such that each member can perform their required individual actions, while minimizing the fuel required for maneuvering and also avoiding conjunctions, to a prescribed probability of collision, for a given amount of time. However, there are inherent errors in any system, and in order to react to them, autonomous algorithms, operating in real time, are required.

Running this for a set of 10 spacecraft, with zoning restrictions set out in Table 2.1 with respect to the Client spacecraft, and to avoid conjunctions within a 50 m buffer corridor from each spacecraft, Fig. 1 shows a set of closed and repeating relative motion trajectories that satisfy this criteria. The trajectories shown are for a 10 day propagation of the initial states determined by the GA solver, during which the Sensor Fusion Kalman Filter determined that there was no probability of collision within a 3-sigma covariance.

Table 1: Constraints for 10 Spacecraft Swarm Example

Swarm Member	**Constraint Ellipsoid Dimensions [km]**	**Constraint Ellipsoid Center [km]**
Spacecraft 1	[2,2,2]	[0,0,0]
Spacecraft 2	[2,2,2]	[0,-5,0]
Spacecraft 3	[3,3,3]	[0,2,0]
Spacecraft 4	[5,5,5]	[0,7,3]
Spacecraft 5	[2,2,4]	[0,10,0]
Spacecraft 6	[2,2,4]	[0,10,0]
Spacecraft 7	[2,2,4]	[0,-20,0]
Spacecraft 8	[5,5,5]	[0,30,0]
Spacecraft 9	[5,5,5]	[0,30,0]
Spacecraft 10	[5,5,5]	[0,30,0]

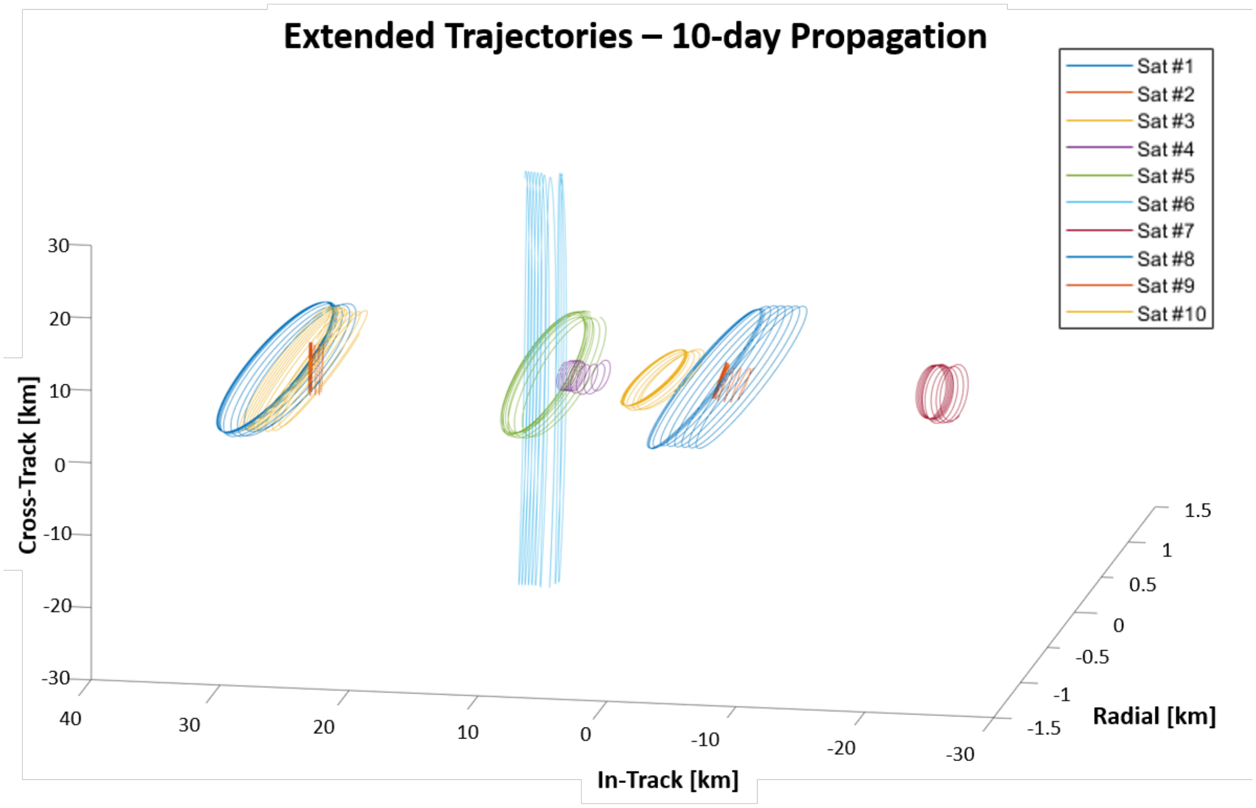

Fig. 1: Swarm Solution for 10 Spacecraft

3 Kalman Filtering

In real-world operations, it is impossible to know the position and velocity of a spacecraft with 100% precision. Position and velocity are measured using on-board sensors, which have inherent error tolerances. These errors, from inputs such as GPS and relative Radar ranging, result in a covariance matrix attached to the state vector for each spacecraft in the swarm. These covariance matrices can be computed not only with respect to an inertial state, but also between each spacecraft in the swarm. This means that if the covariance between spacecraft A and B is desired, sensor fusion between all other spacecraft and spacecraft B can be used to refine the state vector and covariance matrix of spacecraft B, thus minimizing the measurement error. By combining the measurements from multiple spacecraft, the measurement accuracy can be improved over a simple computation of the covariances on each spacecraft separately.

A Kalman filter can be used to reduce the error in an estimated state by propagating a set of points through time, each corresponding to the boundaries of the covariance "bubble of uncertainty" that surrounds the spacecraft. As this is propagated, the covariance ellipsoid is refined by using measurements taken from a sensor at a known position, with a known precision. This is used successively over time to predict what states are more likely, and which are less likely, using a weighted scheme to determine where the spacecraft lies within a 3-sigma Gaussian distribution. The Kalman filtering method is useful for not only simulations, but for real-time operations, since the computational cost of the algorithm is very low, and can be run in real-time onboard a satellite.

3.1 *Mathematical Formulation*

There are two common types of Kalman filters used in practice, an extended Kalman filter (EKF) and an unscented Kalman filter (UKF). The EKF, although more computationally efficient, requires that the Jacobian matrix of the equations of motion be known and well defined, which is quite complex to derive for the nonlinear perturbation equations of RPO. Instead, the UKF uses what is known as sigma-points, a set of virtual points surrounding the unknown object at a 3-sigma distance, which are then propagated using the equations of motion to determine the covariance drift over time. While this is more computationally intensive than modelling the covariance drift using a Kalman filter, it is less sensitive to nonlinear changes in the system, and can be computed in real-time without a-priori knowledge of the Jacobian matrix [16].

To run a step of a Kalman filter, first the previous state is needed. This can either be the initial state, or the end state of a previous step of the Kalman filter. We'll define these known quantities using $\hat{x}^+_{0,k-1}$ for the spacecraft position, and $\hat{x}^+_{i,k-1}$ for each of the sigma points. Note that there are two sigma points for each dimension of the problem, including position, velocity, and noise dimensions. The plus sign denotes that this is a corrected estimate, and thus has been run through a filter (or is the initial state). The $k-1$ indicates that it is from the previous time-step, and the hat denotes that it is an estimate generated by the filter, and not a raw measurement from a sensor. Over time, when using a Kalman filter, the estimate will converge to a minimal covariance offset from the truth value so long as the system remains observable [17]. This minimal covariance will depend on the accuracy of the measurement sensors, and the process noise variances – how much the measurements should be trusted above the model.

First, the points can be run through a propagation function, which in this case is the equations of motion for RPO with gravitational perturbations.

$$\hat{x}^-_{i,k} = f(\hat{x}^+_{i,k-1}, q_{i,k-1}) \tag{1}$$

Here, $q_{i,k-1}$ represents the process noise, which are unknowns that affect the propagation. Typically, in terrestrial applications, this is attributed to wind or other variable sources. There are few of these sources in LEO, though this could be used to model aberrations in atmospheric drag, or thruster variations during a maneuver. For our purposes, the process noise is left as zero for simplification.

Once the sigma points have all been calculated and propagated, the mean predicted state is computed as a weighted average of all the sigma points.

$$\hat{x}^-_k = W_0\hat{x}^-_{0,k} + W\sum_{i=1}^{n}\hat{x}^-_{i,k} \tag{2}$$

Where W_0 and W are the weights associated to the middle point and the sigma points, respectively. Using this weighted information, the sample covariance can be computed around this new covariance, with the covariance weights W_c and W, similar to the mean predicted state

$$P^-_k = W_c(\hat{x}^-_{0,k} - \hat{x}^-_k)(\hat{x}^-_{0,k} - \hat{x}^-_k)^T + W\sum_{i=1}^{n}(\hat{x}^-_{i,k} - \hat{x}^-_k)(\hat{x}^-_{i,k} - \hat{x}^-_k)^T \tag{3}$$

The next step, now that the estimated position and covariance has been computed, is to use this knowledge to attempt to correct the estimated position, to get our predicted state. To do this, the overall predicted measurement $\hat{z}_k$ is computed

$$\hat{z}_{i,k} = h(\hat{x}^-_{i,k}, r_{i,k}) \tag{4}$$

Where $r_{i,k}$ is the *measurement noise* with covariance $\boldsymbol{R}$, a property of the sensors in use. Then the weighted mean of this predicted measurement is computed:

$$\hat{z}_k = W_0 \hat{z}_{0,k} + W \sum_{i=1}^{n} \hat{z}_{i,k} \tag{5}$$

This predicted measurement is then used to correct the state using y_k, which is the *innovation vector*, or residual. This is the vector pointing from the predicted measurement to the actual measurement, which is scaled using $\boldsymbol{K}$ the *Kalman gain*.

$$y_k = z_k - \hat{z_k} \tag{6}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K y_k \tag{7}$$

The Kalman gain is a scaling matrix that enables mapping of the estimates to the true location, based on the covariance matrices. Thus, to compute the predicted measurement, and move forward to the next time-step in the filter, the Kalman gain must first be computed. This is done by first computing the weighted sample covariances of both the estimated measurement value with the estimated position, and the estimated measurement value with itself.

$$P_{xy} = W_c(\hat{x}_{0,k}^- - \hat{x}_k^-)(\hat{z}_{0,k}^- - \hat{z}_k^-)^T + W \sum_{i=1}^{n} (\hat{x}_{i,k}^- - \hat{x}_k^-)(\hat{z}_{i,k}^- - \hat{z}_k^-)^T \tag{8}$$

$$P_{yy} = W_c(\hat{z}_{0,k}^- - \hat{z}_k^-)(\hat{z}_{0,k}^- - \hat{z}_k^-)^T + W \sum_{i=1}^{n} (\hat{z}_{i,k}^- - \hat{z}_k^-)(\hat{z}_{i,k}^- - \hat{z}_k^-)^T \tag{9}$$

The Kalman gain is then the ratio of these two covariance matrices

$$K = P_{xy} P_{yy}^{-1} \tag{10}$$

Finally, the covariance values need to be corrected, similar to how the position estimates were corrected, in order to be used in the next time-step of the filter.

$$\hat{x}_{i,k}^+ = \hat{x}_{i,k}^- + K(z_k - \hat{z}_{i,k}) \tag{11}$$

The new covariance matrix can be computed as a sample covariance of the corrected sigma points, P_k^+:

$$P_k^+ = W_c(\hat{x}_{0,k}^+ - \hat{x}_k^+)(\hat{x}_{0,k}^+ - \hat{x}_k^+)^T + W \sum_{i=1}^{n} (\hat{x}_{i,k}^+ - \hat{x}_k^+)(\hat{x}_{i,k}^+ - \hat{x}_k^+)^T \tag{12}$$

Which can be simplified to be:

$$P_k^+ = P_k^- - K R_k K^T \tag{13}$$

Where R_k is the covariance of the measurement error, taken from the datasheet of whatever sensor is in use (in this case a radar or LIDAR sensor). The covariance values can then be used to generate an error ellipse, or *egg of death*, around the spacecraft, which provides the volume of space within which we expect to find the spacecraft with 3σ precision.

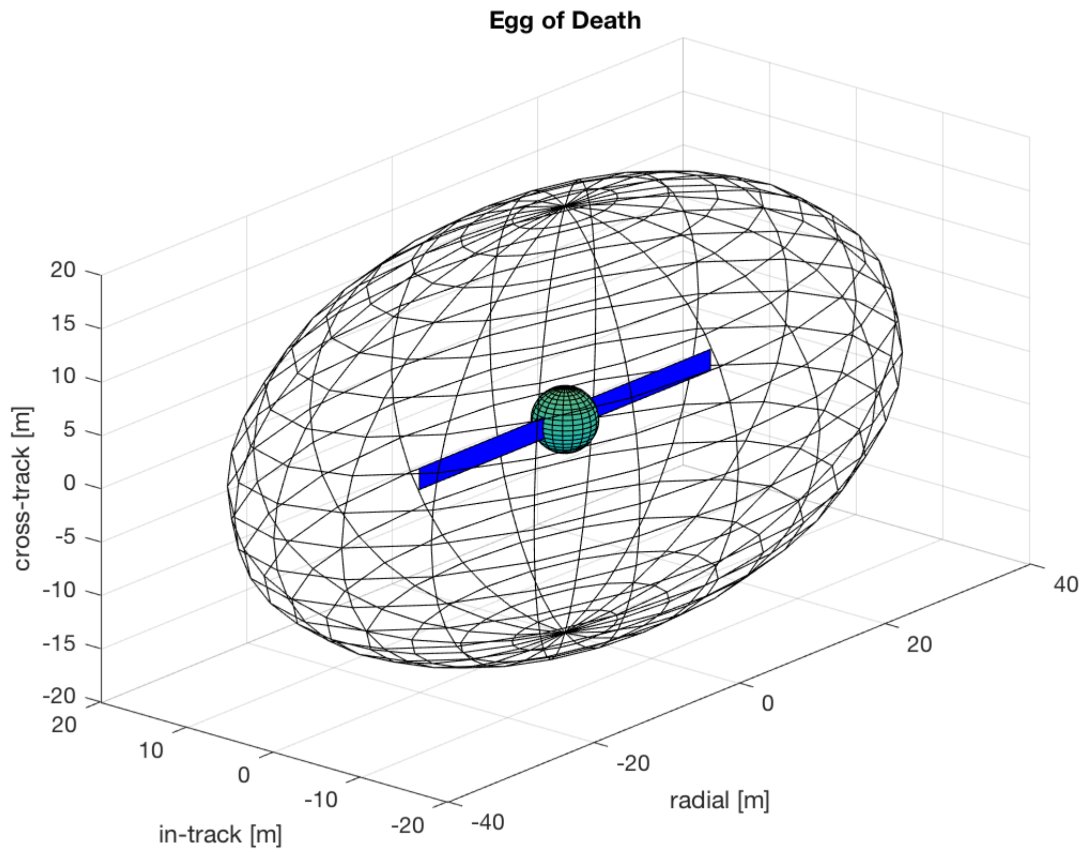

Fig. 2: Example Error Ellipse for a Satellite

Figure 2 above shows an example egg of death around a satellite. The size of the egg is primarily based on two factors; the measurement error from the radar ranging system used to determine the position and velocity of the satellite, and the propagation of error between measurements. The measurement error (covariance) itself cannot be removed, but it can be compensated for and narrowed down using a Kalman filter to sort out readings that are not very likely given the physics of orbital mechanics.

3.2 Sensor Fusion Kalman Filter

In order to take into account the shared data of the swarm, which is the combination of the radar ranging sensors on each spacecraft, a sensor-fusion Kalman Filter is used. This is a modification of the standard Kalman filter described above, which uses multiple measurement update cycles to incorporate the shared data of the swarm to further refine the covariance ellipsoid for each spacecraft. Figure 3 shows an example of the sensor fusion process, where the covariance of the position between *Sat #1* and the *Client* spacecraft can be improved by fusing the data from all the swarm spacecraft, even taking into account the GPS position errors defining the locations of each swarm spacecraft with respect to *Sat #1*.

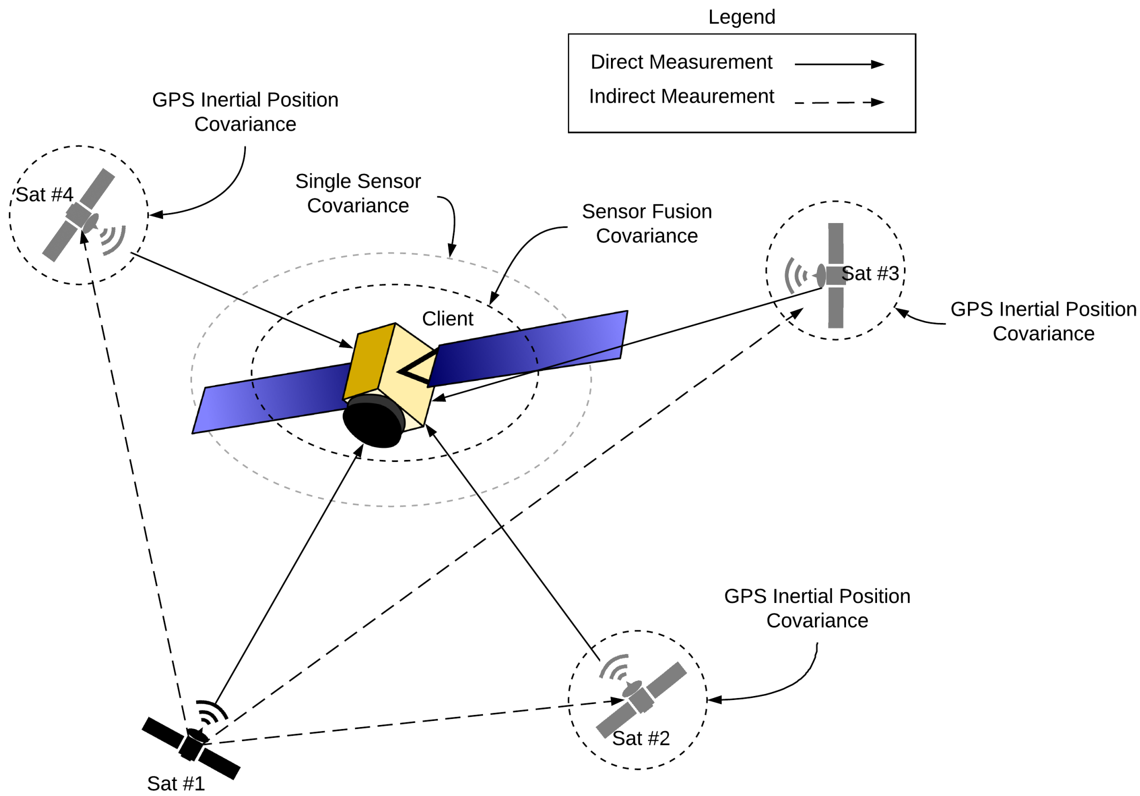

Fig. 3: Sensor Fusion Diagram

In order to take into account multiple sensor inputs, the unscented Kalman filter algorithm itself must be modified to be able to process this additional data. As described in Section 3.1 the Kalman filter operates by propagating an initial state in time using known equations of motion, and then a position and covariance update step is performed using data from an onboard sensor to filter out noise and erroneous data from the sensor data. With a sensor fusion Kalman filter, the propagation step remains the same (see Equation 1), however the correction and update step (see Equation 7) is performed multiple times – once for each sensor. This process is depicted in Figure 4 using pseudocode.

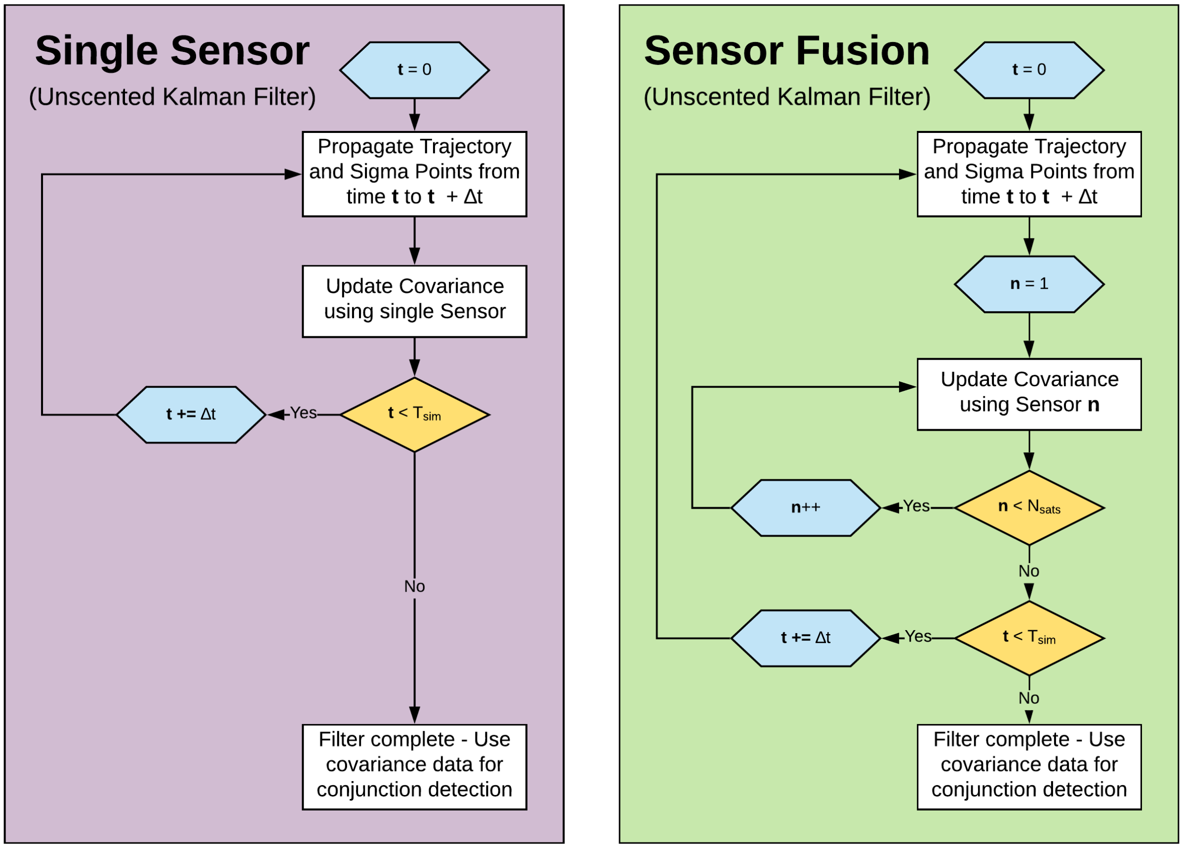

Fig. 4: Sensor Fusion Kalman Filter Process

Since the majority of the computation time in the Kalman filter is spent on the propagation step, rather than the update step, repeating the update step adds very little computational overhead, while greatly improving the spacecraft covariance.

4 Kalman Filtering for Real-Time Operations

In order to maintain safe trajectories that avoid collisions between spacecraft in the swarm, a Sensor Fusion unscented Kalman Filter (SFKF) is used to accurately determine, in real-time, the position and velocity of each spacecraft. This is done using only the information available to the swarm members themselves, through external sensors, and without additional information from operators on the ground. Using the Kalman filter, a set of covariance matrices are obtained for each predicted position and velocity, setting the upper limits of the error bars on the measured and processed data.

In this manner, trajectories can also be assigned a specific corridor, where if the spacecraft drifts too far from the designated trajectory (as determined by fusing sensor data and filtering it through the SFKF), a small correction maneuver is performed to re-align the spacecraft to its target.

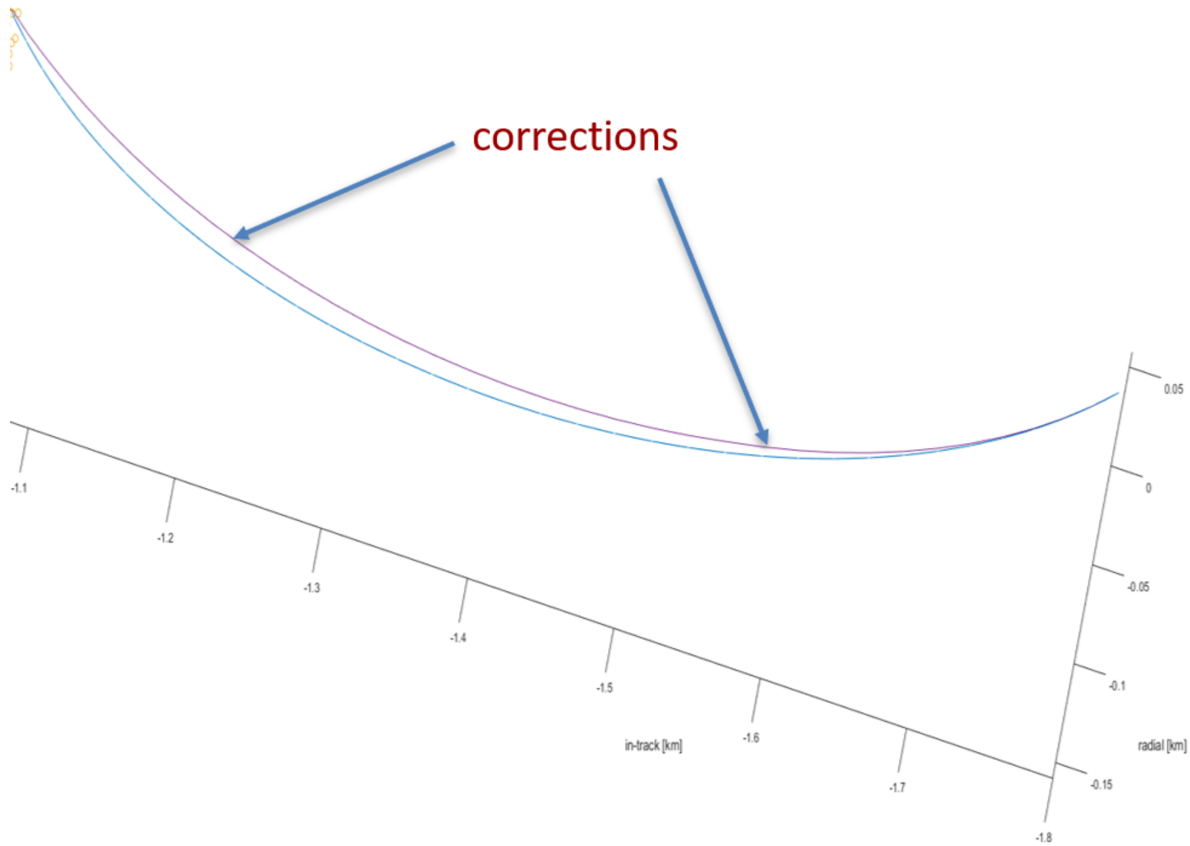

Fig. 5: Filtered Rendezvous Maneuver

Figure 5 shows a two-stage rendezvous process, where the first step of the trajectory is a non-intersecting free-flight trajectory. This means that if the burn to transition from the first to the second segment were to fail, there would be no collision. The total Δv for this maneuver is $4.1\,\mathrm{m/s}$, with $0.29\,\mathrm{m/s}$ of that due to small course corrections applied by the automated Kalman filter system to remain on-track. The blue trajectory is the projected course, and the purple trajectory is the path perceived by the spacecraft to be where it has actually travelled. This differs from the nominal trajectory due to injection errors, attitude knowledge errors on the spacecraft, and uncertainty of true position. Two corrective maneuvers are performed during this transfer, dictated by the Kalman filter, to maintain the desired destination. In this case there is no *truth* trajectory of where the spacecraft truly was, as there is no independent observer to determine this. Instead, the swarm Kalman filtering method uses sensors aboard all spacecraft to compute the relative position of each swarm member, and to determine if any sensors are aberrant.

5 Patched RPO using Kalman Filtering

In order to transition between various relative motion trajectories, a method of patched RPO has been devised. This method takes two defined states, separated by a defined period of time, to determine the most efficient set of impulsive maneuvers to be able to safely transition between these states. This is done using two, three, or four impulsive maneuvers, depending on what method yields the lowest Δv consumption, while maintaining safe operations with nearby spacecraft.

Each impulsive maneuver begins a new trajectory segment. To compute the shape of the transfer trajectory, and the initial velocity vector required to place the spacecraft on that trajectory, a two stage solver is used, using the solution of the linearized C-W equation to seed the nonlinear solver for the perturbed gravitational field solution.

$$\vec{v}_0 = \boldsymbol{\Phi}_{\boldsymbol{rv}}^{-1}\left(\vec{r}_f - \boldsymbol{\Phi}_{\boldsymbol{rr}}\vec{r}_0\right) \tag{14}$$

$$\vec{v}_f = \boldsymbol{\Phi}_{\boldsymbol{vr}}\vec{r}_0 + \boldsymbol{\Phi}_{\boldsymbol{vv}}\vec{v}_0 \tag{15}$$

Equations 14 & 15 describe the process used to solve for the initial and final velocities on a transfer arc, when the initial and final positions are known, as well as the transfer time. Φ_{rr}, Φ_{rv}, Φ_{vr}, and Φ_{vv} come from the definition of the C-W equations [18]. When solving with the C-W equations, the value for $\vec{v_0}$ is close to the desired solution, however it is a linearized approximation of the unperturbed solution. The perturbed solution can be solved iteratively, numerically solving the second order ODE in Equation 16.

$$\frac{d^2\vec{r}}{dt^2} = -\mu\frac{\vec{r}}{\|\vec{r}\|^3} + \vec{a}_{grav-pert} + \vec{a}_{sun-moon} + \vec{a}_{SRP} \tag{16}$$

Where $\vec{a}_{grav-pert}$ accounts for the perturbations in Earth's asymmetrical gravitational field, $\vec{a}_{sun-moon}$ accounts for the Sun and Moon gravitational perturbations, and $\vec{r}_{SRP}$ accounts for the solar radiation pressure. This system of second order ODEs is then solved iteratively to find the initial velocity $\vec{v}_{0_{pert}}$ such that the final position, $\vec{r}_f$, is the desired trajectory endpoint. This was done using the *fsolve* method in MATLAB, using the velocity $\vec{v_0}$ from the C-W linear solution as the initial solver guess to jumpstart the iterative process.

This is initially done prior to the start of the maneuver, in order to plan out the trajectory being travelled and prepare the spacecraft for the burn. Once the initial Δv is applied, the spacecraft has been injected onto the transfer trajectory. A Kalman filter is then used to compute the estimated true position of the spacecraft in real-time, using its onboard sensors. This estimated position is then used to recompute the target point at the end of the transfer arc, at a rate of once per second. If the trajectory is determined to deviate by more than 5 m, then Equation 16 is solved again iteratively, and a small impulsive maneuver is performed to align to the updated trajectory, maintaining the original target point as the destination. Figure 6 depicts this process graphically. This is done continuously in real-time during all swarm operations to maintain trajectories within their corridors.

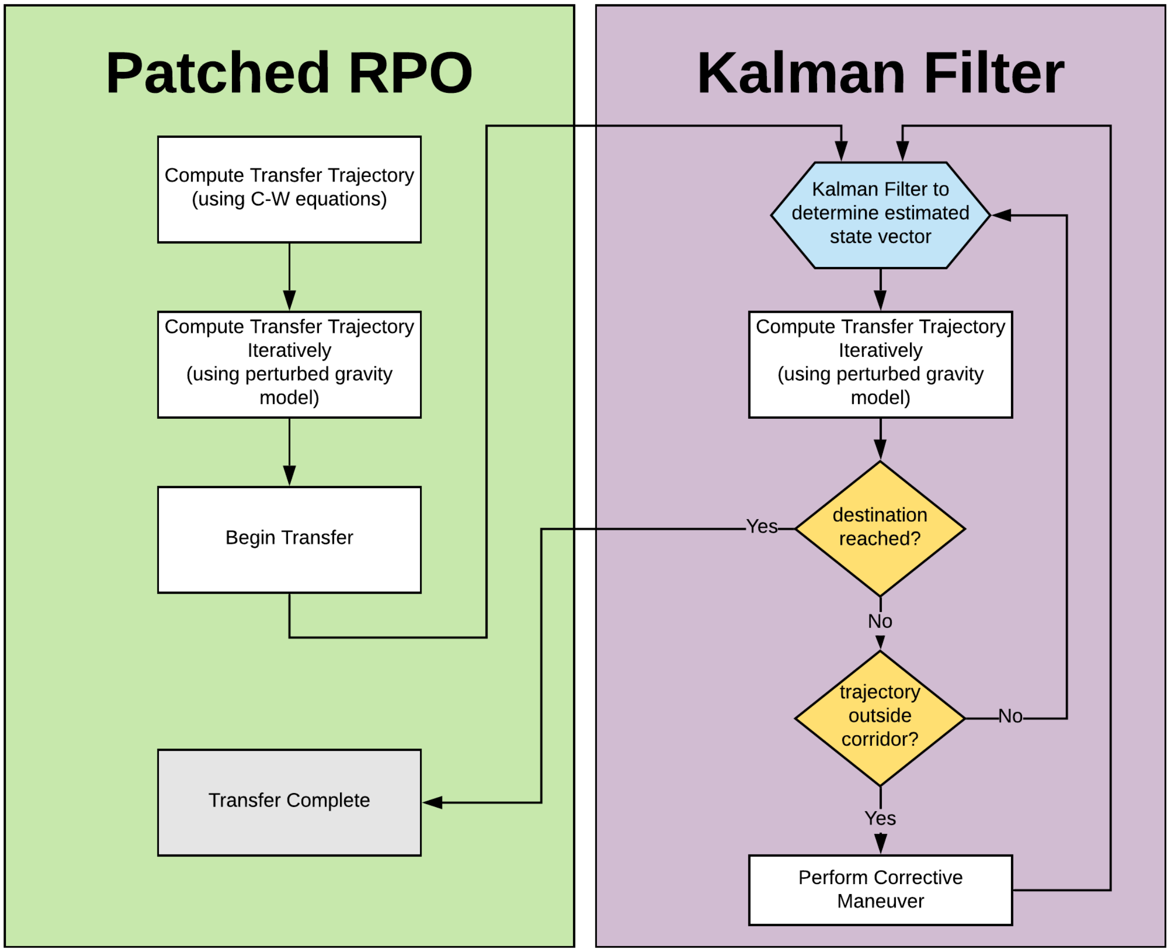

Fig. 6: Patched RPO Process

6 Results

Using the following assumptions, a set of simulations were run to determine the estimated position vs time for each spacecraft in the swarm, computed using the SFKF. This produced a set of covariance matrices for each spacecraft at each timestep.

Swarm spacecraft assumptions (conservative w.r.t COTS products):

1. On-board propulsion system and fuel source (preferably refuelable) with a minimum 200 m/s Δv capacity.
2. Three-axis attitude control, and attitude knowledge with 0.5° accuracy.
3. Relative motion position sensors to determine range to nearby spacecraft with 5% accuracy
4. Relative motion speed sensors to determine the speed of nearby spacecraft with 1% accuracy
5. Redundant communication systems to transmit and receive data between all spacecraft in the swarm.

Comparing the data from this simulation (see Fig. 7) with GPS position accuracy [19], it can be seen that the SFKF offers greater fidelity in close proximity, providing position estimates within 10 m, as compared to GPS in LEO which offers 25 m accuracy.

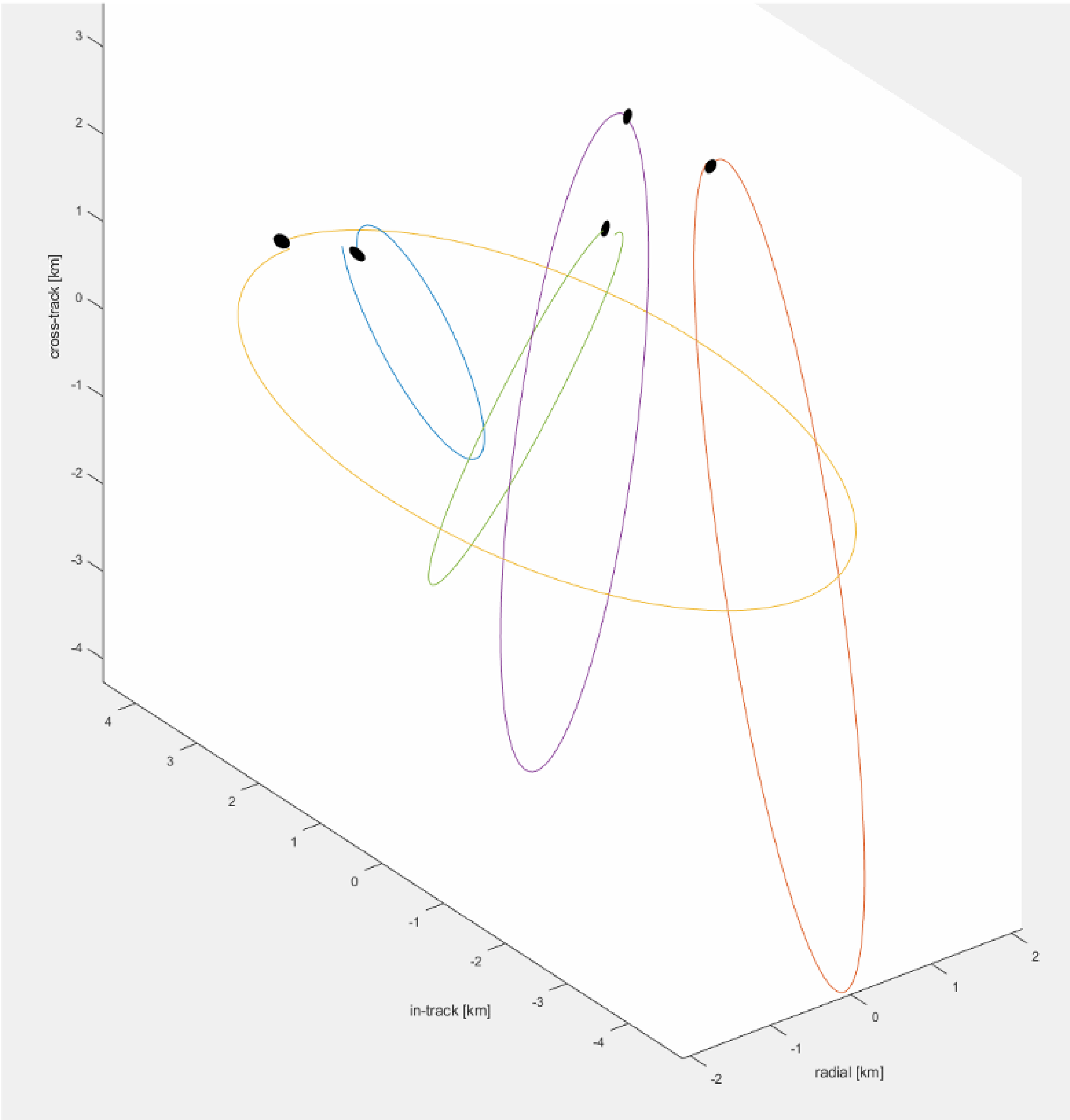

Fig. 7: Swarm Covariance Estimates from SFKF

7 Conclusions

When dealing with large swarms of spacecraft, there is an abundance of data available to the swarm as a whole. In order to make efficient use of this data for navigation and mission operations purposes, new sets of algorithms are needed to process this data in real-time and extract the relevant point for immediate use. The Sensor Fusion Kalman Filtering algorithm for spacecraft swarms presented in this paper is one method to achieve this for large swarms of spacecraft, as demonstrated through simulated models.

A common method of validation for real-time space operations with hardware-in-the-loop is the use of an Air Bearing Platform (ABP) for near-frictionless simulations in three to six degrees of freedom [20–23]. The authors would like to expand upon this research with future hardware demonstration on an ABP to demonstrate a closed-loop control system that implements the SFKF.

References

[1] Caleb Henry. Northrop Grumman's MEV-1 servicer docks with Intelsat satellite. *SpaceNews*, Feb 2020. `https://spacenews.com/northrop-grummans-mev-1-servicer-docks-with-intelsat-satellite/`.

[2] Debra Werner. Orbital ATK's giant leap into satellite servicing begins with baby steps. `https://spacenews.com/orbital-atks-giant-leap-into-satellite-servicing-begins-with-baby-steps/`, 2018. [Online; accessed 6-September-2018].

[3] Neil Dipprey and Scott Rotenberger. Orbital Express Propellant Resupply Servicing. In *39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, page 4898, 2003.

[4] Aman Chandra, Himangshu Kalita, Roberto Furfaro, and Jekan Thangavelautham. End to End Satellite Servicing and Space Debris Management. *arXiv preprint arXiv:1901.11121*, 2019.

[5] Benjamin B Reed, Robert C Smith, Bo J Naasz, Joseph F Pellegrino, and Charles E Bacon. The restore-L servicing mission. In *AIAA SPACE 2016*, page 5478. American Institute of Aeronautics and Astronautics, 2016.

[6] Jason L Forshaw, Guglielmo S Aglietti, Nimal Navarathinam, Haval Kadhem, Thierry Salmon, Aurélien Pisseloup, Eric Joffre, Thomas Chabot, Ingo Retat, Robert Axthelm, et al. Removedebris: An in-orbit active debris removal demonstration mission. *Acta Astronautica*, 127:448–463, 2016.

[7] Brook R Sullivan, Joseph Parrish, and Gordon Roesler. Upgrading In-service Spacecraft with On-orbit Attachable Capabilities. In *2018 AIAA SPACE and Astronautics Forum and Exposition*, page 5223, 2018.

[8] Rahul Rughani and David A Barnhart. Using Genetic Algorithms for Safe Swarm Trajectory Optimization. In *30th AIAA/AAS Space Flight Mechanics Meeting. Orlando Fl, USA*, 6-10 January 2020.

[9] Rahul Rughani and David A Barnhart. Safe Construction in Space: Using Swarms of Small Satellites for In-Space Manufacturing. *SmallSat Conference, Logan, Utah (Virtual)*, 2020.

[10] David A Barnhart and Rahul Rughani. On-Orbit Servicing Ontology Applied to Recommended Standards for Satellites in Earth Orbit. *Journal of Space Safety Engineering*, 2020.

[11] David E Gaylor and Brent William Barbee. Algorithms for Safe Spacecraft Proximity Operations. In *AAS/AIAA Spaceflight Mechanics Meeting*, 2007.

[12] Dario Izzo and Lorenzo Pettazzi. Autonomous and Distributed Motion Planning for Satellite Swarm. *Journal of Guidance, Control, and Dynamics*, 30(2):449–459, 2007.

[13] Ismael Lopez and Colin R McInnes. Autonomous Rendezvous Using Artificial Potential Function Guidance. *Journal of Guidance, Control, and Dynamics*, 18(2):237–241, 1995.

[14] GL Slater, SM Byram, and TW Williams. Collision Avoidance for Satellites in Formation Flight. *Journal of Guidance, Control, and Dynamics*, 29(5):1140–1146, 2006.

[15] I Ross, J King, and Fariba Fahroo. Designing Optimal Spacecraft Formations. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 4635, 2002.

[16] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons, 2004.

[17] Mike West and Jeff Harrison. *Bayesian Forecasting and Dynamic Models*. Springer Science & Business Media, 2006.

[18] Howard D. Curtis. *Orbital Mechanics for Engineering Students*. (pp.383-387) Elsevier Aerospace Engineering Series. Elsevier, 2014.

[19] Anthony S Williams. Expected Position Error for an Onboard Satellite GPS Receiver. Technical report, Graduate School of Air Force Institute of Technology Wright-Patterson AFB, Ohio, 2015.

[20] David Barnhart, Tim Barrett, Jeff Sachs, and Peter Will. Development and Operation of a Micro-Satellite Dynamic Test Facility for Distributed Flight Operations. In *AIAA SPACE 2009 Conference & Exposition*, page 6443, 2009.

[21] David A Barnhart, Ryan H Duong, Lizvette Villafana, Jaimin Patel, and Shreyash Annapureddy. The Development of Dynamic Guidance and Navigation Algorithms for Autonomous on-Orbit Multi-Satellite Aggregation. In *70th International Astronautical Congress (IAC), Washington, DC*, 2019.

[22] Yashwanth Kumar Nakka, Rebecca C Foust, Elena Sorina Lupu, David B Elliott, Irene S Crowell, Soon-Jo Chung, and Fred Y Hadaegh. A Six Degree-of-Freedom Spacecraft Dynamics Simulator for Formation Control Research. *18th AAS/AIAA Spaceflight Mechanics Meeting, Galveston TX*, 2018.

[23] Michael Smat, David Barnhart, Lizvette Villafaña, and Kirby Overman. Cellular Based Aggregated Satellite System: The Design and Architecture of a Three Degree of Freedom Near-Frictionless Testbed for Ground Validation of CubeSat Operations. *2020 SmallSat Conference, Logan, Utah (Virtual)*, 2020.