Correctness Analysis and Optimality Bounds of Multi-spacecraft Formation Initialization Algorithms

Mike Schuresko and Jorge Cortés*

May 16, 2007

Abstract

This paper considers formation initialization for a class of autonomous spacecraft operating in deep space with arbitrary initial positions and velocities. Formation initialization is the task of getting a group of autonomous agents to obtain the relative and/or global dynamic state information necessary to begin formation control. We associate a "worst-case total angle traversed" optimality notion with the execution of any formation initialization algorithm, and present performance bounds valid for any correct algorithm. We design the SPATIAL SPACECRAFT LOCALIZATION ALGORITHM, HALF TWIST ALGORITHM and the WAIT AND CHECK ALGORITHM, analyze their correctness properties and characterize their performance in terms of worst-case optimality and execution time.

Contents

*M. Schuresko and J. Cortés are with the Department of Applied Mathematics and Statistics, Baskin School of Engineering, University of California at Santa Cruz, CA 95064, USA {mds,jcortes}@soe.ucsc.edu

1 Introduction

Motivation and problem statement

Deploying large structures in space requires multiple spacecraft to coordinate their activities, due, in part, to the limited payload capabilities of launch vehicles. Key aspects of spacecraft coordination which are likely to be used in a broad variety of contexts include: (i) formation initialization, i.e., the establishment and maintenance of relative dynamic state information and/or on-board inter-spacecraft communication; (ii) formation acquisition, i.e., making the group of spacecraft attain a desired geometry; and (iii) formation regulation and tracking, i.e., maintaining fixed inter-spacecraft range, bearing, and inertial attitudes with high accuracy along the execution of a desired trajectory.

In this paper, we focus our attention on the formation initialization problem. This problem is especially important for spacecraft operating in deep space, where conventional Earth-based GPS does not provide sufficiently accurate position information. Here, we consider a spacecraft model motivated, in part, by the design possibilities of NASA's "Terrestrial Planet Finder" mission. Our spacecraft model is similar to the one proposed in [1]. The spacecraft have laser-based directional relative position sensors, like the kind described in [2], which require two sensors to lock on to each other before getting a position measurement. The spacecraft are assumed to be in deep space, far from the effects of gravitational curvature.

Literature review

A fairly extensive bibliography of missions which plan to use spacecraft formation flying can be found in [3]. These include Terrestrial Planet Finder [4], EO-1 [5], TechSat-21 [6] and Orion-Emerald [7]. A driving motivation behind formation flying research is that of large aperture adaptive optics in space, e.g. [4] and [8]. A good overview on current research on formation flying for optical missions is contained in [2]. Most of the work on control algorithm design has focused on formation acquisition and tracking. A survey of algorithms is given in [9]. Leader-following approaches, e.g. [10, 11], and virtual structures approaches, e.g. [12], prescribe the overall group behavior by

specifying the behavior of a single leading agent, either real or virtual. Motion planning and optimal control problems are analyzed in [13]. The only work known to us that has dealt in detail with formation initialization is [1].

Statement of contributions

The contributions of this paper are twofold. On the one hand, we provide optimality bounds on the performance of any correct formation initialization algorithm. Our analysis consists of a systematic study of optimality of algorithms, both in two and three dimensions, with regard to worst-case total angle rotated by any member of the group of spacecraft. As a byproduct of our analysis, we provide justification for the **Opposing Sensor Constraint** in [1] by showing that optimal algorithms exist which invoke it. Our optimality bounds give rise to necessary conditions valid for any correct algorithm, that we use to show that the rotation phases of the algorithm presented in [1] fail to achieve formation initialization.

On the other hand, we present three original algorithms. The SPATIAL SPACECRAFT LOCALIZATION ALGORITHM achieves formation initialization through a simple sequence of rotational maneuvers, each of which sweeps a region of a particular partition of space. The WAIT AND CHECK ALGORITHM validates our lower bound on total angle by sacrificing time efficiency for better angle efficiency, while the HALF TWIST ALGORITHM achieves a total angle between that of SPATIAL SPACECRAFT LOCALIZATION ALGORITHM and WAIT AND CHECK ALGORITHM without additional wait times. For these algorithms, we assess their correctness, and formally characterize their performance with regards to the optimality measures mentioned above.

Organization

This tech report is primarily intended as a companion to our conference paper [15] (in which the proofs of various theorems were omitted). Section 2 presents a set of definitions which will be used throughout the paper. In Section 3 we give necessary conditions for the correctness of any candidate solution to this problem, from which we derive lower bounds for the optimality of any working solution. Section 4 presents provable formation initialization algorithms, including a simple algorithm for 3-D in Section 4.2 a slightly more optimal version in Section 4.3 which was not mentioned in [15] for space reasons and a nearly-optimal algorithm which achieves angle efficiency at the expense of long wait times in Section 4.4.

2 Preliminaries

Each spacecraft consists of a rigid body containing instruments on one side, which need to be shielded from the sun (see Fig. 1). To serve this purpose, a *sun shield* is mounted to the spacecraft body on the side opposite the instruments. The *sun shield normal vector*, $\vec{n}_{\text{SUN}}(S)$, indicates the direction of the sun shield of spacecraft S. We make the approximation that the vector to the sun, $\vec{v}_{\text{SUN}}$, is the same for each feasible spacecraft position. In order to operate without damaging the instrumentation, each spacecraft must maintain the constraint $\vec{n}_{\text{SUN}}(S) \cdot \vec{v}_{\text{SUN}} \geq \cos(\Theta_{\text{sun}})$ for some pre-specified Θ_{sun}

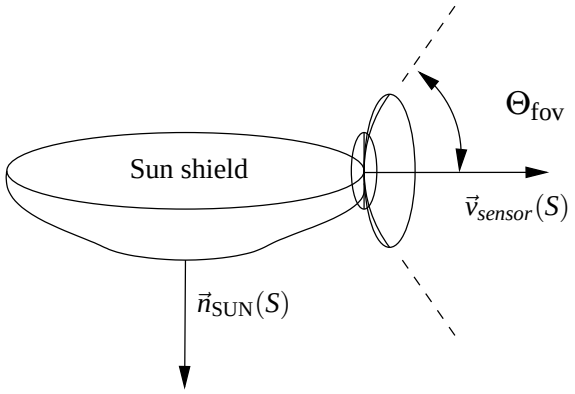

Figure 1: Configuration of spacecraft geometry, and body frame definition.

at all times. Relative position and velocity measurements between two spacecraft are made through the *metrology sensors* of the two craft. The metrology sensor of spacecraft S senses within a conical region (C_S) with a half angle of Θ_{fov} (assumed here to be greater then $\frac{\pi}{4}$ unless otherwise stated) about the *sensor cone centerline*. Accurate orientation information is available for all spacecraft through measurements of reference stars. The spacecraft are placed such that the curvature of earth's gravitational field has a negligible effect (for instance a Lagrange point). We therefore assume that if no spacecraft undergo translational acceleration, then the spacecraft move with constant (initially unknown) velocity in straight lines relative to each other.

Definition 2.1. *The* global frame of reference *is an arbitrary orthonormal frame, $GF = \{X_g, Y_g, Z_g\}$, where $X_g = \vec{v}_{SUN}$. For a spacecraft S, P_S denotes the position of the center of mass of S in the frame GF. The* center of mass frame *of S, $CMF(S)$, corresponds to translating the global frame GF to P_S. The* body frame *of S, $BF(S) = \{\hat{X}_S, \hat{Y}_S, \hat{Z}_S\}$ is defined by $\hat{X}_S = \vec{n}_{SUN}(S)$, $\hat{Z}_S = \vec{v}_{sen}(S)$ and $\hat{Y}_S = \hat{Z}_S \times \hat{X}_S$. In this frame, $\{0,0,0\}$ is at the center of mass of S.*

The state of each spacecraft $S \in \{S_1, \cdots, S_n\}$ can be described by $(P_S, M_S) \in \mathbb{R}^3 \times SO(3)$.$M_S$ transforms $BF(S)$ onto $CMF(S)$ and P_S defines the translation between $CMF(S)$ and GF. The spacecraft are fully actuated.

The sensor cone $C_S : \mathbb{R}^3 \times SO(3) \to 2^{\mathbb{R}^3}$ of S is

$$C_S(P_S, M_S) = \{\vec{x} \in \mathbb{R}^3 \ : \ \frac{[0,0,1]^T M_S(\vec{x} - P_S)}{\|\vec{x} - P_S\|} \leq \cos(\Theta_{\text{fov}})\}. \tag{1}$$

When it is clear from the context, we will use the simpler notation C_S. In order to get a relative position reading between two spacecraft, S_1 and S_2, $P_{S_1} \in C_{S_2}$ and $P_{S_2} \in C_{S_1}$ must hold. This condition is called *sensor lock*.

The algorithms we present all require the n spacecraft performing the algorithm to be split into two disjoint groups, G_1 and G_2 such that $G_1 \cup G_2 = \{S_1, ..., S_n\}$. This can either be done a priori before launch or (preferably) with a distributed algorithm prior to running formation initialization (see [16]).

Two spacecraft, S_1 and S_2, are said to be maintaining the **Opposing Sensor Constraint** if $\vec{v}_{\text{sen}}(S_1) = -\vec{v}_{\text{sen}}(S_2)$.While this constraint is not strictly necessary for a correct solution to the formation initialization problem, we will show in Section 3 that it is a convenient constraint to work with. Note that this does not fully constrain the relative orientation of S_1 with respect to S_2. When specifying an algorithm requiring the **Opposing Sensor Constraint**, we will often specify the more restrictive constraint

$$M_{S_1}^{-1} M_{S_2} = M_{\text{opp}} = \text{diag}(1,-1,-1).$$

We call this constraint the **Opposing Frame Constraint**. In addition to maintaining the **Opposing Sensor Constraint**, the **Opposing Frame Constraint** also guarantees that if spacecraft S_1 verifies the sun-angle constraint, then S_2 also verifies it.

2.1 Algorithm definition

In this section we formally define what we mean by an "algorithm." In what follows, D_{msg} denotes the set of possible messages a spacecraft can communicate at any instant, and $D_{\text{sensor}} = (\mathbf{Z}_2 \times \mathbb{R}^3 \times \mathbb{R}^3)^n$ the set of possible sensor cone readings for a spacecraft.

Definition 2.2 (Algorithm notion)**.** *An algorithm is a tuple $A_S = (St_{S,0}, F_S, G_S, \delta t_{step})$, where $St_{S,0} \in D_{St}$, the initial internal state of S, contains no information about the location of the other spacecraft and F_S is a map of the form*

$$\begin{aligned} F_S : \mathbb{R} \times SO(3) \times D_{St} &\to \mathbb{R}^3 \\ (t, M_S, St_S) &\mapsto \omega_S, \end{aligned}$$

and G_S is a map of the form

$$\begin{aligned} G_S : \mathbb{R} \times SO(3) \times D_{St} \times D_{msg}^{n-1} \times D_{sensor} &\to D_{St} \times D_{msg} \\ (t, M_S, St_S, MSG_{S,in}, SENSOR_S) &\mapsto (St_S, MSG_{S,out}). \end{aligned}$$

Definition 2.3. (Execution of an algorithm): *An execution by a spacecraft S of an algorithm $A_S = (St_S, F_S, G_S, \delta t_{step})$ during the time interval $[t_0, t_f]$ is given by $t \in [t_0, t_f] \to (P_S(t), M_S(t)) \in \mathbb{R}^3 \times SO(3)$ and $St_S : [t_0, t_f] \to D_{St}$ defined as follows:*

- *$\dot{P}_S(t) = V_S$, for some constant $V_S \in \mathbb{R}^3$;*
- *$\dot{M}_S(t) = \widehat{F}_S(t, St_S(t)) M_S(t)$, $t \in [t_0, t_f]$, where $\hat{\omega}$ is the matrix operator for the cross product with $\omega \in \mathbb{R}^3$;*
- *St_S is the piecewise constant function defined by*

$$St_S(t_{i+1}) = G_S((t_i, M_S(t_i), St_S(t_i), MSG_{S,in}(t_i), SENSOR_S)(t_i)),$$

for $i = 0, \ldots, m-1$, with $t_0, t_1, \ldots, t_m \in [t_0, t_f]$ a finite increasing sequence, where $t_i = k\,\delta t_{step}$ for some $k \in \mathbb{N}$ or t_i corresponds to the time instant when a change occurs in the value of the sensor cone readings. The initial value of $St_S(t_0)$ is $St_{S,0}$.

The lack of concrete specification of D_{msg} and D_{St} reflects our intent to provide lower bounds on algorithmic performance for spacecraft with a wide range of computational and communication capabilities. In practice, the working algorithms we present in Section 4 require basic computational capabilities on the part of each spacecraft.

2.2 Total angle traversed and solid angle covered

Here, we present the notions of total angle traversed and solid angle covered during the execution of an algorithm.

2.2.1 Total angle traversed during algorithm execution

In 3-D, recall that $M_S = [m_x, m_y, m_z]$ is an orthonormal basis matrix representing the orientation of spacecraft S. From Equation 8.6.5 of [17], we have $\hat{\omega} = \dot{M}_S M_S^{-1}$. , where $\hat{\omega}$ is the matrix operator for "cross product with ω."

The total angle traversed during the execution of an algorithm in 3-D is therefore

$$\int_{t_0}^{t_f} \sqrt{\hat{\omega}_{1,2}^2 + \hat{\omega}_{1,3}^2 + \hat{\omega}_{2,3}^2}\, dt.$$

One can think of the 2-D problem as the 3-D problem with rotations confined to the $\{Y, Z\}$ plane. Under this constraint, the previous expression reduces to

$$\int_{t_0}^{t_f} |\hat{\omega}_{2,3}|\, dt.$$

2.2.2 Solid angle traversed during algorithm execution

It will be useful to compute the total solid angle covered by the sensor cone C_S of a spacecraft performing a formation initialization algorithm in 3-D. If a spacecraft, S, with sensor cone field of view Θ_{fov} rotates by an angle of π about an axis l where $l \cdot \vec{v}_{\text{sen}}(S) = \cos(\Theta)$, $\Theta > \Theta_{\text{fov}}$, the new solid angle covered in this sweepcan be found by tracing a band about the unit sphere and calculating its area. See Figure 2 for an illustration.

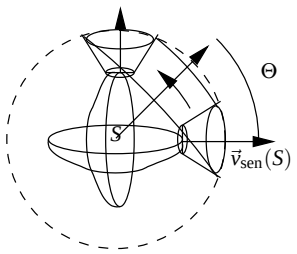

Figure 2: Method to compute rate of change of solid angle swept.

Recall that the solid angle of a cap of half angle α is $\int_0^{\alpha} 2\pi \sin(t)\, dt$. The area of this band can be found by subtracting caps of half angles $\Theta - \Theta_{\text{fov}}$ and $\pi - \Theta - \Theta_{\text{fov}}$

from the unit sphere and dividing by 2. , giving a result of

$$\frac{4\pi - 2\pi(1-\cos(\pi-\Theta-\Theta_{\text{fov}})) - 2\pi(1-\cos(\Theta-\Theta_{\text{fov}}))}{2}.$$

Dividing by π gives a rate of change of coverage of solid angle for this operation , when performed at angular velocity ω, as $2\|\omega\|\sin(\Theta)\sin(\Theta_{\text{fov}})$. A similar argument gives the expression for when $\Theta \leq \Theta_{\text{fov}}$ as $\|\omega\|(1+\sin(\Theta)\sin(\Theta_{\text{fov}})-\cos(\Theta)\cos(\Theta_{\text{fov}}))$

The total solid angle covered by a spacecraft S executing an algorithm, A, between times t_0 and t is then

$$F_{\text{sld}}(t) = \int_{t_0}^{t} f_{\text{sld}}(\omega(\tau))d\tau,$$

where $f_{\text{sld}}(\omega) : \mathbb{R}^3 \to \mathbb{R}$ is defined by $f_{\text{sld}}(\omega) = 2\|\omega \times \vec{v}_{\text{sen}}(S)\sin(\Theta_{\text{fov}})\|$ for $\arccos(\omega \cdot \vec{v}_{\text{sen}}(S)/\|\omega\|) > \Theta_{\text{fov}}$, and $f_{\text{sld}}(\omega) = \|\omega \times \vec{v}_{\text{sen}}(S)\sin(\Theta_{\text{fov}})\| + \|\omega\| - |\omega \cdot \vec{v}_{\text{sen}}(S)|$ otherwise. For us, the total solid angle covered by S during the course of the algorithm to be $F_{\text{sld}}(t_f) + \alpha_0$ where t_f is the earliest time at which formation initialization is guaranteed to be complete and $\alpha_0 = 2\pi(1-\cos(\Theta_{\text{fov}})$ is the solid angle contained in $C_S(t_0)$ at time t_0.

Remark 2.4. *Note that* $0 \leq f_{sld}(\omega) \leq 2\|\omega\|\sin(\Theta_{fov})$. •

Analogously, the total angle covered by a spacecraft S performing an algorithm A in 2-D between times t_0 and t is

$$F_{\text{angle}}(t) = \int_{t_0}^{t} |\omega| dt.$$

2.3 Formation initialization problem

Formation initialization solutions entail establishing communication and/or relative position information. Here we restrict ourselves to the establishment of relative position and velocity information between each pair of spacecraft. We assume that this information can come from any combination of direct sensor readings, odometry and communication with other spacecraft.

Definition 2.5. *Let* $[t_s, t_f]$ *be the duration of time during which a formation initialization algorithm runs. Define* $G(t)$ *to be the* ***relative position connectivity network*** *at time* t*, defined by* $G(T) = (V,E)$ *where* $v(S_i) \in V$ *correspond to the spacecraft* S_i*, and the edge* $(v(S_i), v(S_j))$ *is in* E *if and only if spacecraft* S_i *and* S_j *are in a state of sensor lock. A solution to the formation initialization problem is one that guarantees that the graph* $\cup_{t\in[t_s,t_f]} G(t)$ *is connected.*

3 Correctness and optimality of formation initialization algorithms

We start this section by providing a necessary condition for the correctness of any formation initialization algorithm. Then, we proceed to use this condition as the basis

for a series of optimality bounds. We also present optimality results which justify the **Opposing Sensor Constraint** and allow us to more easily reason about the n spacecraft case (where $n > 2$).

Theorem 3.1. *Let S be executing a correct formation initialization algorithm in d dimensions, with $d \in \{2,3\}$. For every $v \in \mathbb{R}^d$, let t_v be the first time such that $v \in C_S(t_v) = C_S(P_S(t_v), M_S(t_v))$. Then, there must exist $t^* > t_v$ such that $-v \in C_S(t^*)$.*

Proof. For simplicity, let $\text{vers}(u) = u/\|u\|$, for $u \in \mathbb{R}^d$. Consider two spacecraft, S_1 and S_2. S_2 travels in the plane defined by it's velocity (V_{S_2}), and $p_{\text{cl}}(S_1, S_2)$, where $p_{\text{cl}}(S_1, S_2)$ is the point of closest approach between S_1 and S_2 in $CMF(S_1)$. At time t S_2 makes an angle with $p_{\text{cl}}(S_1, S_2)$ of $\arctan(\frac{\|V_{S_2}\|}{\|p_{\text{cl}}(S_1,S_2)\|}t + t_0)$ for some t_0. S_2's initial conditions can be chosen to match any arbitrary V_{S_2}, $p_{\text{cl}}(S_1, S_2)$ and t_0. Because of this, given an ε and times, t_1 and t_2, $\text{vers}(P_{S_2})$ can be made to stay within an angle of ε of $-\text{vers}(V_{S_2})$ until time t_1, and move to within an angle of ε of $\text{vers}(V_{S_2})$ by t_2. Let t_1 be the first time at which the minimum angle between any ray in $C_{S_1}(t_1)$ and $\text{vers}(-V_{S_2})$ is less then or equal to ε and t_2 be the first time at which $C_{S_1}(t_2)$ includes $\text{vers}(-V_{S_2})$. In order to ensure S_1 finds S_2, $C_{S_1}(t^*)$ must include $\text{vers}(V_{S_2})$ at some time $t^* > t_1$. Since ε was picked arbitrarily and the sensor cone is always closed, $C_{S_1}(t^*)$ must include $\text{vers}(V_{S_2})$ at some time $t^* > t_2$. □

Name:	Formation Initialization Algorithm
Assumes:	Spacecraft model in Section 2.

1: **if** $S_i \in G_1$ **then**
2: Rotate to align M_{S_i} with I_3
3: **else**
4: Rotate to align M_{S_i} with M_{opp}
5: **end if**
6: Wait for common start time t_s
7: Rotate by 3π about X_{S_i}.
8: Rotate $-\Theta_{\text{tilt}}$ (in this case 25 degrees) about Y_{S_i}.
9: Rotate $2\Theta_{\text{tilt}}$ about Y_{S_i}.
10: Rotate by π about X_{S_i}.
11: Rotate $2\Theta_{\text{tilt}}$ about Y_{S_i}.
{This is the end of the rotational component of the algorithm}
12: Rotate $-\Theta_{\text{tilt}}$ about Y_{S_i}.
13: Wait for some time $t_{\text{near field}} > 0$
14: **if** $S_i \in G_1$ **then**
15: Begin translating along Z_{S_i} with speed v_{max}, where v_{max} is the maximum relative velocity between any two craft.
16: **end if**

Table 1: Formation Initialization algorithm proposed in [1].

Theorem 3.2. *The algorithm stages described in Steps 1-12 of Table 1 are not, by themselves, sufficient to solve the formation initialization problem.*

Proof. Let $S \in G_1$ perform this algorithm. By Theorem 3.1, for any vector v, $C_S(t)$ must contain $-v$ at least once before the last time $C_S(t)$ contains v. But each $v \in R_{down}$ is last in $C_S(t)$ during Step 9, and no $v \in \{u \in CMF(S) : -u \in R_{down}\}$ is in $C_S(t)$ before Step 10. Thus $R_{down}(S)$ does not satisfy this condition. □

For our purposes, we will consider the algorithm which minimizes the maximum worst-case total angle traversed of any spacecraft S_i to be the optimal algorithm. Other reasonable options would include the algorithm which minimizes the worst-case sum over all spacecraft S_i of the total angle traversed.

Let is now justify that **Opposing Sensor Constraint** is optimal.

Theorem 3.3. *(Justification of the **Opposing Sensor Constraint**): Let S_1 and S_2 be two spacecraft. The most optimal algorithm to guarantee that S_1 and S_2 attain sensor lock is one which uses the **Opposing Sensor Constraint**.*

Proof. Imagine there is some algorithm A which achieves sensor lock between S_1 and S_2 in time t_{lock}. Create a new algorithm A^* in which S_1 implements A, but S_2 maintains the **Opposing Sensor Constraint** with S_1. If S_2 had been following A, the apex of $C_{S_2}(t_{lock})$ would be in $C_{S_1}(t_{lock})$ at time t_{lock}. Since S_1 is following A in algorithm A^*, the apex of $C_{S_2}(t_{lock})$ is in $C_{S_1}(t_{lock})$ when both craft follow A^*. By symmetry properties of the **Opposing Sensor Constraint**, the apex of $C_{S_1}(t_{lock})$ is in $C_{S_2}(t_{lock})$, thus guaranteeing sensor lock at or before time t_{lock}. This means that for any algorithm, A, which guarantees sensor lock, a modified algorithm (A^*) which maintains the **Opposing Sensor Constraint** can be constructed such that A^* guarantees sensor lock in at most as much worst-case rotation as A. □

The next result shows an equivalence between worst-case bounds for 2 spacecraft and worst-case bounds for any number $n > 2$ of spacecraft.

Theorem 3.4 (Extending worst-cases to n Spacecraft)**.** *Given a spacecraft S_n with sensor cone half-angle Θ_{fov}, and any $\varepsilon > 0$, the worst-case total angle traversed by S_n while performing a correct algorithm with $n-1$ other spacecraft is identical to the worst-case total angle traversed by a spacecraft with sensor cone half-angle $\Theta_{fov} + \varepsilon$ performing a correct algorithm with one other spacecraft.*

Proof. Let t_{worst} be the worst-case time for 2 spacecraft to find each other given a maximum angular velocity of ω_{max}. Clearly the worst-case time for n craft is no worse then this. Pick the initial conditions of the first $n-1$ spacecraft arbitrarily. Let C be the set of communications the first $n-1$ craft would send if they start from these conditions and fail to achieve sensor lock with S_n by time t_{worst}. Let T be the trajectory S_n would take given communications C. Let A_t be the algorithm for two spacecraft, S_1 and S_2, under which each S_1 blindly follows T and S_2 maintains the opposing sensor constraint with respect to S_1. Let P_{worst} and v_{worst} be the initial position and velocity of S_1 with respect to S_2 that achieves the worst-case total angle traversed for S_1 under A_t. In the n spacecraft case, pick some spacecraft S_i. Set the initial position and velocity of S_n with respect to S_i to be λP_{worst} and λv_{worst} for λ such that $\min_{t\in[0,t_{\text{worst}}]}(\|P_{\text{worst}} + v_{\text{worst}}t\|)\lambda > \frac{r_{\text{worst}}}{\sin(\varepsilon)}$. Since $S_1, \cdots, S_{n-1}$ never get more then r_{worst} apart, these spacecraft are contained within a ball of radius r_{worst} centered at S_i. By construction of λ, these

craft stay within an angular ball of ε from S_n's point of view, and thus none of these craft achieve sensor lock with S_n before time t_{worst}. □

Theorem 3.4 allows the result from Theorem 3.3 to be generalized to any number of spacecraft. In addition, we will use Theorem 3.4 throughout the remainder of the paper to allow us to analyze worst-case total angle bounds by considering the 2 spacecraft case.

Next, we provide gives a lower bound of the total angle covered for the 2-D problem

Theorem 3.5 (2-D lower bounds on angle traversed)**.** *For any algorithm A which solves the 2-D formation initialization problem, and $\Theta_{fov} < \frac{\pi}{2}$, the worst-case total angle covered by S_1 performing A is 3π.*

Proof. For $\Theta_{\text{fov}} < \frac{\pi}{2}$, by Theorem 3.1, every vector, v, on the 2-sphere must be scanned at least once before the final scan of $-v$. This means S_1 must scan at least half the directions on the unit 2-sphere twice for a total angle covered of 3π. □

From Theorem 3.5 we can deduce that the worst-case minimum total angle traversed by any correct formation initialization algorithm in 2-D is $min(3\pi - 2\Theta_{\text{fov}}, 4\pi - 4\Theta_{\text{fov}})$.

The next result gives a lower bound on the solid angle covered by any algorithm solving the 3-D problem.

Theorem 3.6 (3-D lower bounds on solid angle covered)**.** *For any algorithm A which solves the 3-D formation initialization problem, and $\Theta_{fov} < \frac{\pi}{2}$, the worst-case total solid angle covered by S_1 performing A is 6π.*

Proof. The total solid angle of a sphere is 4π. For $\Theta_{\text{fov}} < \frac{\pi}{2}$, by Theorem 3.1, every vector, v, on the 3-sphere must be scanned at least once before the final scan of $-v$. This means S_1 must scan at least half the directions on the unit 3-sphere twice for a total solid angle covered of 6π. □

The bound in this result induces a lower bound on angle traversed in 3-D.

Corollary 3.7 (3-D lower bounds on total angle)**.** *For any algorithm A which solves the 3-D formation initialization problem, and $\Theta_{fov} < \frac{\pi}{2}$, the worst-case total angle traversed by S_1 performing A is at least $\frac{3\pi - \alpha_0/2}{\sin\Theta_{fov}}$ where $\alpha_0 = 2\pi(1 - \cos(\Theta_{fov}))$.*

Proof. Recall from Remark 2.4 that $\frac{d}{dt}F_{\text{sld}}(t) = f_{\text{sld}}(\omega) \leq 2\|\omega\sin(\Theta_{\text{fov}})\|$. Since $6\pi - \alpha_0 \leq \int f_{\text{sld}}(\omega(t))dt \leq \int 2\|\omega\sin(\Theta_{\text{fov}})\|dt = 2\sin(\Theta_{\text{fov}})\int\|\omega\|dt$ and the total angle rotated is defined as $\int\|\omega\|dt$, we can say that the total angle rotated by any spacecraft S_1 performing A is $\frac{6\pi - \alpha_0}{2\sin(\Theta_{\text{fov}})}$. □

4 Provably correct formation initialization algorithms

Having given lower bounds on what is necessary for a correct formation initialization solution, in this section we set out to answer whether the problem as we pose it has a solution. Section 4.1 describes an algorithm from the literature for a 2-D variant of

this problem. Section 4.2 presents a purely rotational algorithm for formation initialization in 3-D and Theorem 4.5 gives a proof of its correctness. Section 4.4 provides an algorithm which comes closer to the optimality bounds presented in Section 3 at the expense of other practical considerations. This algorithm is presented as a demonstration of the tightness of the optimality bounds.

4.1 Formation initialization in two dimensions

To prove the correctness of the algorithm in 3-D, we will need a simpler algorithm for the 2-D case, which we term "in-plane search". This algorithm, described in Table 2, solves the formation initialization problem for a group of spacecraft residing in a plane, see [1]

Name:	PLANAR SPACECRAFT LOCALIZATION ALGORITHM
Goal:	Solve the Formation Initialization problem in 2-D
Assumes:	Spacecraft model in Section 2

1: **if** $S_i \in G_1$ **then**
2: Turn to common reference orientation Θ_{start}
3: **else**
4: Turn to $\Theta_{\text{start}} + \pi$
5: **end if**
6: At synchronized start time t_s, begin rotating with constant angular velocity $\omega > 0$. Continue this rotation for 3π radians.

Table 2: The PLANAR SPACECRAFT LOCALIZATION ALGORITHM.

Proposition 4.1 ([1])**.** *With the spacecraft model in Section 2, the* PLANAR SPACECRAFT LOCALIZATION ALGORITHM *achieves formation initialization.*

Note that the PLANAR SPACECRAFT LOCALIZATION ALGORITHM achieves the lower bound from Theorem 3.5.

4.2 SPATIAL SPACECRAFT LOCALIZATION ALGORITHM

Both the description of the full 3-D algorithm and its proof of correctness require some additional specific definitions, that we briefly expose next.

For the purpose of this algorithm, we will define $\Theta_{\text{tilt}} = \min\{\Theta_{\text{sun}}, \Theta_{\text{fov}}\}$ and assume $\Theta_{\text{fov}} \geq \frac{\pi}{4}$.

Definition 4.2. *Let S be a spacecraft. Define*

- $R_1(S) = \{\vec{u} \in CMF(S) \,:\, \vec{u} \cdot X_S \leq 0\}$;
- $R_2(S) = CMF(S) \setminus R_1(S)$.

Remark 4.3. *Let Θ_{tilt} be an angle such that $\frac{\pi}{2} - \Theta_{fov} < \Theta_{tilt} < \Theta_{fov}$. $R_1(S)$ is chosen so as to be included within the region swept out by spacecraft S's sensor cone while it is tilted by an angle Θ_{tilt} towards the sun axis and performing a 3π rotation about the sun axis. $R_2(S)$ is chosen so as to be included within the region swept out by spacecraft S's sensor cone while it is tilted $\frac{\pi}{2} - \Theta_{fov} < \Theta_{tilt} < \Theta_{fov}$ away from the sun axis and performing a 3π rotation about the sun axis. Also, note that in the frame $CMF(S)$, $R_1(S) \cup R_2(S) = \mathbb{R}^3$.* •

The full 3-D algorithm will invoke the subroutine described in Table 3.

Name:	3-D REGION SWEEP ALGORITHM
Goal:	Scan a region for use as a subroutine by SPATIAL SPACECRAFT LOCALIZATION ALGORITHM
Inputs:	(i) A spacecraft, S_i (ii) An integer, $n \in \{1,2\}$, indicating the region to be swept
Assumes:	(i) Spacecraft model in Section 2. (ii) $\Theta_{\text{fov}} \geq \frac{\pi}{4}$ and $\Theta_{\text{fov}} + \Theta_{\text{sun}} \geq \frac{\pi}{2}$.

Require: At the start of this subroutine, there exist matrices $M_1, M_2 \in SO(3)$ such that for all $S_i \in G_1$, $M_{S_i} = M_1$, for all $S_j \in G_2$, $M_{S_j} = M_2$, $M_1[1,0,0]^T = M_2[1,0,0]^T$ and $M_1[0,0,1]^T = -M_2[0,0,1]^T$.

Require: At the start of this subroutine, $[0,0,1]M_1[0,1,0]^T = 0$.

1: Set $\Theta_{\text{ROT}} = [0,0,1]M_S[0,0,1]^T(-1^n) \cdot \Theta_{\text{tilt}}$
2: Rotate by Θ_{ROT} about Y_{S_i}
3: Begin rotating about X_{S_i} by a constant angular velocity ω. Continue this rotation for 3π radians and then stop.
4: Rotate by Θ_{ROT} about Y_{S_i}

Table 3: The 3-D REGION SWEEP ALGORITHM.

At the end of the execution of the 3-D REGION SWEEP ALGORITHM, if S_i is in G_1, then $R_n(S_i)$ has been swept, otherwise S_i has maintained an orientation such that for all S_j in G_1 $M_{S_i}[0,0,1]^T = -M_{S_j}[0,0,1]^T$. With these ingredients, we can now define the SPATIAL SPACECRAFT LOCALIZATION ALGORITHM in Table 4.

Let us discuss the correctness of the SPATIAL SPACECRAFT LOCALIZATION ALGORITHM. As in Section 4.1, we reduce the problem to that of two spacecraft finding each other. Call these spacecraft $S_1 \in G_1$ and $S_2 \in G_2$. Recall that S_2's motion in $CMF(S_1)$ is along a straight line with constant velocity. Consider then the two half-spaces defined by the $\{Y,Z\}$ plane in $CMF(S_1)$. Because S_2 moves with constant velocity with respect to S_1, it can cross from one half-space to the other at most once. The paths it can take are as follows. S_2 can begin in $R_1(S_1)$ and cross to $R_2(S_1)$ at most once. Likewise S_2 can begin in $R_2(S_1)$ and cross into $R_1(S_1)$ at most once.

Because we make no assumptions about the speed at which these spacecraft take these paths, or at which part of the path they start, handling these cases will automatically handle the cases for paths that fail to cross the $\{Y,Z\}$ plane.

Name:	SPATIAL SPACECRAFT LOCALIZATION ALGORITHM
Goal:	Solve the Formation Initialization problem in 3-D
Assumes:	(i) Spacecraft model in Section 2. (ii) $\Theta_{\text{fov}} \geq \frac{\pi}{4}$ and $\Theta_{\text{fov}} + \Theta_{\text{sun}} \geq \frac{\pi}{2}$.

1: **if** $S_i \in G_1$ **then**
2: Rotate to align M_{S_i} with I_3
3: **else**
4: Rotate to align M_{S_i} with M_{opp}
5: **end if**
6: Wait for common start time t_s
7: Call **3-D REGION SWEEP ALGORITHM** on S_i and $R_1(S_i)$
8: Call **3-D REGION SWEEP ALGORITHM** on S_i and $R_2(S_i)$
9: Call **3-D REGION SWEEP ALGORITHM** on S_i and $R_1(S_i)$

Table 4: The SPATIAL SPACECRAFT LOCALIZATION ALGORITHM.

Lemma 4.4 (Partial reduction to in-plane search)**.** *Doing a 3π sweep (turning about the sun line) through $R_n(S)$, $n \in \{1,2\}$, $S \in G_1$, finds all spacecraft in G_2 that stay in $R_n(S)$ during the entire duration of the 3π rotation.*

Proof. Projecting the centerline of the cone and the spacecraft path onto the $\{Y,Z\}$ plane in $CMF(S)$ reduces this to the 2-D algorithm. In the cases where $R_n(S)$ contains points which project directly onto $(0,0)$ there can be a collision in the 2-D projection which does not correspond to a collision of the craft in 3-D. In these cases, the sensor cone of S_1 always contains all such points, and any colliding craft are found. □

Finally, we are in a position to establish the correctness of the full 3-D algorithm.

Theorem 4.5. *With the spacecraft model in Section 2, the* SPATIAL SPACECRAFT LOCALIZATION ALGORITHM *solves the formation initialization problem.*

Proof. Consider two spacecraft, S_1 and S_2. Let S_2 start in $R_{begin}(S_1)$ and end in $R_{end}(S_1)$. If $R_{begin}(S_1) = R_{end}(S_1)$ we are done. Otherwise S_1 must scan $R_{end}(S_1)$ at least once after the first scan of $R_{begin}(S_1)$. If the scan of $R_{begin}(S_1)$ did not find S_2, then S_2 must be in $R_{end}(S_1)$

If S_2 never crosses the $\{Y,Z\}$ plane, either the scan of $R_1(S_1)$ or the scan of $R_2(S_1)$ must find it. Otherwise, S_2 starts in one region and ends in the other. The sequence of region sweeps performed by S_1 guarantee that S_1 will scan the region S_2 starts in at least once before scanning the region S_2 ends in. If S_2 is not found when S_1 first performs a sweep of the region in which S_2 begins (call this $R_{begin}(S_1)$), then S_2 must be in the remaining region ($R_{end}(S_1)$) by the end of the sweep. Since this was the first sweep of $R_{begin}(S_1)$, S_1 must scan at $R_{end}(S_1)$ at least once after this point and find S_2. □

Remark 4.6. *The* SPATIAL SPACECRAFT LOCALIZATION ALGORITHM *sweeps a total solid angle of $9\pi + \frac{5\Theta_{tilt}}{\sin\Theta_{fov}}$ and performs rotations totaling $9\pi + 5\Theta_{tilt}$, where $\Theta_{tilt} = \min(\frac{\pi}{2} - \Theta_{fov}, \Theta_{sun})$.* •

4.3 HALF TWIST ALGORITHM

We have discovered a set of alternative algorithms for the case where the sensor cone half-angle (Θ_{fov}) is wider than $\frac{\pi}{4}$. These algorithms are in the same flavor as those presented above, and have a common relation to the problem of "Shortest Common Supersequence" in computer science. We call them "Half-Twist Algorithms" due to their use of 180 degree sweeps of the sensor cone.

4.3.1 Preliminaries

For the duration of this paper the symbol"$\oplus$" will refer to the operator "bitwise exclusive or".

Definition 4.7. *The "bitwise exclusive or" of two numbers (denoted here by $n \oplus m$) is defined by taking the binary representations of m and n and adding each successive digit modulo 2. In other words the ith digit of $n \oplus m$ is defined by $(n \oplus m)_i = (n_i + m_i \mod 2)$.*

For the HALF TWIST ALGORITHM, the following region definitions will be used.

Definition 4.8. *Let S be a spacecraft. For $n \in \{0,1,2,3\}$, define*

$$R_{twist,n}(S) = \{v \in CMF(S) \ : \ (v \cdot Y_S)(1-2(n \mod 2)) \leq 0 \\ \text{and } (v \cdot X_S)(1-2(\lfloor n/2 \rfloor \mod 2)) \leq 0\}.$$

Remark 4.9. *If two regions, $R_{twist,i}(S)$ and $R_{twist,j}(S)$, abut at the $\{Y,Z\}$ plane, then $j \oplus i = 2$. Likewise, if $R_{twist,i}(S)$ and $R_{twist,j}(S)$ abut at the $\{X,Z\}$ plane, then $j \oplus i = 1$.*

Remark 4.10. *In the frame $CMF(S)$, $R_{twist,0}(S) \cup R_{twist,1}(S) \cup R_{twist,2}(S) \cup R_{twist,3}(S) = \mathbb{R}^3$*

A diagram of the regions can be found in Figure 3. This diagram is from a viewpoint looking directly down the Z axis in $CMF(S)$ where S is the spacecraft shown in the diagram. Note that any two regions, $R_{twist,i}(S)$ and $R_{twist,j}(S)$, which share a border along the YZ plane have the property that $i \oplus j = 2$, and any two regions ,$R_{twist,k}(S)$ and $R_{twist,l}(S)$, sharing a border along the XZ lane have the property that $k \oplus l = 1$. This property of the labeling will be used in the path enumeration algorithm presented in Table 5.

4.3.2 2d Lemmas

Lemma 4.11. *Let there be two spacecraft, S_1 and S_2, traveling in straight lines in $\mathbb{R}^2$ with constant velocity, maintaining the constraint $\vec{v}_{sen}(S_1) \cdot \vec{v}_{sen}(S_2) = -1$. Let the ray defined by $\vec{v}_{sen}(S_1)$ perform an angular sweep of less then 2π and let this ray be $\vec{r}_{begin}$ at the beginning of the sweep and $\vec{r}_{end}$ at the end of the sweep. Let Ω be the area swept out by the ray defined by $\vec{v}_{sen}(S_1)$. If S_2 is initially in Ω, then at the end of the sweep either S_2 has been found, or S_2 has crossed $\vec{r}_{end}$.*

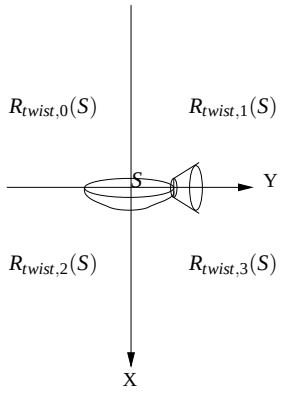

Figure 3: Cross section of regions for alternative algorithm.

Proof. Let $\vec{v}_{12} := P_{S_2} - P_{S_1}$ in $CMF(S_1)$. Consider $\angle(\vec{v}_{12}, \vec{v}_{\text{sen}}(S_1))$. Without loss of generality, this quantity must be positive at the beginning of the sweep. Likewise, at the beginning of the sweep, $\angle(\vec{v}_{12}, \vec{r}_{begin}) > 0$ and $\angle(\vec{v}_{12}, \vec{r}_{end}) < 0$. So long as S_1 and S_2 do not collide, $\angle(\vec{v}_{12}, \vec{v}_{\text{sen}}(S_1))$ varies continuously with respect to time. At the end of the sweep $\angle(\vec{v}_{\text{sen}}(S_1), \vec{r}_{end}) = 0$. Since $\angle(\vec{v}_{\text{sen}}(S_1), \vec{r}_{end})$ also varies continuously, either $\angle(\vec{v}_{12}, \vec{r}_{end})$ or $\angle(\vec{v}_{12}, \vec{v}_{\text{sen}}(S_1))$ must have reached 0 by the end of the sweep. □

As a side note, if S_1 sweeps out a region of angle 2π, then S_2 can cross $\vec{r}_{end}$ without leaving the region.

Lemma 4.12. *Given the scenario in Lemma 4.11, let the sensor ray sweep out a region of less than π radians, and let that region contain Ω. At the end of the sweep, either S_2 has been found, or it has exited Ω*

Proof. Let $\vec{r}_{begin}$ and $\vec{r}_{end}$ be defined as in Lemma 4.11. For S_2 to have evaded detection, it must have crossed $\vec{r}_{end}$ from the direction of the swept cone. Since the sweep angle is less than π, it cannot have re-entered Ω. □

4.3.3 Regions in 3d space and partial reduction to 2d lemmas

Lemma 4.13. *Let there be two spacecraft, S_1 and S_2, maintaining the constraint $M_{S_1}^{-1} M_{S_2} = M_{opp}$. Let $[0,0,1]M_{S_1}[0,0,1]^T = \pm 1$ and $[1,0,0]^T = M_{S_1}[1,0,0]^T$. Let there be an integer $n \in \{0,1,2,3\}$ indicating a region to be swept. Let S_1 tilt by $[0,0,1]M_{S_1}[0,0,1]^T(-1)^{(1+\lfloor \frac{n}{2} \rfloor)}\Theta_{tilt}$ about Y, then rotate by $[0,0,1]M_{S_1}[0,0,1]^T(-1)^n\pi$ about X and finally tilt by $[0,0,1]M_{S_1}[0,0,1]^T(-1)^{(1+\lfloor \frac{n}{2} \rfloor)}\Theta_{tilt}$ about Y. Let S_1 maintain $M_{S_1}^{-1} M_{S_2} = M_{opp}$ during the course of this maneuver. If S_2 is in $R_{twist,n}(S_1)$ during this maneuver, then at the end of the maneuver, either S_2 has been found, or S_2 has left $R_{twist,n}(S_1)$.*

Proof. Consider the projection of the sensor cone centerline onto the YZ plane at some moment in time. Any point within $R_{twist,n}(S_1)$ that projects onto this line is in the region of the sensor cone at this moment in time. Since S_2 proceeds in a straight path with

Name:	Path enumerator
Goal:	Enumerate the set of traversal sequences resulting from a spacecraft traveling in a straight line through the regions defined in Section 4.3.3
Assumes:	(i) Craft travel in straight lines and can thus cross each plane exactly once (ii) Labels l_{XZ} and l_{YZ} have been given to each plane such that each label is a distinct power of 2 as in Remark 4.9.

1: Initialize l_{result} to be an empty list. $\{l_{result}$ will hold the list of traversal sequences$\}$
2: **for all** Regions R **do**
3: Insert the sequence $(R, R \oplus l_{XZ}, R \oplus l_{XZ} \oplus l_{YZ})$ into l_{result}
4: Insert the sequence $(R, R \oplus l_{YZ}, R \oplus l_{YZ} \oplus l_{XZ})$ into l_{result}
5: **end for**
6: Return l_{result} (the list of possible traversal sequences)

Table 5: Path enumeration algorithm.

constant velocity, its projection down onto the YZ plane is also a straight path with constant velocity. If S_2 crosses the YZ plane then it has automatically left $R_{twist,n}(S_1)$, otherwise this reduces to Lemma 4.12. □

4.3.4 Use of "Shortest Common Supersequence" algorithm

Note that the 4 regions defined in the previous section (see Figure 3) are divided by the XZ and YZ planes. Any path taken by a craft traveling in a straight line can cross each of these planes at most once. We will consider paths that cross both of these planes, as region traversal sequences corresponding to paths that cross one or fewer plane can be thought of as subsequences of region traversal sequences corresponding to paths that cross two planes.

Constructing the set of such traversal sequences is quite easy, and an algorithm to do so is described in Table 5.

Lemma 4.14. *Construction of a sequence of regions to sweep that contains each possible traversal sequence as a (non-contiguous) subsequence yields an algorithm that provably finds S_2 regardless of the path that S_2 takes.*

Proof. Consider an arbitrary path p and its traversal sequence $s_p = (s_{p1}, s_{p2}, ..., s_{pn})$. Let $s_{\text{sweep}} = (s_{\text{sweep},1}, s_{\text{sweep},2}, \cdots, s_{\text{sweep},m})$ be the sequence of regions swept out by S_1 during the course of the algorithm. Let $s_{sub} = (s_{sub,1}, s_{sub,2}, ..., s_{sub,n})$ be a subsequence of the sequence of s_{sweep} that is exactly equal to s_p (by construction, this must exist). After S_1 sweeps s_{p1}, spacecraft B must either have been found or be further along in its path than s_{p1}. Using an inductive hypothesis stating that after S_1 has swept s_{pi}, S_2 must be further along in its path than s_{pi}, we can easily prove that when S_1 begins its sweep of $s_{p(i+1)}$, B is either in the segment of its path corresponding to $s_{p(i+1)}$ or further along

Name:	Shortest common supersequence
Goal:	Find the shortest common supersequence(s) of n sequences.
Assumes:	(i) Input is of the form $(s_1, s_2, ..., s_n)$ (ii) Each s_i is comprised of $(s_{i,1}, s_{i,2}, ...s_{i,length(s_i)})$ (iii) An integer n_{max} is provided as an additional input, to specify the max length of candidate solutions. (iv) Sequence items range over the alphabet Σ

1: Set the list $l_{result} := \emptyset$
2: Set $n_{minlength} := n_{max}$
3: Let Π be the set of sequences of length n_{max} of symbols from Σ
4: **for all** Sequences, $p_k \in \Pi$ **do**
5: **if** p_k contains each s_i as a subsequence and length(p_k) $\leq n_{minlength}$ **then**
6: **if** length(p_k) $< n_{minlength}$ **then**
7: Set $n_{minlength}$:= length(p_k)
8: Set $l_{result} := \emptyset$
9: **end if**
10: Set l_{result} := append(l_{result}, p_k)
11: **end if**
12: **end for**
13: return l_{result}

Table 6: Shortest common supersequence algorithm.

its path. If S_2 is further along its path at the beginning of the sweep, then it remains so at the end. Otherwise either S_1 finds S_2, or S_2 has progressed further in its path. Since the sequence of regions S_2 travels through is finite, at the end of this operation, S_2 has been found. □

What remains is to come up with a way to find such a sequence of regions to sweep. The problem of finding a sequence S given a set of sequences $\{S_1, S_2, etc\}$ such that S contains each S_i as a subsequence is known as the "shortest common supersequence" problem in computer science. It is closely related to the "shortest common subsequence" problem. While this can be solved in polynomial time for a set of two sequences, in general it should be noted that "shortest common supersequence" is NP-complete [18]. This is actually not a problem for spacecraft initialization, as the input size is small, and the solution can be computed offline, before the robots are sent into space. It should be clarified here that "shortest common subsequence" is not part of the formation initialization algorithm we intend to present in this section, but rather a method of arriving at a particular formation initialization algorithm. One method of solving the "shortest common supersequence problem is presented in Table 6.

The algorithm we use for "shortest common supersequence" is not optimized for speed. For cases where more regions of space need to be considered, a fast approximation to "shortest common supersequence" might need to be substituted.

4.3.5 HALF TWIST ALGORITHM

Running "shortest-common-supersequence" on the possible sequences for this problem yields a solution of

$$(0,1,2,3,0,1,2,0)$$

This particular sequence is convenient, as many of the pairs of region sweeps can be composed into single 360 degree rotations (for example (0,1) or (2,3) can be composed in this way. Note that the algorithm described in Section 4.2 contains 9 rotations of 180 degrees, while this algorithm needs 8 such rotations. However, each algorithm has different patterns of tilting up and down. In Section 4.3.5 we will demonstrate that the differences between these patterns amount to fewer than 180 degrees of rotation.

Name:	HALF TWIST SWEEP
Goal:	Scan a region for use as a subroutine by HALF TWIST ALGORITHM
Inputs:	(i) A spacecraft, S_i (ii) An integer, n, indicating the region to be initially swept (e.g. 3 means $R_3(S_i)$) (iii) An integer $n_{regions}$ which is set to 2 if we are to continue and sweep $R_{twist,n\oplus 1}(S_i)$ and set to 1 otherwise
Assumes:	(i) Spacecraft model in Section 2 (ii) $\Theta_{\text{fov}} \geq \frac{\pi}{2}$.

Require: At the beginning of the algorithm, $\forall S_i \in G_1$, $[0,0,1]M_{S_1}[0,0,1]^T = \pm 1$ and $[1,0,0]^T = M_{S_1}[1,0,0]^T$

Require: $\forall S_i \in G_1, S_j \in G_2$, S_i and S_j maintain $M_{S_i}^{-1} M_{S_j} = M_{\text{opp}}$.

1: **if** $S \in G_1$ **then**
2: Rotate by $[0,0,1]M_S[0,0,1]^T(-1)^{(1+\lfloor \frac{n}{2} \rfloor)}\Theta_{\text{tilt}}$ about Y.
3: Rotate by $[0,0,1]M_{S_1}[0,0,1]^T(-1)^n\pi$ about X.
4: Rotate by $[0,0,1]M_{S_1}[0,0,1]^T(-1)^{(1+\lfloor \frac{n}{2} \rfloor)}\Theta_{\text{tilt}}$ about Y.
5: **else**
6: Rotate by $[0,0,1]M_S[0,0,1]^T(-1)^{\lfloor \frac{n}{2} \rfloor}\Theta_{\text{tilt}}$ about Y.
7: Rotate by $[0,0,1]M_{S_1}[0,0,1]^T(-1)^n\pi$ about X.
8: Rotate by $[0,0,1]M_{S_1}[0,0,1]^T(-1)^{\lfloor \frac{n}{2} \rfloor}\Theta_{\text{tilt}}$ about Y.
9: **end if**

Table 7: HALF TWIST SWEEP.

Theorem 4.15. *The* HALF TWIST ALGORITHM *described in Table 8 completes formation initialization.*

Proof. Reduction to the two-spacecraft sky-search can be done as in previous proofs (see Section 4.2). By Lemma 4.12 each 180 degree sweep finds all spacecraft that begin the sweep in a particular region and stay in the region. Since the sequence of region

Name:	HALF TWIST ALGORITHM
Goal:	Solve the formation initialization problem using less total rotation then algorithm in Section 4.
Assumes:	(i) Spacecraft model in Section 2 (ii) $\Theta_{\text{fov}} \geq \frac{\pi}{2}$.

1: **if** $S_i \in G_1$ **then**
2: Rotate to align M_{S_i} with I_3
3: **else**
4: Rotate to align M_{S_i} with M_{opp}
5: **end if**
6: Wait for common start time t_s
7: Call HALF TWIST SWEEP on S_i, $n = 0$, $n_{regions} = 2$
8: Call HALF TWIST SWEEP on S_i, $n = 2$, $n_{regions} = 2$
9: Call HALF TWIST SWEEP on S_i, $n = 0$, $n_{regions} = 2$
10: Call HALF TWIST SWEEP on S_i, $n = 2$, $n_{regions} = 1$
11: Call HALF TWIST SWEEP on S_i, $n = 0$, $n_{regions} = 1$

Table 8: HALF TWIST ALGORITHM.

sweeps has been chosen to match a shortest common supersequence for all possible region traversal sequences of the other spacecraft, all spacecraft have been found (see Lemma 4.14). □

Angular distance efficiency of 3d algorithm

The algorithm described in the previous section makes 3 rotations of 3π radians each, and the equivalent of 5 tilts of Θ_{tilt}. The algorithm described in this section makes the equivalent of 8 rotations of π radians each and 9 tilts of Θ_{tilt}.

The difference between the total angle traveled in these cases is $9\pi + 5\Theta_{\text{tilt}} - (8\pi + 9\Theta_{\text{tilt}})$ leading to a difference of $\pi - 4\Theta_{\text{tilt}}$ radians. If $\Theta_{\text{tilt}} < \frac{\pi}{4}$ then the new algorithm requires less total angular rotation. Since Θ_{fov} must be greater than or equal to $\frac{\pi}{4}$ for this algorithm to work in the first place, and Θ_{tilt} can be anything greater than $\frac{\pi}{2} - \Theta_{\text{fov}}$, Θ_{tilt} can be chosen to yield a shorter total angular path than SPATIAL SPACECRAFT LOCALIZATION ALGORITHM.

4.4 WAIT AND CHECK ALGORITHM

As pointed out in Remark 4.6, the provably correct SPATIAL SPACECRAFT LOCALIZATION ALGORITHM is far from optimal both in terms of total angle traversed and solid angle covered. In what follows, we introduce the WAIT AND CHECK ALGORITHM (cf. Table 9). This algorithm has a much better performance with regards to solid angle covered, at the expense of a longer execution time. After establishing its correctness in Theorem 4.17, we show how to modify it to achieve an optimal total rotation given its solid angle covered (cf. Remark 4.18).

Name:	WAIT AND CHECK ALGORITHM
Goal:	Solve the formation initialization problem using near-optimal solid angle coverage.
Assumes:	(i) Spacecraft model in Section 2. (ii) $\Theta_{\text{fov}} > \frac{\pi}{4}$.

1: Define $\Theta_\varepsilon = \Theta_{\text{fov}} - \frac{\pi}{4}$
2: **if** $S_i \in G_1$ **then**
3: Rotate to align M_{S_i} with I_3
4: **else**
5: Rotate to align M_{S_i} with M_{opp}
6: **end if**
7: Wait for common start time t_s
8: Rotate by $\frac{\pi}{4}$ about $Y_{S_i}\{$Call the time at the end of this step $t_1\}$
9: Rotate about X_{S_i} by 2π with angular velocity $\omega\{$Call this time $t_2\}$
10: Wait $\frac{\tan(\frac{\pi}{2}-\Theta_\varepsilon)}{\Theta_\varepsilon}(t_2 - t_1)\{$Call this time $t_3\}$
11: Rotate about Y_{S_i} by $\frac{-\pi}{2}\{$Call this time $t_4\}$
12: Rotate about X_{S_i} by 3π with angular velocity $\omega\{$Call this time $t_5\}$
13: Rotate about Y_{S_i} by $\frac{-\pi}{2}\{$Call this time $t_6\}$
14: Wait $\frac{\tan(\frac{\pi}{2}-\Theta_\varepsilon)}{\Theta_\varepsilon}(t_5 - t_1)\{$Call this time $t_1\}$
15: Rotate about X_{S_i} by 2π with angular velocity $\omega\{$Call this time $t_7\}$

Table 9: The WAIT AND CHECK ALGORITHM.

The next lemma will be used in establishing the correctness of the WAIT AND CHECK ALGORITHM.

Lemma 4.16. *Consider a spacecraft S_2 traveling in a path with respect to S_1 with velocity V_{S_2} and point of closest approach $p_{cl}(S_1,S_2)$. Let $\Pi_{1,2}$ be the plane in $CMF(S_1)$ spanned by the vectors $p_{cl}(S_1,S_2)$ and V_{S_2}. Define a parameterization of vectors in $\Pi_{1,2}$ by the function $\Theta_{scan}(P) = \arctan(p_{cl}(S_1,S_2)\cdot P, -V_{S_2}\cdot P)$. For any angles $\Theta \in [0,\pi]$ and $\varepsilon \in [0,\Theta]$, if S_1 first verifies that $\Theta_{scan}(P_{S_2}) < \Theta - \varepsilon$ at time t_1 and then verifies that $\Theta_{scan}(P_{S_2}) > \Theta + \varepsilon$ at time t_2, then by time $t_2 + \frac{\tan(\frac{\pi}{2}-\varepsilon)}{\varepsilon}(t_2-t_1)$, S_2 will be within ε of its final angle.*

Proof. Since $\Theta_{scan}(P_{S_2}(t2)) - \Theta_{scan}(P_{S_2}(t1)) > 2\varepsilon$, $\frac{\|V_{S_2}\|}{\|p_{\mathrm{cl}}(S_1,S_2)\|}$ is at least $\frac{2\varepsilon}{t_2-t_1}$. $\Theta_{scan}(P_{S_2}(\frac{\tan(\frac{\pi}{2}-\varepsilon)}{\varepsilon}(t_2-t_1))) \geq \arctan(\tan(\Theta+\varepsilon) + \frac{2\varepsilon}{t_2-t_1}\frac{\tan(\frac{\pi}{2}-\varepsilon)}{\varepsilon}(t_2-t_1)) \geq \pi - \varepsilon$. □

Next, we characterize the correctness of the WAIT AND CHECK ALGORITHM.

Theorem 4.17. *The* WAIT AND CHECK ALGORITHM *correctly solves the formation initialization problem.*

Proof. Consider a spacecraft, S_1. Any other spacecraft whose X position is less than zero at time t_1 must either be found, cross the $\{Y,Z\}$ plane, or cross the $\{X,Z\}$ plane before t_2. If S_2 crossed the $\{X,Z\}$ plane between t_1 and t_2 and was not found, then it must have been moving with sufficient velocity to have moved to within Θ_ε of its final angle by time t_3. By this logic, by t_3, any craft with a final angle corresponding to a positive X component of position must have been found by time t_2, or be on the $+X$ side of the $\{Y,Z\}$ plane by time t_3. Between t_4 and t_5 all such craft are found, along with any craft that started on the $+X$ side of the $\{Y,Z\}$ plane and have not left it by t_5 (by Lemma 4.16). Any craft which have left the $+X$ side of the $\{Y,Z\}$ plane by t_5 but were not found during the sweep of the $-X$ half of the $\{Y,Z\}$ plane must have been moving with sufficient angular velocity as to be within Θ_ε of their final angles (on the $-X$ half of the $\{Y,Z\}$ plane) by t_6 (cf. Lemma 4.16). For this reason, the final sweep of the $-X$ side of the $\{Y,Z\}$ plane need only be a 2π sweep. □

Remark 4.18 (Angle-optimal region sweeps)**.** *The* WAIT AND CHECK ALGORITHM *covers a solid angle of $7\pi + \frac{5\Theta_{tilt}}{\sin(\Theta_{tilt})}$. Clearly, the ratio of total angle traversed to solid angle covered in the* WAIT AND CHECK ALGORITHM *is not at the optimal $\frac{1}{2\sin(\Theta_{fov})}$. The algorithm can be modified to traverse a total angle of $7\pi\sin(\Theta_{tilt}) + 5\Theta_{tilt}$, where $\Theta_{tilt} = \min(\pi/2 - \Theta_{fov}, \Theta_{sun}, \Theta_{fov})$, at the expense of not respecting the sun-angle constraint. We describe how next. The optimal ratio of total angle traversed to solid angle covered is achievable for any rotational trajectory of $\vec{v}_{sen}(S)$ over time. While a rotational velocity, ω, specifies the instantaneous rotation of the entire body frame of S, the instantaneous motion of $\vec{v}_{sen}(S)$ only fixes two degrees of freedom of this rotation. By choosing ω to lie along $\vec{v}_{sen}(S) \times \frac{d}{dt}\vec{v}_{sen}(S)$, we can always achieve the maximum instantaneous $f_{sld}(\omega)/\|\omega\|$.*

Let us suppose that $\vec{v}_{sen}(S)$ is within an angle of $\frac{\pi}{2} - \alpha$ of the sun line, and we wish for $\vec{v}_{sen}(S)$ to sweep out the arc defined by $C_\alpha = \{\vec{v} \in \mathbb{R}^3 : \|\vec{v}\| = 1 \wedge \arccos(\vec{v}\cdot\vec{v}_{SUN}) =$

$\frac{\pi}{2} - \alpha\}$. *At any instant during which* $\vec{v}_{sen}(S) \in C_\alpha$*, the optimal axis of rotation,* ω*, is both perpendicular to* $\vec{v}_{sen}(S)$ *and guarantees* $\vec{v}_{sen}(S)$ *remains in* C_α*. One such* ω *always lies on a cone which we will define as* $C_{tumble} = \{\vec{v} \in \mathbb{R}^3 \ : \ \|\vec{v}\| = 1 \wedge \arccos(\vec{v} \cdot \vec{v}_{SUN}) = \alpha\}$*, see Figure 4. Note that the body frame,* $BF(S)$ *does not move*

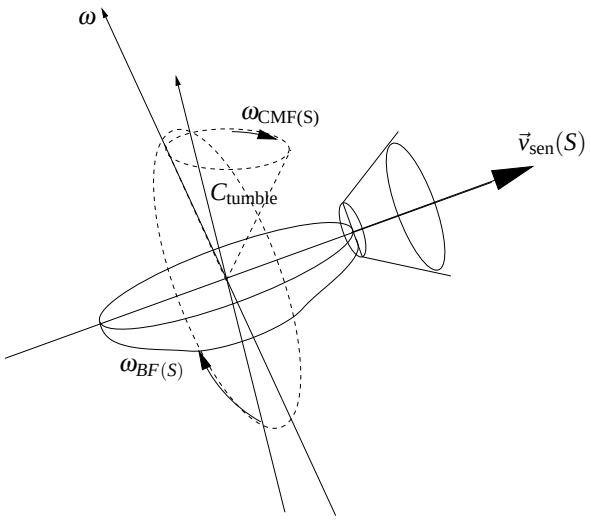

Figure 4: Performing a sweep of 2π with less then 2π rotation

with respect to $CMF(S)$ *at any point along the axis* ω*. When the sweeps about the sun line of the* WAIT AND CHECK ALGORITHM *are executed as we just described, the algorithm requires a total angular rotation of* $\frac{7\pi}{\sqrt{2}} + 5\Theta_{tilt}$*.* •

5 Conclusions and future work

We have considered the formation initialization problem for a group of spacecraft endowed with limited field-of-view relative position sensors and omnidirectional communication. We have obtained optimality bounds for the performance of any correct algorithm in terms of worst-case solid angle covered and total angle traversed. In 2-D, the angle traversed bound is hard and in 3-D, the angle traversed bound is no worse than the solid angle bound. Our analysis of optimality justifies several decisions made in both our own algorithm designs and those of previous works, including the PLANAR SPACECRAFT LOCALIZATION ALGORITHM and the **Opposing Sensor Constraint**. We have also synthesized three provably correct formation initialization algorithms. In particular, the SPATIAL SPACECRAFT LOCALIZATION ALGORITHM is simple and easily provable, while the WAIT AND CHECK ALGORITHM is nearly optimal according to the optimality bounds obtained. In addition, HALF TWIST ALGORITHM gives a tighter angle bound then SPATIAL SPACECRAFT LOCALIZATION ALGORITHM without the wait times of WAIT AND CHECK ALGORITHM at the expense of requiring a more complicated search procedure.

Areas of future work will include (i) the determination of the optimality of **Opposing Sensor Constraint** when the spacecraft start in random orientations (this is easily seen for the case of two spacecraft). If this is true in general, then it will be of interest to determine the optimal way to move the spacecraft to satisfy the **Opposing Sensor Constraint**; (ii) the investigation of other notions of optimality, such as minimum time to complete formation initialization on a fixed fuel budget; (iii) the determination of whether the 6π solid angle bound in 3-D is a hard bound. For $\Theta_{\text{fov}} = \frac{\pi}{2}$, this bound gives a total angle rotated bound of 3π, which matches the intuitive result from reducing this special subproblem to 2-D.

Acknowledgments

The authors wish to thank Prof. Roy Smith for pointing our attention to the algorithms described in [1]. This research was supported in part by NASA University Aligned Research Program Award TO.030.MM.D.

References

[1] D. P. Scharf, S. R. Ploen, F. Y. Hadaegh, J. A. Keim, and L. H. Phan, "Guaranteed initialization of distributed spacecraft formations," in *AIAA Guidance, Navigation and Control Conference*, Austin, TX, 2003.

[2] H. Hemmati, W. Farr, B. Liu, J. Kovalik, M. Wright, and J. Neal, "Formation metrology," Nov. 2003, high-level description of sensors developed at JPL for precision gormation control. [Online]. Available: http://dst.jpl.nasa.gov/metrology/index.htm

[3] M. Tillerson, G. Inalhan, and J. How, "Coordination and control of distributed spacecraft systems using convex optimization techniques," *International Journal on Robust and Nonlinear Control*, vol. 12, pp. 207–242, 2002.

[4] P. R. Lawson, S. C. Unwin, and C. A. Beichman, "Precursor science for the terrestrial planet finder," Jet Propulsion Laboratory Publication 04-14, Oct. 2004. [Online]. Available: http://planetquest.jpl.nasa.gov/documents/RdMp273.pdf

[5] F. Bauer, J. Bristow, D. Folta, K. Hartman, D. Quinn, and J. How, "Satellite formation flying using an innovative autonomous control system (autocon) environment," in *AIAA Conf. on Guidance, Navigation and Control*, Reston, VA, 1997, pp. 657–666.

[6] A. Das and R. Cobb, "Techsat 21 - space missions using collaborating constellations of satellites," in *AIAA Conf. on Small Satellites*, Logan, UT, 1998.

[7] J. How, R. Twiggs, D. Weidow, K. Hartman, and F. Bauer, "Orion - a low cost demonstration of formation flying in space using GPS," in *AIAA/AAS Astrodynamics Specialist Conf. and Exhibit*, Reston, VA, 1998, pp. 276–286.

[8] J. A. Dooley and P. R. Lawson, "Technology plan for the terrestrial planet finder coronagraph," Jet Propulsion Laboratory Publication 05-8, Mar. 2005. [Online]. Available: http://planetquest.jpl.nasa.gov/TPF/TPF-CTechPlan.pdf

[9] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, "A survey of spacecraft formation flying guidance and control (part ii): control," in *American Control Conference*, Boston, MA, June 2004, pp. 2976–2985.

[10] V. Kapila, A. G. Sparks, J. M. Buffington, and Q. Yan, "Spacecraft formation flying: dynamics and control," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 23, pp. 561–564, 2000.

[11] M. Mesbahi and F. Y. Hadaegh, "Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 369–377, 2001.

[12] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 6, pp. 777–790, 2001.

[13] I. I. Hussein, "Motion planning for multi-spacecraft interferometric imaging systems," Ph.D. dissertation, University of Michigan, 2005.

[14] M. Schuresko and J. Cortés, "Correctness analysis and optimality bounds of multi-spacecraft formation initialization algorithms," in *IEEE Conf. on Decision and Control*, San Diego, CA, Dec. 2006, pp. 5974–5979.

[15] N. A. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 1997.

[16] J. E. Marsden and T. S. Ratiu, *Introduction to Mechanics and Symmetry*, 2nd ed. New York: Springer Verlag, 1999.

[17] K. Räihä and E. Ukkonen, "The shortest common supersequence problem over the binary alphabet is NP-complete," *Theoretical Computer Science*, vol. 16, no. 2, pp. 187–198, 1981.