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

Matthew King-Smith[1], Panagiotis Tsiotras[2], Frank Dellaert[3]

***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.

*This work was supported by a contract from the Aerospace Corporation.

[1]M. King-Smith is a Robotics PhD Candidate in the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA, mcks3@gatech.edu

[2]P. Tsiotras is the David & Andrew Lewis Chair and Professor with the School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA, tsiotras@gatech.edu

[3]F. Dellaert is a Professor with the School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, 30332, USA, frank.dellaert@cc.gatech.edu

A. Trajectory Optimization as Probabilistic Inference

We represent a trajectory as a continuous-valued function that maps time t to robot states $\boldsymbol{x}(t)$ so as to determine the *maximum a posteriori* (MAP) continuous-time trajectory of $\boldsymbol{x}(t)$ 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) $\boldsymbol{x}(t) \sim GP(\boldsymbol{\mu}(t), \mathbf{K}(t,t'))$, where $\boldsymbol{\mu}(t)$ is the vector-valued mean function and $\mathbf{K}(t,t')$ is a matrix-valued covariance function. For any collection of times $\boldsymbol{t} = \{t_0, \dots, t_N\}$, $\boldsymbol{x}$ has a joint Gaussian distribution $\boldsymbol{x} \triangleq [\boldsymbol{x}_0 \cdots \boldsymbol{x}_N]^\mathsf{T} \sim N(\boldsymbol{\mu}, \mathbf{K})$, with mean vector $\boldsymbol{\mu}$ and covariance kernel $\mathbf{K}$ defined as

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

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

$$p(\boldsymbol{x}) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{\mu}\|^2_{\mathbf{K}}\right\}. \tag{2}$$

Information known *a priori* is then encoded with such priors.

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

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

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

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 $p(\boldsymbol{x}|\boldsymbol{z}) \propto p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})$. Then, we can compute the MAP of the set $\Theta \triangleq \{\boldsymbol{x}\}$ as $\Theta^* = \arg\max_\Theta\{p(\boldsymbol{x}|\boldsymbol{z})\} = \arg\max_\Theta\{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x}))\} = \arg\min_\Theta\{-\log(p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})))\}$ or

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

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 $\Phi \triangleq \{\phi_0, \dots, \phi_V\}$, variable nodes given by the set $\Theta \triangleq \{\boldsymbol{x}\}$, and edges E which connect the two types of nodes, as shown in Figure 1.

Letting Θ_i be a variable subset of Θ, then the posterior distribution can be expressed as the product of the factors

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

Factor graphs that are sparse lend to sparse precision matrices, which are exploitable, yielding a computationally efficient way to determine Θ^* 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, $\boldsymbol{x}$, is represented by the GP prior given in (2), with mean and covariance functions $\boldsymbol{\mu}, \mathbf{K}$, 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

$$p(\hat{\boldsymbol{x}} \cup \check{\boldsymbol{x}}|\boldsymbol{z}) = \phi^{\text{gp}}\phi^{\text{meas}}\phi^{\text{obs}}\phi^{\text{fix}}, \tag{6}$$

where $\phi^{\text{gp}}, \phi^{\text{meas}}, \phi^{\text{obs}}, \phi^{\text{fix}}$, 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.

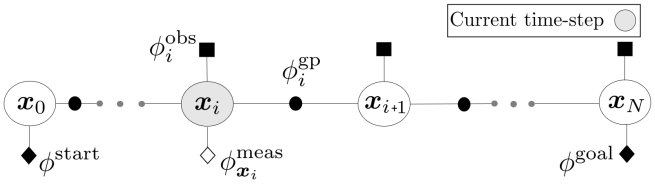


Fig. 1: STEAP factor graph for trajectory estimation and planning.

The MAP solution to the factor graph formulation given in (6), i.e., $\Theta^* = \{\hat{\boldsymbol{x}} \cup \check{\boldsymbol{x}}\}$, solves both the estimation (i.e., $\hat{\boldsymbol{x}}$) and planning (i.e., $\check{\boldsymbol{x}}$) 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 $\phi^{\text{gp}} = \prod_i \phi_i^{\text{gp}}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$, 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 $\Theta \triangleq \{\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{l}\}$, where $\boldsymbol{u}$ is the control input trajectory, and $\boldsymbol{l}$ is the set of obstacle-$\boldsymbol{l}$ 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 $\Theta^* \triangleq \{\hat{\boldsymbol{x}} \cup \check{\boldsymbol{x}}, \hat{\boldsymbol{u}} \cup \check{\boldsymbol{u}}, \hat{\boldsymbol{l}}\}$, where $\hat{\boldsymbol{u}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{l}}$ are estimates of $\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{l}$, and $\check{\boldsymbol{x}}, \check{\boldsymbol{u}}$ are the planned state and control input, respectively, to execute along the remaining trajectories.

Reactive SCATE factor graphs are given by the posterior distribution

$$p(\hat{\boldsymbol{x}} \cup \check{\boldsymbol{x}}, \hat{\boldsymbol{u}} \cup \check{\boldsymbol{u}}, \hat{\boldsymbol{l}} | \boldsymbol{z}) = \phi^{\text{dyn}} \phi^{\text{meas}} \phi^{\text{lim}} \phi^{\text{obs}} \phi^{\text{fix}}, \quad (7)$$

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 $\boldsymbol{x}_i$ assuming obstacles observed are static in the environment.

Predictive SCATE factor graphs, are given the obstacle trajectory, $\tilde{\boldsymbol{l}}$, and are expressed by the posterior distribution

$$p(\hat{\boldsymbol{x}} \cup \check{\boldsymbol{x}}, \hat{\boldsymbol{u}} \cup \check{\boldsymbol{u}}, \hat{\boldsymbol{l}} | \boldsymbol{z}, \tilde{\boldsymbol{l}}) = \phi^{\text{dyn}} \phi^{\text{meas}} \phi^{\text{lim}} \phi^{\text{obs}} \phi^{\text{fix}}. \quad (8)$$

Knowing $\tilde{\boldsymbol{l}}$ in advance enables predictive SCATE to allocate cost to future obstacle locations via the remaining obstacle factors, i.e., the set $\{\phi^{\text{obs}}_{i+1}, \dots, \phi^{\text{obs}}_{N}\}$, during planning for $\boldsymbol{x}_i$.

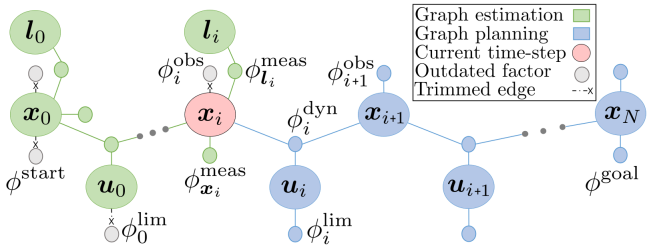


Fig. 2: SCATE factor graph for control and trajectory estimation.

Figure 2 also shows the removal of outdated planning factors, i.e., the set $\{\phi^{\text{start}}, \phi^{\text{lim}}_0, \dots, \phi^{\text{obs}}_i\}$, 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 Guassian factors

$$\phi^{\text{start}}(\boldsymbol{x}_0) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}_{\text{start}}\|^2_{\boldsymbol{\Sigma}_{\text{fix}}}\right\}, \quad (9)$$

$$\phi^{\text{goal}}(\boldsymbol{x}_N) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{x}_N - \boldsymbol{x}_{\text{goal}}\|^2_{\boldsymbol{\Sigma}_{\text{fix}}}\right\}, \quad (10)$$

with the mean as the start or goal and a small covariance $\boldsymbol{\Sigma}_{\text{fix}}$ define $\phi^{\text{fix}} = \phi^{\text{start}}(\boldsymbol{x}_0)\phi^{\text{goal}}(\boldsymbol{x}_N)$ and root the trajectory endpoints to the start and goal locations.

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

$$\phi^{\text{meas}}_{\boldsymbol{x}_i}(\boldsymbol{x}_i) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{z}^{\text{meas}}_{\boldsymbol{x}_i}\|^2_{\boldsymbol{\Sigma}^{\boldsymbol{x}}_{\text{meas}}}\right\}, \quad (11)$$

where, with a slight abuse of notation, $\boldsymbol{z}^{\text{meas}}_{\boldsymbol{x}_i}$ denotes the i-th state measurement with covariance $\boldsymbol{\Sigma}^{\boldsymbol{x}}_{\text{meas}}$. Assuming an obstacle is observed via relative bearing and range measurements, $\boldsymbol{z}^{\text{meas}}_{\theta_i}, \boldsymbol{z}^{\text{meas}}_{r_i}$, with covariance matrices $\boldsymbol{\Sigma}^{\theta}_{\text{meas}}, \boldsymbol{\Sigma}^{r}_{\text{meas}}$, respectively, we can then define a bearing and range factors as

$$\phi^{\text{meas}}_{\theta_i}(\boldsymbol{x}_i, \boldsymbol{l}_i) \propto \exp\left\{-\frac{1}{2}\|h_{\theta_i}(\boldsymbol{x}_i, \boldsymbol{l}_i) - \boldsymbol{z}^{\text{meas}}_{\theta_i}\|^2_{\boldsymbol{\Sigma}^{\theta}_{\text{meas}}}\right\},$$
$$\phi^{\text{meas}}_{r_i}(\boldsymbol{x}_i, \boldsymbol{l}_i) \propto \exp\left\{-\frac{1}{2}\|h_{r_i}(\boldsymbol{x}_i, \boldsymbol{l}_i) - \boldsymbol{z}^{\text{meas}}_{r_i}\|^2_{\boldsymbol{\Sigma}^{r}_{\text{meas}}}\right\}, \quad (12)$$

where $h_{\theta_i}(\boldsymbol{x}_i, \boldsymbol{l}_i) = \text{atan2}(\boldsymbol{l}_{i,y} - \boldsymbol{x}_{i,y}, \boldsymbol{l}_{i,x} - \boldsymbol{x}_{i,x})$, and $h_{r_i}(\boldsymbol{x}_i, \boldsymbol{l}_i) = \|\boldsymbol{r}^{\mathcal{S}}_{\text{B}/A}\|_2$ is the relative range given the planar displacement vector between the robot and obstacle, $\boldsymbol{r}^{\mathcal{S}}_{\text{B}/A} = \begin{bmatrix} \text{c}(\boldsymbol{x}_{i,\psi^{\mathcal{V}/\mathcal{S}}}) & -\text{s}(\boldsymbol{x}_{i,\psi^{\mathcal{V}/\mathcal{S}}}) \\ \text{s}(\boldsymbol{x}_{i,\psi^{\mathcal{V}/\mathcal{S}}}) & \text{c}(\boldsymbol{x}_{i,\psi^{\mathcal{V}/\mathcal{S}}}) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{l}_{i,x} - \boldsymbol{x}_{i,x} \\ \boldsymbol{l}_{i,y} - \boldsymbol{x}_{i,y} \end{bmatrix}$, where $\text{c}(\cdot) = \cos(\cdot)$, and $\text{s}(\cdot) = \sin(\cdot)$, as shown in Figure 5. A 2D bearing range factor is then given by $\phi^{\text{meas}}_{\boldsymbol{l}_i} = \phi^{\text{meas}}_{\theta_i}\phi^{\text{meas}}_{r_i}$.

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.,

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t), \quad (13)$$

which, after discretization with $\delta t = t_{i+1} - t_i$, yields

$$\boldsymbol{x}_{i+1} = F_x \boldsymbol{x}_i + F_u \boldsymbol{u}_i, \quad (14)$$

where $F_x = e^{A\delta t}$ and $F_u = e^{A\delta t}\int_{t_i}^{t_{i+1}} e^{-A\tau}\text{d}\tau B$. Given (14), we define the discrete LTI dynamics factor as

$$\phi^{\text{dyn}}_i(\boldsymbol{x}_{i+1}, \boldsymbol{x}_i, \boldsymbol{u}_i) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{x}_{i+1} - F_x\boldsymbol{x}_i - F_u\boldsymbol{u}_i\|^2_{\boldsymbol{\Sigma}_{\text{dyn}}}\right\}, \quad (15)$$

where $\boldsymbol{\Sigma}_{\text{dyn}}$ is the covariance matrix of the factor.

Control limit factor: The control limit factor is designed to ensure that control trajectories respect limits. Defining the hinge loss cost function for the $\boldsymbol{u}_i$ control input

$$\boldsymbol{h}(\boldsymbol{u}_i) = \left[\begin{cases} u^j_{i,-} + u^j_{i,ths} - u^j_i & \text{if } u^j_i < u^j_{i,-} + u^j_{i,ths} \\ 0 & \text{if } u^j_i \leq u^j_{i,+} - u^j_{i,ths} \\ u^j_i - u^j_{i,+} + u^j_{i,ths} & \text{otherwise} \end{cases}\right]\Bigg|_{1 \leq j \leq m}, \quad (16)$$

where $u^j_{i,-}, u^j_{i,+}$ are the lower and upper limit of the u^j_i component of $\boldsymbol{u}_i$, respectively, and $u^j_{i,ths}$ is a threshold value, then given (16), the control limit factor is given as

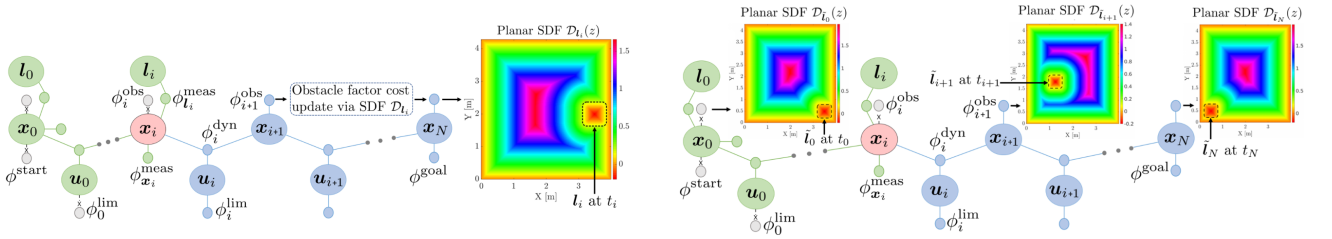
$$\phi^{\text{lim}}_i(\boldsymbol{u}_i) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{h}(\boldsymbol{u}_i)\|^2_{\boldsymbol{\Sigma}_{\text{lim}}}\right\}, \quad (17)$$

where $\boldsymbol{\Sigma}_{\text{lim}}$ is the covariance matrix of the factor.

Obstacle factor: The obstacle factor derives from the likelihood function, which is given by the conditional distribution $l_{\text{obs}}(\boldsymbol{x}_i; \boldsymbol{z}_i = 0) = p(\boldsymbol{z}_i = 0 | \boldsymbol{x}_i)$, which specifies the probability of being clear of collisions, or the probability that collision event $\boldsymbol{z}_i = 0$, given the current configuration $\boldsymbol{x}_i$. This likelihood is represented as a distribution in the exponential family

$$\phi^{\text{obs}}_i(\boldsymbol{x}_i) = l_{\text{obs}}(\boldsymbol{x}_i; \boldsymbol{z}_i = 0) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{h}_{\boldsymbol{l}_i}(\boldsymbol{x}_i)\|^2_{\boldsymbol{\Sigma}_{\text{obs}}}\right\}, \quad (18)$$

where $\boldsymbol{h}_{\boldsymbol{l}_i}(\boldsymbol{x}_i)$ is a vector-valued *obstacle cost function* with embedded obstacle location $\boldsymbol{l}_i \in \mathbb{R}^q$ and $\boldsymbol{\Sigma}^{-1}_{\text{obs}} = \sigma_{\text{obs}}\boldsymbol{I}$ is a hyperparameter [18].



(a) Reactive: continuous factor cost (re)assignment using $\mathcal{D}_{\boldsymbol{l}_i}(z)$.

(b) Predictive: preemptive factor cost assignment using $\{\mathcal{D}_{\tilde{\boldsymbol{l}}_0}, \ldots, \mathcal{D}_{\tilde{\boldsymbol{l}}_N}\}$.

Fig. 3: SCATE obstacle factor cost assignment. The color bar indicates the value of SDF $\mathcal{D}_{\boldsymbol{l}_i}$ at a point z.

Given the likelihood in (18), we define the hinge loss [1]

$$\boldsymbol{c}(z, \boldsymbol{l}_i) = \begin{cases} -\mathcal{D}_{\boldsymbol{l}_i}(z) + \epsilon, & \text{if } \mathcal{D}_{\boldsymbol{l}_i}(z) \leq \epsilon, \\ 0, & \text{if } \mathcal{D}_{\boldsymbol{l}_i}(z) > \epsilon, \end{cases} \tag{19}$$

where $\mathcal{D}_{\boldsymbol{l}_i}(z)$ is the *signed distance field* (SDF) about the point z given obstacle-$\boldsymbol{l}_i$, and provides the distance from point z in the workspace to the closest obstacle surface, and ϵ 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 $\boldsymbol{x}_i$ 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

$$\boldsymbol{h}_{\boldsymbol{l}_i}(\boldsymbol{x}_i) = [\boldsymbol{c}(\boldsymbol{f}(\boldsymbol{x}_i, s_j), \boldsymbol{l}_i)] \Big|_{1 \leq j \leq M}, \tag{20}$$

where $\boldsymbol{f}(\cdot)$ maps the state $\boldsymbol{x}_i$ 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 $\{\phi^{\text{obs}}_{i+1}, \ldots, \phi^{\text{obs}}_N\}$, assuming obstacle, $\boldsymbol{l}_i$, is static in the environment with SDF $\mathcal{D}_{\boldsymbol{l}_i}(z)$. In Figure 3b, predictive SCATE factor graphs use $\tilde{\boldsymbol{l}}$ embedded into a sequence of SDFs, i.e., the set $\{\mathcal{D}_{\tilde{\boldsymbol{l}}_0}, \mathcal{D}_{\tilde{\boldsymbol{l}}_1}, \ldots, \mathcal{D}_{\tilde{\boldsymbol{l}}_N}\}$, 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 problem.

2) Computational Complexity Analysis: We solve for Θ^*_i at time-step t_i by linearizing (4), given (7) or (8), about a trajectory (starting with guess $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}$ 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 ColAMD algorithm has complexity $O(E)$ for a bounded degree graph with E edges [40], then the total complexity is $O(i + (N-i) \cdot (\kappa_1 n^3 + \kappa_2 m^3))$ for $\kappa_1, \kappa_2 > 0$.

3) Pseudocode for Computer Implementation of SCATE:

Algorithm 1 SCATE Factor Graph Optimization

Generate Initial Plan:

G = gtsam.NonlinearFactorGraph() ▷ Create graph via gtsam [41]
$G.\Phi$.add $\left(\phi^{\text{fix}}\right)$ ▷ Root graph endpoints with factors in (9) and (10)
for $i = 0 : N$ **do** ▷ Add constraint factors to graph over $\{t_0, \ldots, t_N\}$
  if $i < N$ **then**
    $G.\Phi$.add $\left(\phi^{\text{dyn}}_i (\boldsymbol{x}_{i+1}, \boldsymbol{x}_i, \boldsymbol{u}_i)\right)$ ▷ Dynamics factor in (15)
    $G.\Phi$.add $\left(\phi^{\text{lim}}_i (\boldsymbol{u}_i)\right)$ ▷ Control limit factor in (17)
  if $PlanningMode$ = Reactive **then** ▷ Plan without $\tilde{\boldsymbol{l}}$
    $G.\Phi$.add $\left(\phi^{\text{obs}}_i (\boldsymbol{x}_i), \emptyset\right)$[a] ▷ Obstacle factor in (18)
  else if $PlanningMode$ = Predictive **then** ▷ Plan given $\tilde{\boldsymbol{l}}$
    $G.\Phi$.add $\left(\phi^{\text{obs}}_i (\boldsymbol{x}_i), \tilde{\boldsymbol{l}}_i\right)$ ▷ Obstacle factor in (18)
Θ_0 = gtsam.Values($\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}$) ▷ Initialize solution guess given $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}$ at t_0
$\Theta^*_0 \leftarrow$ gtsam.LevenbergMarquardt(G, Θ_0).Optimize() ▷ Find solution

Iterative Online Factor Graph Optimization:

for $i = 0 : N$ **do** ▷ Iteratively solve factor graph along $\{t_0, \ldots, t_N\}$
  $G.\Phi$.add $\left(\phi^{\text{meas}}_{\boldsymbol{x}_i}(\boldsymbol{x}_i), \phi^{\text{meas}}_{\boldsymbol{l}_i}(\boldsymbol{x}_i, \boldsymbol{l}_i)\right)$ ▷ Factors in (11) and (12)
  Θ^*_i.add.Values($\boldsymbol{l}^{\text{meas}\,b}_i$) ▷ Add obstacle location to solution
  if $i = 0$ **then**
    $G.\Phi$.remove $\left(\phi^{\text{start}}(\boldsymbol{x}_0)\right)$ ▷ Remove start factor in (9)
  else if $i = N$ **then**
    $G.\Phi$.remove $\left(\phi^{\text{goal}}(\boldsymbol{x}_N)\right)$ ▷ Remove goal factor in (10)
  $G.\Phi$.remove $\left(\phi^{\text{lim}}_i(\boldsymbol{u}_i), \phi^{\text{obs}}_i(\boldsymbol{x}_i)\right)$ ▷ Remove outdated factors
  if $PlanningMode$ = Reactive **then** ▷ Plan given $\boldsymbol{l}^{\text{meas}}_i$
    for $k = i+1 : N$ **do** ▷ Assign cost for future obstacle factors
      $G.\Phi$.replace $\left(\phi^{\text{obs}}_k (\boldsymbol{x}_k), \boldsymbol{l}^{\text{meas}}_i\right)$ ▷ Obstacle factor in (18)
  $\Theta^*_{i+1} \leftarrow$ gtsam.LevenbergMarquardt(G, Θ^*_i).Optimize() ▷ Solve

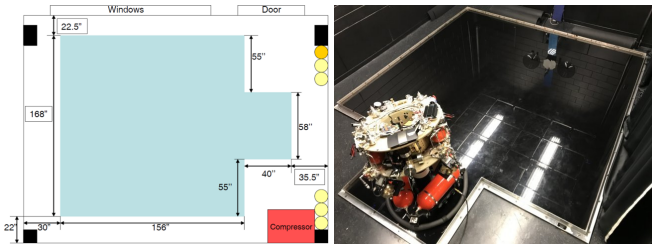
[a] Obstacle factors encode an empty bounded workspace when $\boldsymbol{l}_i = \emptyset$.
[b] $\boldsymbol{l}^{\text{meas}}_i = \boldsymbol{f}(\boldsymbol{z}^{\text{meas}}_{\boldsymbol{x}_i}, \boldsymbol{z}^{\text{meas}}_{\theta_i}, \boldsymbol{z}^{\text{meas}}_{r_i})$ denotes measured obstacle location.

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

[1] The hinge loss is not differentiable at $d(z) = \epsilon$, so in this implementation $c'(z) = -0.5$ when $d(z) = \epsilon$ [26].



(a) Dimensions of the testing arena. (b) ASTROS platform in arena bay.

Fig. 4: The DCSL's robotic air-bearing spacecraft testing facility.

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 $\boldsymbol{x} = \begin{bmatrix} x^{\mathcal{V}} & \dot{x}^{\mathcal{V}} & y^{\mathcal{V}} & \dot{y}^{\mathcal{V}} & \psi^{\mathcal{V}/\mathcal{S}} & \dot{\psi}^{\mathcal{V}/\mathcal{S}} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^6$, where $x^{\mathcal{V}}, y^{\mathcal{V}}$ are the planar components of displacement vector $\boldsymbol{r}^{\mathcal{V}}_{\mathrm{A/O}} = [x^{\mathcal{V}}, y^{\mathcal{V}}]^{\mathsf{T}}$, and $\psi^{\mathcal{V}/\mathcal{S}}$ is the heading angle between $\boldsymbol{v}_1$ of the inertial frame $\mathcal{V} = \{\mathrm{O}, \boldsymbol{v}_1, \boldsymbol{v}_2\}$ and $\boldsymbol{s}_1$ of body-fixed frame $\mathcal{S} = \{\mathrm{A}, \boldsymbol{s}_1, \boldsymbol{s}_2\}$, as shown in Figure 5. The translational dynamics are

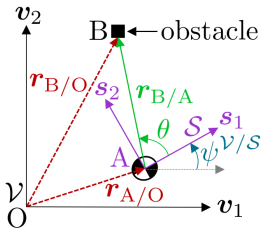


Fig. 5: Heading angle, $\psi^{\mathcal{V}/\mathcal{S}}$, bearing angle, θ, frames, and center of mass definitions.

$$m\ddot{\boldsymbol{r}}^{\mathcal{V}}_{\mathrm{A/O}} = {}^{\mathrm{A}}\boldsymbol{f}^{\mathcal{V}}, \tag{21}$$

where m is the mass, and ${}^{\mathrm{A}}\boldsymbol{f}^{\mathcal{V}} = [{}^{\mathrm{A}}f^{\mathcal{V}}_x, {}^{\mathrm{A}}f^{\mathcal{V}}_y]^{\mathsf{T}}$ is the applied planar force and the single-DOF rotational attitude dynamics are given by

$${}^{\mathrm{A}}I^{\mathcal{V}}_{zz}\ddot{\psi}^{\mathcal{V}/\mathcal{S}} = {}^{\mathrm{A}}\tau^{\mathcal{V}}_z, \tag{22}$$

where ${}^{\mathrm{A}}I^{\mathcal{V}}_{zz}$ is the moment of inertia and ${}^{\mathrm{A}}\tau^{\mathcal{V}}_z$ 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 $\boldsymbol{u} = [{}^{\mathrm{A}}f^{\mathcal{V}}_x, {}^{\mathrm{A}}f^{\mathcal{V}}_y, {}^{\mathrm{A}}\tau^{\mathcal{V}}_z]^{\mathsf{T}}$.

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

$$\boldsymbol{u} = \check{\boldsymbol{u}} - K(\boldsymbol{x} - \check{\boldsymbol{x}}), \tag{23}$$

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

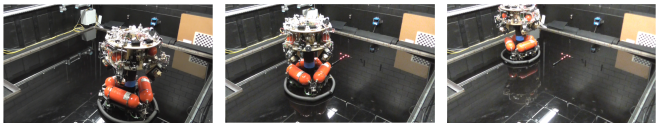
The proposed online factor graph planning is integrated in these experiments by using by two computers: one as the control computer, which (re)solves the factor graph given the available information, and an onboard real-time Simulink Speedgoat™ computer, which executes the current plan. UDP packets of the set $\{\check{\boldsymbol{x}}, \check{\boldsymbol{u}}\}$ from the SCATE MAP $\Theta^* = \{\hat{\boldsymbol{x}} \cup \check{\boldsymbol{x}}, \hat{\boldsymbol{u}} \cup \check{\boldsymbol{u}}, \hat{\boldsymbol{l}}\}$ are communicated to the onboard computer. The onboard computer simultaneously solves a linear program [42] to allocate the onboard thrusters to execute $\boldsymbol{u}$ in (23) such that $\boldsymbol{u} = \check{\boldsymbol{u}} - K(\hat{\boldsymbol{x}}_{\mathrm{EKF}} - \check{\boldsymbol{x}})$, where $\hat{\boldsymbol{x}}_{\mathrm{EKF}}$ is an extended Kalman Filter (EKF) estimate of ASTROS' state [43], and sends UDP packets of $\hat{\boldsymbol{x}}_{\mathrm{EKF}}$, and measurements of the obstacle, $\boldsymbol{z}^{\mathrm{meas}}$. Note that $\hat{\boldsymbol{x}}_{\mathrm{EKF}}$ is computed at 100 Hz, whereas Θ^* is computed at 3 Hz, hence $\hat{\boldsymbol{x}}_{\mathrm{EKF}}$ 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 [2].

A. Reactive SCATE Navigation With A Static Obstacle

Reactive SCATE navigation is tested with a static obstacle (i.e., the VICON™ wand) environment, as shown in Figure 6. The system EKF state estimate, $\hat{\boldsymbol{x}}_{\mathrm{EKF}}$, is given



(a) Time: $t = 0$s. (b) Time: $t = 20$s. (c) Time: $t = 60$s.

Fig. 6: Reactive SCATE navigation in a static obstacle environment.

in Figure 7, where along with Figure 6, we see that the ASTROS platform navigates collision-free along the MAP reference trajectory, $\check{\boldsymbol{x}}$, reaching the terminal goal state.

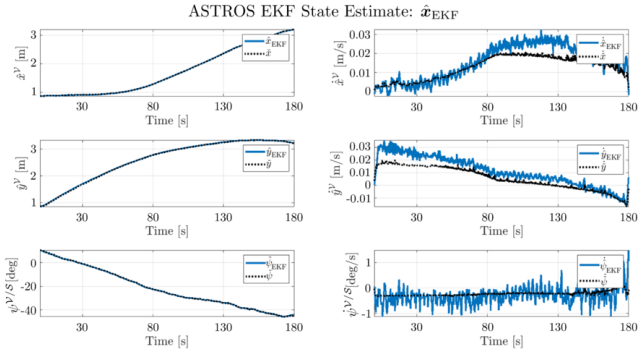


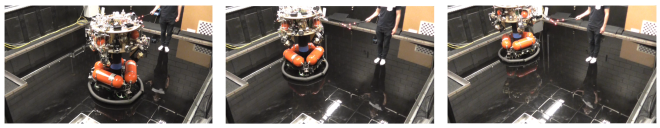
Fig. 7: State estimate, $\hat{\boldsymbol{x}}_{\mathrm{EKF}}$, and MAP reference, $\check{\boldsymbol{x}}$.

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

[2]Results available online at https://youtu.be/C03TKAtqm8I.



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

Fig. 8: Reactive SCATE navigation with a moving obstacle.

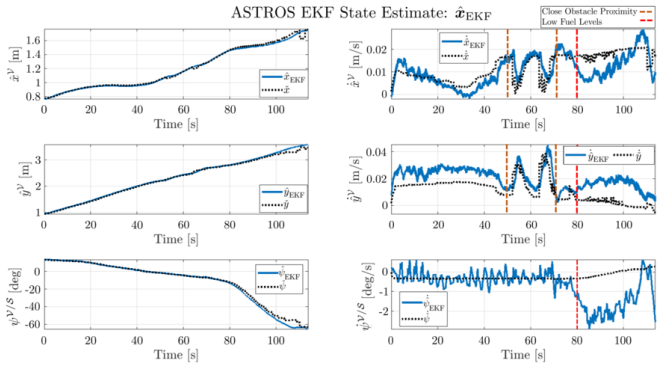
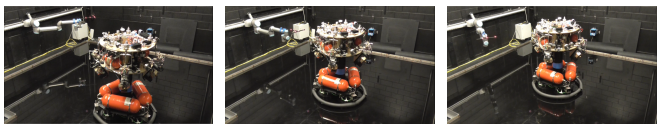


Fig. 9: State estimate, $\hat{\boldsymbol{x}}_{\text{EKF}}$, and MAP reference, $\check{\boldsymbol{x}}$, during reactive SCATE navigation around a moving obstacle.

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$^{\text{TM}}$ manipulator such that the obstacle follows a predefined trajectory, i.e., $\tilde{\boldsymbol{l}}$. The ASTROS platform navigates collision-free around the obstacle, as shown in Figure 10, and



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

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

reaches the terminal goal state with small deviations from the MAP reference trajectory, as shown in Figure 11.

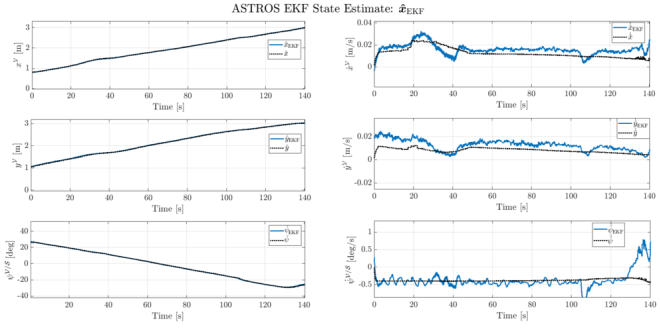


Fig. 11: State estimate, $\hat{\boldsymbol{x}}_{\text{EKF}}$, and MAP reference, $\check{\boldsymbol{x}}$, during predictive SCATE navigation around a moving obstacle.

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

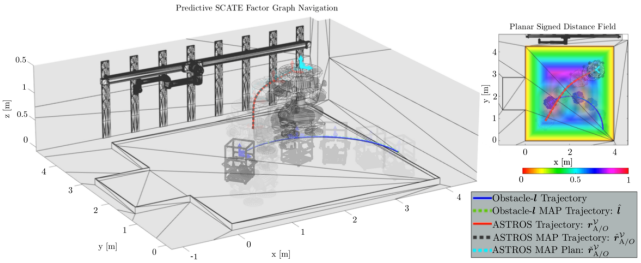
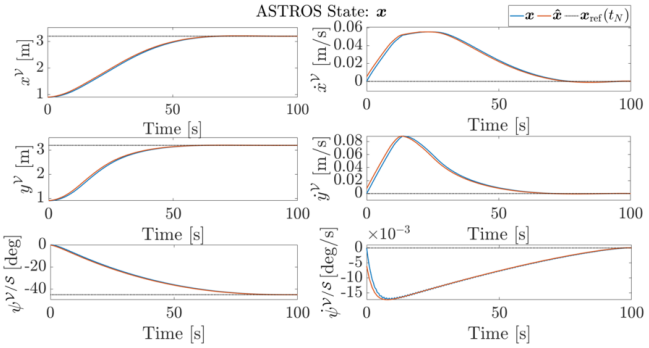
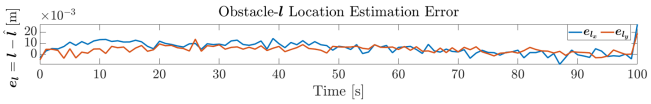


Fig. 12: Predictive SCATE navigation. Faded figures show the past.

navigates without collisions around the obstacles to reach the terminal state, as shown in Figures 12 and 13a. Finally,



(a) Platform state, $\boldsymbol{x}$, MAP estimate, $\hat{\boldsymbol{x}}$, and reference, $\boldsymbol{x}_{\text{ref}}(t_N)$.



(b) MAP obstacle trajectory estimation error, $\boldsymbol{e}_{\boldsymbol{l}} = \boldsymbol{l} - \hat{\boldsymbol{l}}$.

Fig. 13: Predictive SCATE simulation trajectories and estimations.

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

REFERENCES

[1] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich, "A Review of Space Robotics Technologies for On-Orbit Servicing," *Progress in Aerospace Sciences*, vol. 68, pp. 1–26, July 2014. [Online]. Available: https://doi.org/10.1016/j.paerosci.2014.03.002

[2] G. Gefke, A. Janas, R. Chiei, M. Sammons, and B. B. Reed, "Advances in Robotic Servicing Technology Development," in *AIAA SPACE 2015 Conference and Exposition*, Pasadena, CA., 31 Aug. - 2 Sept. 2015. [Online]. Available: https://doi.org/10.2514/6.2015-4426

[3] B. B. Reed, R. C. Smith, B. J. Naasz, J. F. Pellegrino, and C. E. Bacon, "The Restore-L Servicing Mission," in *AIAA SPACE 2016*, Long Beach, CA., 13-16 Sept. 2016. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2016-5478

[4] A. Jonsson, R. A. Morris, and L. Pedersen, "Autonomy in Space: Current Capabilities and Future Challenge," *AI Magazine*, vol. 28, no. 4, p. 27, Dec. 2007. [Online]. Available: https://doi.org/10.1609/aimag.v28i4.2066

[5] M. Tipaldi and L. Glielmo, "A Survey on Model-Based Mission Planning and Execution for Autonomous Spacecraft," *Systems Journal*, vol. 12, no. 4, pp. 3893–3905, Dec. 2018. [Online]. Available: https://doi.org/10.1109/JSYST.2017.2720682

[6] M. A. Vavrina, C. E. Skelton, K. D. DeWeese, B. J. Naasz, D. E. Gaylor, and C. D'Souza, "Safe rendezvous trajectory design for the restore-l mission," in *Spaceflight Mechanics Meeting*, ser. Advances in the Astronautical Sciences, vol. 168, Maui, HI., 13-17 Jan. 2019, pp. 3649–3668. [Online]. Available: http://www.scopus.com/inward/record.url?scp=85070098230&partnerID=8YFLogxK

[7] T. Grant, A. Soler, A. Bos, U. Brauer, M. Neerinex, and M. Wolff, "Space Autonomy as Migration of Functionality: The Mars Case," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology*, Pasadena, CA., 17-20 July 2006. [Online]. Available: https://doi.org/10.1109/SMC-IT.2006.72

[8] J. A. Starek, B. Açıkmeşe, I. A. Nesnas, and M. Pavone, "Spacecraft Autonomy Challenges for Next-Generation Space Missions," in *Advances in Control System Technology for Aerospace Applications*. Springer Berlin Heidelberg, 17 Sep. 2015, pp. 1–48. [Online]. Available: https://doi.org/10.1007/978-3-662-47694-9_1

[9] O. Halbe and M. Hajek, "Online Waypoint Trajectory Generation Using State-Dependent Riccati Equation," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 12, pp. 2687–2693, Dec. 2019. [Online]. Available: https://doi.org/10.2514/1.G003774

[10] D. Malyuta, T. Reynolds, M. Szmuk, B. Acikmese, and M. Mesbahi, "Fast Trajectory Optimization via Successive Convexification for Spacecraft Rendezvous with Integer Constraints," in *AIAA Scitech 2020 Forum*, Orlando, FL., Jan. 2020, pp. 1–24. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-0616

[11] F. Capolupo and P. Labourdette, "Receding-Horizon Trajectory Planning Algorithm for Passively Safe On-Orbit Inspection Missions," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 5, pp. 1023–1032, May 2019. [Online]. Available: https://doi.org/10.2514/1.G003736

[12] M. Maestrini and P. Di Lizia, "Guidance for Autonomous Inspection of Unknown Uncooperative Resident Space Object," in *AAS/AIAA Astrodynamics Specialist Conference*, South Lake Tahoe, CA., Aug. 2020, pp. 1–13. [Online]. Available: https://www.researchgate.net/publication/349318751_Guidance_for_Autonomous_Inspection_of_Unknown_Uncooperative_Resident_Space_Objects

[13] F. Capolupo, T. Simeon, and J.-C. Berges, "Heuristic Guidance Techniques for the Exploration of Small Celestial Bodies," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8279–8284, Jul. 2017, 20th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896317319432

[14] G. Francis, E. Collins, O. Chuy, and A. Sharma, "Sampling-Based Trajectory Generation for Autonomous Spacecraft Rendezvous and Docking," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, Boston, MA., Aug. 2013, p. 4549. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2013-4549

[15] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 1 Dec. 2006. [Online]. Available: https://doi.org/10.1177%2F0278364906072768

[16] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017. [Online]. Available: https://scholar.archive.org/work/vwclhxa4anhsvb7g6dckbhg4ue/access/wayback/http://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf

[17] A.-a. Agha-mohammadi, S. Agarwal, S.-K. Kim, S. Chakravorty, and N. M. Amato, "SLAP: Simultaneous Localization and Planning Under Uncertainty Via Dynamic Replanning in Belief Space," *Transactions on Robotics*, vol. 34, no. 5, pp. 1195–1214, 2 Oct. 2018. [Online]. Available: https://doi.org/10.1109/TRO.2018.2838556

[18] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression," in *Robotics: Science and Systems*, vol. 10, Berkeley, CA, 14 July 2014. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.707.3225&rep=rep1&type=pdf

[19] S. Anderson and T. D. Barfoot, "Full STEAM Ahead: Exactly Sparse Gaussian Process Regression for Batch Continuous-Time Trajectory Estimation on SE(3)," in *International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 28 Sep. - 2 Oct. 2015, pp. 157–164. [Online]. Available: https://doi.org/10.1109/IROS.2015.7353368

[20] S. Anderson, T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch nonlinear continuous-time trajectory estimation as exactly sparse Gaussian process regression," *Autonomous Robots*, vol. 39, no. 3, pp. 221–238, Jul 2015. [Online]. Available: https://doi.org/10.1007/s10514-015-9455-y

[21] H. J. Kappen, V. Gómez, and M. Opper, "Optimal Control as a Graphical Model Inference Problem," *Machine Learning*, vol. 87, no. 2, pp. 159–182, Feb. 2012. [Online]. Available: https://doi.org/10.1007/s10994-012-5278-7

[22] D.-N. Ta, M. Kobilarov, and F. Dellaert, "A Factor Graph Approach to Estimation and Model Predictive Control on Unmanned Aerial Vehicles," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, Orlando, FL., May 2014, pp. 181–188. [Online]. Available: https://doi.org/10.1109/ICUAS.2014.6842254

[23] S. Levine, "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review," *CoRR*, vol. abs/1805.00909, May 2018. [Online]. Available: http://arxiv.org/abs/1805.00909

[24] J. Watson, H. Abdulsamad, and J. Peters, "Stochastic Optimal Control as Approximate Input Inference," in *Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 697–716. [Online]. Available: https://proceedings.mlr.press/v100/watson20a.html

[25] S. Yang, G. Chen, Y. Zhang, F. Dellaert, and H. Choset, "Equality Constrained Linear Optimal Control With Factor Graphs," *CoRR*, vol. abs/2011.01360, Nov. 2020. [Online]. Available: https://arxiv.org/abs/2011.01360

[26] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs," in *Robotics: Science and Systems*, vol. 12, no. 4, Ann Arbor, MI., June 2016. [Online]. Available: https://homes.cs.washington.edu/~bboots/files/GPMP2.pdf

[27] J. Dong, M. Mukadam, B. Boots, and F. Dellaert, "Sparse Gaussian Processes on Matrix Lie Groups: A Unified Framework for Optimizing Continuous-Time Trajectories," in *International Conference on Robotics and Automation*, Brisbane, QLD, Australia, May 21-25 2018, pp. 6497–6504. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8461077

[28] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "Simultaneous Trajectory Estimation and Planning Via Probabilistic Inference," in *Robotics: Science and Systems*, Cambridge, MA., July 2017. [Online]. Available: https://par.nsf.gov/servlets/purl/10046313

[29] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "STEAP: Simultaneous Trajectory Estimation and Planning," *Autonomous Robots*, vol. 43, no. 2, pp. 415–434, Jul. 2018. [Online]. Available: https://doi.org/10.1007/s10514-018-9770-1

[30] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-Time Gaussian Process Motion Planning Via Probabilistic Inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, Sep. 2018. [Online]. Available: https://doi.org/10.1177/0278364918790369

[31] D.-M. Cho, D. Jung, and P. Tsiotras, "A 5-dof Experimental Platform for Spacecraft Rendezvous and Docking," in *AIAA Infotech@Aerospace Conference*, Seattle, WA., Apr. 2009, pp. 1–20. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2009-1869

[32] P. Tsiotras, "ASTROS: A 5DOF Experimental Platform for Research in Space Proximity Operations," in *ASS Guidance and Control Conference*. Breckenridge, CO: Georgia Institute of

Technology, Jan. 31-Feb. 5 2014, paper 14-114. [Online]. Available: http://hdl.handle.net/1853/53259

[33] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001. [Online]. Available: https://doi.org/10.1109/18.910572

[34] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian Optimization for Motion Planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, Aug. 2013. [Online]. Available: https://doi.org/10.1177/0278364913488805

[35] J. Blair and B. Peyton, "An Introduction to Chordal Graphs and Clique Trees," Oak Ridge National Lab, TN, USA, Tech. Rep. ORNL/TM-12203 ON: DE93012508, Nov. 1992. [Online]. Available: https://www.osti.gov/biblio/10145949

[36] M. Yannakakis, "Computing the Minimum Fill-In is NP-Complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 2, no. 1, pp. 77–79, Mar 1981. [Online]. Available: https://doi.org/10.1137/0602010

[37] R. Pradhan, S. Yang, F. Dellaert, H. Choset, and M. J. Travers, "Optimal Control for Structurally Sparse Systems using Graphical Inference," *ArXiv*, vol. abs/2104.02945, April 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:233169157

[38] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng, "A Column Approximate Minimum Degree Ordering Algorithm," *Transactions on Mathematical Software*, vol. 30, no. 3, pp. 353–376, Sept. 2004. [Online]. Available: https://doi.org/10.1145/1024074.1024079

[39] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental Smoothing and Mapping with Fluid Relinearization and Incremental Variable Reordering," in *International Conference on Robotics and Automation*, Shanghai, China, May 2011. [Online]. Available: https://doi.org/10.1109/ICRA.2011.5979641

[40] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An Approximate Minimum Degree Ordering Algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886–905, Oct. 1996. [Online]. Available: https://doi.org/10.1137/S0895479894278952

[41] F. Dellaert, "Factor Graphs and GTSAM: A Hands-On Introduction," Georgia Institute of Technology, Atlanta, GA, USA, Tech. Rep. GT-RIM-CP&R-2012-002, Sept. 2012. [Online]. Available: http://hdl.handle.net/1853/45226

[42] A. Makhorin, "GLPK (GNU Linear Programming Kit)," https://www.gnu.org/software/glpk/, 2011, Moscow: Department for Applied Informatics, Moscow Aviation Institute.

[43] N. Filipe, M. Kontitsis, and P. Tsiotras, "Extended Kalman Filter for Spacecraft Pose Estimation Using Dual Quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 9, pp. 1625–1641, Sep 2015. [Online]. Available: https://doi.org/10.2514/1.G000977