Simultaneous Control and Trajectory Estimation for Collision Avoidance of Autonomous Robotic Spacecraft Systems*

Matthew King-Smith¹, Panagiotis Tsiotras², Frank Dellaert³

Abstract—We propose factor graph optimization for simultaneous planning, control, and trajectory estimation for collision-free navigation of autonomous systems in environments with moving objects. The proposed online probabilistic motion planning and trajectory estimation navigation technique generates optimal collision-free state and control trajectories for autonomous vehicles when the obstacle motion model is both unknown and known. We evaluate the utility of the algorithm to support future autonomous robotic space missions.

I. INTRODUCTION

On-orbit satellite servicing (OSS) holds the promise to refuel, maintain, upgrade, and repair existing spacecraft, enable space construction, and actively remove orbital debris [1]–[3]. During an OSS mission, which involves close proximity operations with other space objects, it is imperative for the servicing spacecraft to be able to adapt to a changing environment while simultaneously achieving the primary mission objective. As such, the vehicle trajectory planning should be implemented online [4], while also considering the combined information (and its uncertainty) collected from onboard system sensors [5], [6]. In fact, onboard spacecraft autonomy has been identified by the National Aeronautics Space Administration and the European Space Agency as a necessary technology for future space missions [7], [8].

To achieve the desired onboard spacecraft autonomy, there is a need for both online trajectory estimation and planning, given model and sensor uncertainty [7]. Traditional online trajectory optimization methods for spacecraft navigation usually solve the optimal control problem [9]–[11], by performing vehicle trajectory estimation independently from planning [5]. However, such two-step optimization processes may potentially lead to suboptimal results given that both the estimation and planning problems can be viewed as variants of a single trajectory optimization problem.

Alternative trajectory optimization approaches, such as sampling-based motion planning (SBMP), offer techniques that combine the planning and estimation into a single optimization problem and have already been applied to spacecraft OSS applications, such as, inspection [12] and exploratory guidance about resident space objects [13]. Additionally, SBMP techniques have been used for planning collision-free spacecraft trajectories in static obstacle environments [14]. However, despite SBMP contributions to the spacecraft navigation and estimation communities, all aforementioned SBMP studies do not account for measurement uncertainty from sensors during optimization.

In the robotics community, probabilistic inference has been used to address various problems related to state estimation [15], [16], localization [17]–[20], optimal control problems [21]–[25], and path planning [26]–[28] given uncertain sensor measurements and models. In particular, Mukadam et al. [29], have used factor graphs to solve probabilistic inference problems for simultaneous trajectory estimation and planning (STEAP) for robotic systems in static environments. STEAP solves the probabilistic estimation and path planning problem in a single factor graph, enabling information to quickly flow between the two problems during online optimization [30].

Building on this prior work, this research proposes simultaneous control and trajectory estimation (SCATE) for online probabilistic trajectory motion planning and estimation for autonomous satellites in environments with moving obstacles that addresses some of the shortcomings of STEAP, such as, lacking robot dynamics and planning restricted to static obstacle environments. Specifically, this work contributes the following advancements over STEAP:

1) Adding realistic vehicle dynamics to factor graphs.
2) Collision-free path planning and trajectory estimation in environments with moving obstacles.
3) Evaluation of SCATE navigation on a physical autonomous spacecraft simulator platform [31], [32].

The remainder of the paper is organized as follows: Section II establishes the mathematics necessary for SCATE trajectory optimization, Section III describes the robotic spacecraft platform and testing facility, Section IV presents the results of real-time SCATE navigation. Section V reviews the viability of SCATE navigation for OSS missions, discusses some of its limitations, and suggests some potential avenues for future work.

II. APPROACH AND METHODOLOGY

In this section, we review the use of factor graphs for probabilistic inference for trajectory optimization. We then introduce STEAP, a factor graph motion planning approach for collision avoidance of kinematic robotic systems in static environments. Motivated by STEAP, we propose SCATE, a new algorithm for collision-free navigation of dynamic robotic systems in environments with moving obstacles using factor graphs.
A. Trajectory Optimization as Probabilistic Inference

We represent a trajectory as a continuous-valued function that maps time $t$ to robot states $x(t)$ so as to determine the maximum a posteriori (MAP) continuous-time trajectory of $x(t)$ given a prior distribution on the space of state trajectories and a likelihood function.

1) Trajectory Prior: A prior distribution over trajectories can be defined as a vector-valued Gaussian process (GP) $x(t) \sim GP(\mu(t), K(t, t'))$, where $\mu(t)$ is the vector-valued mean function and $K(t, t')$ is a matrix-valued covariance function. For any collection of times $t = \{t_0, \ldots, t_N\}$, $x$ has a joint Gaussian distribution $x \triangleq [x_0 \cdots x_N]^T \sim N(\mu, K)$, with mean vector $\mu$ and covariance $K$ defined as

$$\mu \triangleq [\mu(t_0) \cdots \mu(t_N)]^T, \quad K \triangleq [K(t_i, t_j)]_{i,j,0 \leq i,j \leq N}. \quad (1)$$

The prior distribution is then defined by the GP mean $\mu$ and the covariance $K$ as

$$p(x) \propto \exp \left\{ -\frac{1}{2} \| x - \mu \|^2_K \right\}. \quad (2)$$

Information known a priori is then encoded with such priors.

2) Likelihood Function: Let $z$ be a collection of binary events, where events are defined as such a collision, or receiving a sensor measurement. The likelihood function is the conditional distribution $l(x; z) = p(z|x)$, which specifies the probability of events $z$ given a trajectory $x$. We define the likelihood as a distribution in the exponential family

$$l(x; z) \propto \exp \left\{ -\frac{1}{2} \| h(x, z) \|^2_\Sigma \right\}, \quad (3)$$

where $h(x, z)$ is a cost function with covariance matrix $\Sigma$.

3) Computing the Maximum A Posteriori Trajectory: Using Bayes rule, we express the posterior distribution of the trajectory given the events in terms of the prior and the likelihood as $p(x|z) \propto p(x)p(z|x)$. Then, we can compute the MAP of the set $\Theta \triangleq \{ x \}$ as $\Theta^* = \arg \max_{x \in \Theta} p(x|z) = \arg \max_{x \in \Theta} \{ p(x)p(z|x) \} = \arg \min_{z \in \Theta} \left\{ -\log(p(x)p(z|x)) \right\}$ or

$$\Theta^* = \arg \min_{\Theta} \left\{ \frac{1}{2} \| x - \mu \|^2_K + \frac{1}{2} \| h(x, z) \|^2_\Sigma \right\}. \quad (4)$$

where the last expression follows from (2) and (3). The underlying sparsity of the problem is computationally exploited by formulating (4) into an inference problem on a graphical model [15].

4) Factor Graphs for Estimation and Planning: A computationally efficient way to compute the MAP trajectory given in (4) is to exploit the known structure of the problem by representing the posterior distribution as a factor graph. As shown in [33], a factor graph allows for any distribution to be factored into a product of functions that is organized as a bipartite graph $G = (\Theta, \Phi, E)$. The graph consists of factor nodes $\Phi \triangleq \{ \phi_0, \ldots, \phi_V \}$, variable nodes given by the set $\Theta \triangleq \{ x \}$, and edges $E$ which connect the two types of nodes, as shown in Figure 1. Letting $\Theta_i$ be a variable subset of $\Theta$, then the posterior distribution can be expressed as the product of the factors

$$p(x) \propto \prod_{i=0}^{V} \phi_i(\Theta_i). \quad (5)$$

Factor graphs that are sparse lend to sparse precision matrices, which are exploitable, yielding a computationally efficient way to determine $\Theta^*$ from (4) [15].

B. Simultaneous Trajectory Estimation and Planning

The simultaneous trajectory estimation and planning (STEAP) algorithm proposed by Mukadam et al. [29], is a unified probabilistic framework constructed via factor graphs for both past state estimation and future path planning of robotic systems. The state trajectory, $x$, is represented by the GP prior given in (2), with mean and covariance functions $\mu, K$, given all sensor data and cost information collected into a single likelihood. The posterior distribution is thought to represent events that happen in the past and in the future simultaneously and is represented by

$$p(\hat{x} | \hat{\mu}) \propto \phi_{\text{mean}} \phi_{\text{obs}} \phi_{\text{fix}}, \quad (6)$$

where $\phi_{\text{mean}}, \phi_{\text{obs}}, \phi_{\text{fix}}$ are the GP prior, state measurement, obstacle, and goal factors, respectively, of the graph, as shown in Figure 1, and are defined in Section II-C.1.

![Fig. 1: STEAP factor graph for trajectory estimation and planning.](image)

The MAP solution to the factor graph formulation given in (6), i.e., $\Theta^* = \{ \hat{x}, \hat{\mu} \}$, solves both the estimation (i.e., $\hat{x}$) and planning (i.e., $\hat{\mu}$) problems in a single step. As the robot transverses the trajectory over time, new measurements and cost information is used to appropriately update the likelihood and graph for online planning and estimation [30].

STEAP, however, only considers the kinematics for robot systems that operate in static object environments. Furthermore, STEAP generates GP priors by linear time-varying stochastic differential equations maintaining constant velocity, i.e., noise-on-acceleration input [27], [29]. Such GP priors factor according to $\phi_{\text{GP}} = \prod_i \phi_{\text{GP}}(x_i, x_{i+1})$, where any GP prior factor connects to only its two neighboring states (without control), forming a (Gauss-Markov) chain.

C. Simultaneous Control and Trajectory Estimation

To account for environments with moving obstacles, we must also consider the robot (spacecraft) dynamics. Hence, we propose simultaneous control and trajectory estimation (SCATE) to determine the MAP solution to the set $\Theta \triangleq \{ x, u, l \}$, where $u$ is the control input trajectory, and $l$ is the set of obstacle-$l$ locations. We investigate both reactive and predictive SCATE navigation given an unknown and known obstacle motion model, respectively.
Finally, the MAP solution to SCATE factor graphs is given as \( \Theta^* = \{ \hat{x}, \hat{u}, \hat{l} \} \), where \( \hat{x}, \hat{u}, \hat{l} \) are estimates of \( u, x, l \) and \( \hat{x}, \hat{u}, \hat{l} \) are the planned state and control input, respectively, to execute along the remaining trajectories.

Reactive SCATE factor graphs are given by the posterior distribution

\[
p(\hat{x} \cup \hat{u} \cup \hat{l} | z) = \phi^\text{dyn} \phi^\text{meas} \phi^\text{lim} \phi^\text{obs} \phi^\text{fix},
\]

where \( \phi^\text{dyn}, \phi^\text{lim} \) are dynamic, and control limit factors, respectively, as shown in Figure 2, and are defined in Section II-C.1. Reactive SCATE factor graphs plan for state \( z \), assuming obstacles observed are static in the environment.

Predictive SCATE factor graphs, are given the obstacle trajectory, \( l \), and are expressed by the posterior distribution

\[
p(\hat{x} \cup \hat{u} \cup \hat{l} | \hat{l}) = \phi^\text{dyn} \phi^\text{meas} \phi^\text{lim} \phi^\text{obs} \phi^\text{fix}.
\]

Knowing \( \hat{l} \) in advance enables predictive SCATE to allocate cost to future obstacle locations via the remaining obstacle factors, i.e., the set \( \{ \phi^\text{obs}, \phi^\text{lim} \} \), during planning for \( x \).

\[\text{Fig. 2: SCATE factor graph for control and trajectory estimation.}\]

Figure 2 also shows the removal of outdated planning factors, i.e., the set \( \{ \phi^\text{start}, \phi^\text{lim}, \phi^\text{obs} \} \), from the graph such that they do not influence the \( t_i \)-th optimization.

1) **SCATE Factor Definitions:** We now define the factors \( \phi^\text{fix}, \phi^\text{meas}, \phi^\text{obs}, \phi^\text{dyn}, \phi^\text{lim} \) given in (6) to (8).

**Start and goal factor:** The multivariate Gaussian factors

\[
\phi^\text{start}(x_0) \propto \exp \left\{ - \frac{1}{2} \| x_0 - x_\text{start} \|_\Sigma^\text{fix}^2 \right\},
\]

\[
\phi^\text{goal}(x_N) \propto \exp \left\{ - \frac{1}{2} \| x_N - x_\text{goal} \|_\Sigma^\text{fix}^2 \right\},
\]

with the mean as the start or goal and a small covariance \( \Sigma^\text{fix} \) define \( \phi^\text{fix} = \phi^\text{start}(x_0)\phi^\text{goal}(x_N) \) and root the trajectory endpoints to the start and goal locations.

**Measurement factors:** For simplicity, we use multivariate Gaussian measurement factors for state measurements

\[
\phi^\text{meas}_{x_i}(x_i) \propto \exp \left\{ - \frac{1}{2} \| x_i - x_{\text{meas}}^i \|_\Sigma^\text{meas}^2 \right\},
\]

where, with a slight abuse of notation, \( x_{\text{meas}}^i \) denotes the \( i \)-th state measurement with covariance \( \Sigma^\text{meas} \). Assuming an obstacle is observed via relative bearing and range measurements, \( x_{\text{meas}}^i, z_{\text{meas}}^i \), with covariance matrices \( \Sigma^\text{meas}_i, \Sigma^\text{meas} \), respectively, we can then define a bearing and range factors as

\[
\phi^\text{meas}_{\theta_i}(x_i, l_i) \propto \exp \left\{ - \frac{1}{2} \| h_{\theta_i}(x_i, l_i) - z_{\theta_i}^\text{meas}^i \|_\Sigma^\text{meas}_\theta^2 \right\},
\]

\[
\phi^\text{meas}_{r_i}(x_i, l_i) \propto \exp \left\{ - \frac{1}{2} \| h_{r_i}(x_i, l_i) - z_{r_i}^\text{meas}^i \|_\Sigma^\text{meas}_r^2 \right\},
\]

where \( h_{\theta_i}(x_i, l_i) = \tan^{-1}(l_{y,i} - x_{y,i}, l_{x,i} - x_{x,i}) \) and \( h_{r_i}(x_i, l_i) = |r_i| \), respectively.

**Linear time-invariant dynamics factor:** We assume that the dynamics of the robotic system are governed by a linear time-invariant (LTI) state-space model, i.e.,

\[
\dot{x}(t) = Ax(t) + Bu(t),
\]

with the following discrete-time state transition matrix \( F \) and input matrix \( U \)

\[
x_{i+1} = Fx_i + Tu_i.
\]

\[\text{Control limit factor:} \quad \text{The control limit factor is designed to ensure that control trajectories respect limits.}\]

where \( \Sigma^\text{dy} \) is the covariance matrix of the factor.

**Obstacle factor:** The obstacle factor derives from the likelihood function, which is given by the conditional distribution \( \phi^\text{obs}_{\text{obs}}(x_i; z_i = 0) = p(z_i = 0 | x_i) \), which describes the probability of being clear of collisions, or the probability that collision event \( z_i = 0 \), given the current configuration \( x_i \). This likelihood is represented as a distribution in the exponential family

\[
\phi^\text{obs}_{\text{obs}}(x_i) = \phi^\text{obs}(x_i; z_i = 0) \propto \exp \left\{ - \frac{1}{2} \| h_{\text{obs}}(x_i) \|_\Sigma^\text{obs}^2 \right\},
\]

where \( h_{\text{obs}}(x_i) \) is a vector-valued obstacle cost function with embedded obstacle location \( l_i \in \mathbb{R}^3 \) and \( \Sigma^\text{obs} = \sigma_{\text{obs}} I \) is a hyperparameter [18].
Given the likelihood in (18), we define the hinge loss 1
\[ c(z, l_i) = \begin{cases} -D_i(z) + \epsilon, & \text{if } D_i(z) \leq \epsilon, \\ 0, & \text{if } D_i(z) > \epsilon, \end{cases} \] (19)
where \( D_i(z) \) is the signed distance field (SDF) about the point \( z \) given obstacle \( l_i \), and provides the distance from point \( z \) in the workspace to the closest obstacle surface, and \( \epsilon \) is a “safety distance” indicating the boundary of the “danger area” near obstacle surfaces, as shown in Figure 3.

For fast collision checking, we adopt sphere representation of \( M \)-body systems [34]. The obstacle cost function for each state \( x_i \) is determined by computing the signed distance for a sphere representing each body, i.e., \( s_j (j = 1, \ldots, M) \) bodies, and collecting them into a single vector such that
\[ h_i(x_i) = \left| c(f(x_i, s_j), l_i) \right|_{1 \leq j \leq M}, \] (20)
where \( f(\cdot) \) maps the state \( x_i \) to the corresponding set of sphere locations in the workspace (more details in [26]).

Figure 3 shows how assignment of obstacle factors differs between reactive and predictive SCATE factor graphs. In Figure 3a, reactive SCATE factor graph assigns future obstacle factors, i.e., the set \( \{ \phi_{i+1}^{\text{obs}}, \ldots, \phi_N^{\text{obs}} \} \), assuming obstacle, \( l_i \), is static in the environment with SDF \( D_i(z) \). In Figure 3b, predictive SCATE factor graphs use \( l_i \) embedded into a sequence of SDFs, i.e., the set \( \{ D_{l_i}, D_{l_i+1}, \ldots, D_{Ml_i} \} \), such that the obstacle factor assignment reflects the anticipated obstacle trajectory. Finally, Algorithm 1 shows that predictive SCATE factor graphs do not need to reassign cost for future obstacles and hence require fewer computational steps than reactive SCATE factor graphs for the same cost.

2) Computational Complexity Analysis: We solve for \( \Theta_i^* \) at time-step \( t_i \) by linearizing (4), given (7) or (8), about a trajectory (starting with guess \( x, \dot{x} \) at \( t_0 \) and then use variable elimination [35] to solve \((n + m)N + iq \) local linear sub-problems [15]. Since finding an optimal elimination order is NP-complete [36], we follow [37], and use Column Approximate Minimum Degree (ColAMD) [38]. When eliminating variables in the estimation portion of SCATE factor graphs, because the number of constraints per pose is constant, the complexity is \( O(1) \) [39]. Elimination of planning state and control variables results in \( O(n^3) \), \( O(m^3) \) complexity per sub-problem, respectively [25], but since the

\[ c'(z) = -0.5 \text{ when } d(z) = \epsilon \] [26].

ColAMD algorithm has complexity \( O(E) \) for a bounded degree graph with \( E \) edges [40], then the total complexity is \( O(i + (N - i) \cdot (\kappa_1 m^3 + \kappa_2 m^2 n)) \) for \( \kappa_1, \kappa_2 > 0 \).

3) Pseudocode for Computer Implementation of SCATE:

Algorithm 1 SCATE Factor Graph Optimization

Generate Initial Plan:
1: \( G = \text{gtsam.NonlinearFactorGraph()} \) \( \triangleright \) Create graph via gtsam [41]
2: \( G.\Phi.\text{add}(\phi^{\text{dyn}}) \) \( \triangleright \) Root graph endpoints with factors in (9) and (10)
3: \( \text{for } i = 0 : N \text{ do } \text{Add constraint factors to graph over } \{t_0, \ldots, t_N\} \)
4: \( \text{if } i < N \text{ then} \)
5: \( G.\Phi.\text{add}(\phi^{\text{dyn}}(x_i, l_i)) \) \( \triangleright \) Dynamics factor in (15)
6: \( G.\Phi.\text{add}(\phi^{l_i}(u_i)) \) \( \triangleright \) Control limit factor in (17)
7: \( \text{if PlanningMode = Reactive then} \)
8: \( G.\Phi.\text{add}(\phi^{\text{obs}}(x_i, \tilde{t})) \) \( \triangleright \) Obstacle factor in (18)
9: \( \text{else if PlanningMode = Predictive then} \)
10: \( G.\Phi.\text{add}(\phi^{\text{obs}}(x_i, \hat{t})) \) \( \triangleright \) Obstacle factor in (18)
11: \( \Theta_0 = \text{gtsam.Values}(\hat{x}, \hat{u}) \) \( \triangleright \) Initialize solution guess given \( \hat{x}, \hat{u} \) at \( t_0 \)
12: \( \Theta_0^k \leftarrow \text{gtsam.LevenbergMarquardt}(\Theta_0, \Theta_0^k).\text{Optimize()} \) \( \triangleright \) Find solution

Iterative Online Factor Graph Optimization:
13: \( \text{for } i = 0 : N \text{ do } \) \( \triangleright \) Iteratively solve factor graph along \( \{t_0, \ldots, t_N\} \)
14: \( G.\Phi.\text{add}(\phi^{\text{meas}}(x_i, l_i)) \) \( \triangleright \) Factors in (11) and (12)
15: \( \Theta^k.\text{add}.Values(\text{meas}) \) \( \triangleright \) Add obstacle location to solution
16: \( \text{if } i = 0 \text{ then} \)
17: \( G.\Phi.\text{remove}(\phi^{\text{start}}(x_i)) \) \( \triangleright \) Remove start factor in (9)
18: \( G.\Phi.\text{remove}(\phi^{\text{goal}}(x_i)) \) \( \triangleright \) Remove goal factor in (10)
19: \( G.\Phi.\text{remove}(\phi^{\text{meas}}(x_i)) \) \( \triangleright \) Remove outdated factors
20: \( G.\Phi.\text{add}(\phi^{\text{obs}}(x_i, \tilde{t})) \) \( \triangleright \) Obscure cost for future obstacle factors
21: \( G.\Phi.\text{replace}(\phi^{\text{obs}}(x_i, t_i), \text{meas}) \) \( \triangleright \) Obstacle factor in (18)
22: \( \Theta^k_{i+1} \leftarrow \text{gtsam.LevenbergMarquardt}(\Theta_i^k, \Theta_{i+1}^k).\text{Optimize()} \) \( \triangleright \) Solve

\( \Theta^e_{i+1} = f(x_{i+1}, z_{i+1}^{\text{meas}}, z_{i+1}^{\text{obs}}, \tilde{t}_{i+1}^{\text{meas}}) \) denotes measured obstacle location.

III. EXPERIMENTAL VALIDATION

We have verified SCATE for onboard spacecraft navigation in the Dynamics and Control Systems Laboratory’s (DCSL) friction-less robotic spacecraft simulator experimental facility, shown in Figure 4. A mounted overhead network of 12 VICON™ cameras monitor a 4m x 4m flat epoxy floor arena shown in Figure 4b, providing localization necessary for inertial pose estimation of all robots within the facility with sub-millimeter and sub-degree accuracy post calibration [32].

Figure 4b also shows the 5-DOF air-bearing robotic platform, ASTROS (Autonomous Spacecraft Testing of Robotic Operations in Space) [31, 32], which is equipped with a three-axis inertial measurement unit, a three-axis rate gyro, and 12 pressurized air thrusters, arranged in a 3-3-3-3
configuration, which impart changes in linear and angular momentum necessary for trajectory tracking.

A. Planar Mechanics and SCATE MAP Trajectory Tracking

For this analysis and experiment, we assume that the ASTROS platform is confined to 3-DOF planar air-bearing motion such that the state is \( x = [x_V \  \dot{x}_V \ \dot{y}_V \ \dot{\psi}_{V/S} \ \dot{\psi}_{V/S} \] \( \in \mathbb{R}^6 \), where \( x_V, y_V \) are the planar components of displacement vector \( r_{V/O} = [x_V \ \ y_V \] \) and \( \psi_{V/S} \) is the heading angle between \( v_1 \) of the inertial frame \( V = \{O, v_1, v_2\} \) and \( s_1 \) of body-fixed frame \( S = \{A, s_1, s_2\} \), as shown in Figure 5. The translational dynamics are

\[
m \dot{r}_{A/O} = \Lambda f_V, \tag{21}\]

where \( m \) is the mass, and \( \Lambda f_V = [\Lambda f^x_V \ \Lambda f^y_V \ \Lambda f^z_V \] \( \in \mathbb{R}^{2 \times 3} \) is the applied planar force and the single-DOF rotational attitude dynamics are given by

\[
\Lambda \dot{\psi}_{V/S} = \tau_z, \tag{22}\]

where \( \Lambda I^z_{zz} \) is the moment of inertia and \( \Lambda \tau_z \) is the applied torque about the body-fixed z-axis. The system can be written in a state-space form (13) where the control input is \( u = [A f^x_V, A f^y_V, \dot{\psi}_{V/S}] \).

We track the SCATE factor graph MAP solution \( \Theta^* = \{\hat{x} \cup \hat{x}, \hat{u} \cup \hat{u}, \hat{l}\} \) to either (7) or (8), during online implementation by letting \( u \) be

\[
u = \hat{u} - K(x - \hat{x}), \tag{23}\]

and designing a feedback matrix \( K \in \mathbb{R}^{3 \times 6} \) such that the matrix \( A - BK \) is Hurwitz.

B. Implementing Online Factor Graph Planning

The proposed online factor graph planning is integrated in these experiments by using by two computers: one as the control computer, which (re)solves the factor graph given the available information, and an onboard real-time Simulink Speedgoat™ computer, which executes the current plan. UDP packets of the set \( \{\hat{x}, \hat{u}\} \) from the SCATE MAP \( \Theta^* = \{\hat{x} \cup \hat{x}, \hat{u} \cup \hat{u}, \hat{l}\} \) are communicated to the onboard computer. The onboard computer simultaneously solves a linear program [42] to allocate the onboard thrusters to execute \( u \) in (23) such that \( u = \hat{u} - K(\hat{x}_{EKF} - \hat{x}) \), where \( \hat{x}_{EKF} \) is an extended Kalman Filter (EKF) estimate of ASTROS’ state [43], and sends UDP packets of \( \hat{x}_{EKF}, \) and measurements of the obstacle, \( z_{\text{meas}} \). Note that \( \hat{x}_{EKF} \) is computed at 100 Hz, whereas \( \Theta^* \) is computed at 3 Hz, hence \( \hat{x}_{EKF} \) is used for feedback and is treated as a state measurement during optimization.

IV. EXPERIMENTAL RESULTS

The viability of reactive SCATE factor graph navigation is evaluated experimentally in an environment with an unknown obstacle which is: a) static and b) moving, while predictive SCATE navigation is tested both on hardware and in software via a to-scale-CAD mesh rendering of the DCSL spacecraft testing facility, ASTROS, and a new 3-DOF air-bearing robot. The objective is for the ASTROS platform to navigate collision-free from rest at one corner of the workspace to a fixed position and attitude in the diagonal corner.

A. Reactive SCATE Navigation With A Static Obstacle

Reactive SCATE navigation is tested with a static obstacle (i.e., the VICONT™ environment), as shown in Figure 6. The system EKF state estimate, \( \hat{x}_{EKF}, \) is given in Figure 7, where along with Figure 6, we see that the ASTROS platform navigates collision-free along the MAP reference trajectory, \( \hat{x} \), reaching the terminal goal state.

B. Reactive SCATE Navigation With A Moving Obstacle

We now move the obstacle around the workspace, forcing the ASTROS platform to navigate to the perimeter of the arena in order to avoid an obstacle collision while navigating to the terminal reference waypoint, as shown in Figure 8. Note that the large instantaneous changes in the MAP reference planar velocities during the 45-70 seconds is the

---

2Results available online at https://youtu.be/C03TKAtqm8I.
Fig. 8: Reactive SCATE navigation with a moving obstacle.

(a) Time: $t = 0$ s.
(b) Time: $t = 40$ s.
(c) Time: $t = 80$ s.

Fig. 9: State estimate, $\hat{x}_{EKF}$, and MAP reference, $\hat{x}$, during reactive SCATE navigation around a moving obstacle.

result of close obstacle proximity, as shown in Figure 9. After 80 seconds, Figure 9 shows that the deviation from the MAP reference trajectory increases, which is a result of the exhausting the fuel necessary for proper thruster allocation.

C. Predictive SCATE Navigation With A Moving Obstacle

1) Hardware Validation: For predictive SCATE implementation, given in (8), we attached the obstacle to a 7-DOF UR10e™ manipulator such that the obstacle follows a predefined trajectory, i.e., $\hat{l}$. The ASTROS platform navigates collision-free around the obstacle, as shown in Figure 10, and reaches the terminal goal state with small deviations from the MAP reference trajectory, as shown in Figure 11.

Fig. 10: Predictive SCATE navigation with a moving obstacle.

(a) Time: $t = 0$ s.
(b) Time: $t = 40$ s.
(c) Time: $t = 60$ s.

Fig. 11: State estimate, $\hat{x}_{EKF}$, and MAP reference, $\hat{x}$, during predictive SCATE navigation around a moving obstacle.

2) Numerical Simulation: Predictive SCATE factor graph navigation is also simulated with the same navigation constraints as the reactive SCATE case, with another robot, as shown in Figure 12. The ASTROS platform successfully navigates without collisions around the obstacles to reach the terminal state, as shown in Figures 12 and 13a. Finally, Figure 13 shows that the MAP estimates $\hat{x}$ and $\hat{l}$, are in close correspondence with $x$, $l$, respectively.

Fig. 12: Predictive SCATE navigation. Faded figures show the past.

(a) Platform state, $x$, MAP estimate, $\hat{x}$, and reference, $x_{ref}(t_N)$.
(b) MAP obstacle trajectory estimation error, $e_l = l - \hat{l}$.

Fig. 13: Predictive SCATE simulation trajectories and estimations.

Fig. 13 shows that the MAP estimates $\hat{x}$ and $\hat{l}$, are in close correspondence with $x$, $l$, respectively.

V. CONCLUSION

We show the utility of using factor graphs for simultaneous control and trajectory estimation (SCATE) for collision-free navigation of autonomous robotic spacecraft systems in environments with moving objects with and without an obstacle motion model. We have numerically and experimentally validated SCATE factor graph navigation, confirming the algorithm’s capacity to: a) incorporate realistic vehicle dynamics, b) generate online collision-free reference trajectories and estimates of vehicle state, control, and obstacle trajectories in a moving obstacle environment, and c) navigate spacecraft systems autonomously in support of future OSS missions.

Although the utility of this algorithm has only been demonstrated for linear planar mechanics, SCATE factor graph navigation is readily extendable to 6-DOF nonlinear single- and multi-body dynamics, as well as, 3D motion planning, where 3D SDFs are necessary. This investigation is part of ongoing and future work.
