Information-Driven and Risk-Bounded Autonomy for Scientist Avatars

Eric Timmons,[1] Marlyse Reeves,[2] Benjamin Ayton, [3] and Brian C. Williams[4]
MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St, Cambridge, MA 02139, USA

Michel D. Ingham,[5] Julie Castillo-Rogez,[6] William Seto,[7] Klaus Havelund,[8] Ashkan Jasour,[9] Benjamin Donitz,[10] Declan Mages,[11] Amir Rahmani,[12] Peyman Tavallali,[13] and Seung Chung[14]
NASA Jet Propulsion Laboratory, 4800 Oak Grove Dr, Pasadena, CA 91109, USA

We present an overview of an information-seeking risk-bounded planning and execution system to enable future spacecraft to make decisions and adapt their behavior to seek out the most high-value scientific information, while bounding risk of failure. Information-seeking autonomy is an innovation with potential for transformational impact on the way our spacecraft enable scientific discoveries, by complementing traditional scientist-in-the-loop operations with appropriately-conservative onboard direction of science measurement activities (based on scientist-specified models). Our executive selects a measurement strategy that maximizes information based on the environment and initial measurements. It takes various mission risks into account when planning and executing activities to deal with uncertainties and disturbances. It can adapt its strategy based on collected measurements on-the-fly. Finally, it demonstrates resilience, despite failures and degradations, to achieve its high-level scientific objectives. We have performed an initial demonstration of the capabilities of our executive against a basic spacecraft simulation of an asteroid flyby scenario, leveraging the Robot Operating System and the Basilisk open-source simulation framework. It has also been demonstrated on autonomous underwater vehicle scenarios.

[1] Student, Department of Electrical Engineering and Computer Science and MIT-WHOI Joint Program.
[2] Student, Department of Electrical Engineering and Computer Science.
[3] Student, Department of Aeronautics and Astronautics and MIT-WHOI Joint Program.
[4] Professor, Department of Aeronautics and Astronautics.
[5] Chief Technologist, Systems Engineering Division. AIAA Associate Fellow.
[6] Planetary Scientist, Science Division.
[7] Technologist, Maritime and Multi-Agent Autonomy Group.
[8] Senior Research Scientist, Advanced Flight Software Group.
[9] Robotics Technologist, Robotic Aerial Mobility Group.
[10] Systems Engineer, Advanced Design Engineering Group.
[11] Navigation Engineer, Outer Planet Navigation Group.
[12] Technical Group Supervisor, Maritime and Multi-Agent Autonomy Group. AIAA Senior Member.
[13] Technologist, Machine Learning and Instrument Autonomy Group.
[14] Deputy Section Manager, Planning and Execution Systems Section. Senior Member – Lifetime.

I. Introduction

Robotic exploration spacecraft missions are designed and operated to collect data in order to answer scientific queries while simultaneously respecting operational constraints, with additional requirements on mission resilience and integrity of the scientific data. Standard mission systems engineering practices largely rely on static mission sequences to specify what data to collect, when, and how. The process of generating these sequences typically involves iterative trades between the science and engineering teams: the science team proposes measurements to collect relevant data, while the engineering team ensures the measurements are feasible in the context of the larger mission system capabilities and resources (e.g., power, pointing, and timing constraints are satisfied).

When situations occur that are outside the expected operating conditions of the script, the typical response is to enter a fail-safe mode and wait for ground-based operators to diagnose the situation and determine how to proceed. Additionally, static sequences are not typically written to respond to science observations. Instead, the data is generally transmitted to the ground for interpretation (and possible sequence modification) while the robot continues running its active script. While this approach is battle tested and leads to predictable behavior that nominally follows the operational constraints throughout the mission, it ultimately results in missions that are overly conservative with respect to science return and brittle with respect to unexpected situations, and uses mission-specific behaviors that are difficult to translate to new scenarios. Furthermore, there are entire classes of missions that require adaptation on a time scale too short to allow for round-trip communications with an operator that are largely impossible to perform under the current paradigm.

To enable these classes of missions to be successfully performed, we propose a set of techniques to enable remote robotic agents (satellites, rovers, landers, aerobots, or otherwise) to act as scientist and operations engineer avatars in the field. The key insight is that the mission's scientists and engineers on the ground must be able to directly specify their queries and high-level safety constraints to their agent and the agent must be able to autonomously plan and execute a mission based on them. To that end, we are engaged in a multi-year project to make three major contributions.

First, we are developing an information-theoretic programming language that allows the relevant stakeholders to specify their constraints and goals. Operators and engineers will be able to specify a model of the system (including what actions it can perform, along with their conditions and effects), behavior "patterns" that can be reused, and constraints on the state of the system that cannot be violated. Scientists will be able to specify their objectives in terms of questions that must be answered about scientific phenomena, as well as how much risk they are willing to incur to collect their desired information. The language will be implemented as an extension to an existing model-based, decision-theoretic, state-based, and risk-bounded language called RMPL (the Reactive Model-based Programming Language) [1]. This will allow us to include operational constraints already expressible in that language. Examples include specifying that a spacecraft agent must be 95% confident in its position and orientation (and associated uncertainty) estimates at all times (without specifying the exact set of sensors and sensing modalities to use), assuring that the spacecraft not collide with obstacles with 99.99% probability (without specifying the exact trajectory), and letting the spacecraft determine if a science measurement is feasible or not based on its estimated orientation relative to a target body and its camera slew rate.

Second, we are developing an information-driven, risk-bounded planning and execution system that is capable of accomplishing missions that satisfy the goals and constraints expressed in these programs. In particular, this system will be capable of generating a plan meeting the user specification, executing it, and continuously adapting the plan in response to in-situ measurements. This planning and execution system builds upon the Enterprise execution architecture [5], which consists of a collection of interoperable planners and executives [2,3,4], each specialized to specific tasks, that can be combined in various ways.

Third, we are developing an approach to architecting information-seeking autonomous systems. Based on the experience gleaned from prototype development in the first year of our effort, we will develop a trade space for the allocation of information and risk to different parts of the architecture, and we will analyze how different choices impact the system's decision-making performance. For example, one architectural choice would combine into a single complex planning problem the optimization of both science performance and spacecraft motion (which could be very computationally expensive to solve), whereas another choice would separate these two concerns, solving each problem independently based on constraints/assumptions made on the other, leading to a more computationally efficient but less accurate solution. Similar trade-offs can be made for activity planning, kinematic path planning, and dynamic trajectory planning, for example.

We will demonstrate these contributions in two motivating autonomous exploration scenarios. The first is an autonomous asteroid flyby mission using multiple spacecraft. The second uses autonomous underwater vehicles to find undersea vents.

We have previously described this project in Ref. [6]. In the remainder of this paper, we detail the progress we have made over the past year, which has been largely focused on formalizing the asteroid exploration problem, from both a science and autonomous planning perspective, and developing a simulated space systems testbed on which information-driven and risk-bounded autonomy frameworks can be tested. In Section II, we ground our goals by describing our two motivating scenarios. In Section III, we summarize related work and existing practices in this area. Next, in Section IV we intuitively describe the information seeking problem, its mathematical formulation, and how its solution is represented. Following that, we describe in Section V the information-driven, risk-bounded planning and execution architecture. In Section VI, we describe our initial demonstration of this autonomy capability, including integration of the planning and execution framework with a spacecraft simulator, and we present our current results. Finally, in the concluding Section VII, we summarize the benefits of our approach and describe future work to go on the project.

II. Scenarios

A. Asteroid Flyby

The first type of scenario considered for this study is motivated by the science community's interest in optimizing and increasing the science return of flyby missions. Flyby missions are seeing a renewed interest among the planetary science community: depending on the target, mission cost class, and science objectives, a flyby architecture may be the only affordable way to explore a destination of interest (e.g., Trident concept at Neptune's moon Triton [7]).

We are developing our information-seeking executive in the context of a science-driven asteroid family mission responsive to decadal science. One concept under consideration assumes that multiple smallsats explore several members of the Themis family located in the outer main belt (~3 au). The Themis family is of great scientific interest because it is formed from the disruption of a large (~290 km) icy asteroid that likely formed in the outer solar system [8] and represents the population of planetesimals that supplied water and organics to the inner solar system. Although an orbiter around the largest member of the family, 24 Themis, may be affordable under NASA's Discovery or New Frontiers cost caps [34], an architecture investigating several members of the family with a fleet of smallsats would test hypotheses about the internal structure of the family parent body. This multi-spacecraft exploration approach would enable varied measurements tailored to the various targets of interest in the family. However, each smaller spacecraft would be more resource limited and would have to deal with relative velocities of ~6 km/s and lighting conditions dictated by the mission trajectory design. Furthermore, very little would be known about the physical properties of the targets, except for their ephemerides and rotation periods, and even these would have significant uncertainties. These properties, as well as the targets' shapes, pole position, and albedo would be refined on approach. A target's spatially resolved phase function (brightness versus solar phase angle) informs about the presence of sites of particular scientific interest (e.g., ice exposed in crater floors), and helps optimize instrument settings in order to meet data quality requirements. However, a scene's variation in albedo will become resolved too late for ground in the loop updates to an observation sequence.

In the ideal autonomous scenario, CubeSats ("Scouts" or "Daughtercraft") are deployed ahead of the smallsats ("Mothercraft") to perform reconnaissance of regions of the asteroids whose characteristics of interest (e.g., regions of high brightness) are transmitted to the Mothercraft. Based on that information, the latter spacecraft refine their observation plans by targeting the regions believed to be ice-rich and with optimized instrument gain settings (e.g., narrow angle camera, infrared spectrometer).

For the experimentation in this paper, we will focus on a slightly simpler scenario. This scenario maintains the multiple vehicle aspect; however, we focus our exploration on a single body of interest, 24 Themis. Furthermore, the end goal is not to image only ice-rich craters. Instead, the goal is understand the relationship between the amount of ice in a crater as a result of its morphology, albedo, and location. In this scenario, it is perfectly acceptable for the Mothercraft to image a crater with little to no ice in it, so long as it helps refine the knowledge of the relationships between the quantities of interest. Our work to date has focused on an autonomously targeted flyby of 24 Themis, informed by observations taken by a single Mothercraft during its approach of the asteroid and by precursor flyby images taken by a Daughtercraft CubeSat deployed by the Mothercraft during the cruise/approach phase.

B. Unmanned Underwater Exploration

The second motivating scenario is unmanned underwater exploration. In this scenario, an autonomous underwater vehicle (AUV) is exploring a non-terrestrial ocean, such as Europa's under-ice oceans, and has the ultimate goal of finding underwater vents. These locations are interesting scientifically because they are expected to be the most likely locations where life can be found.

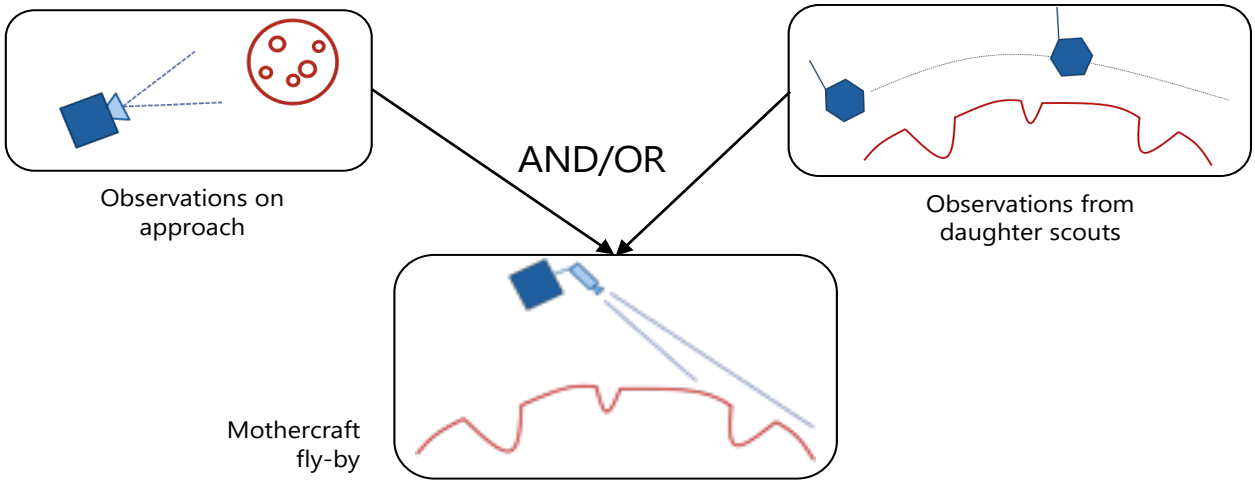

Fig. 1. Asteroid flyby scenario.

In this scenario, loss of the vehicle (for example by passing beyond crush depth or colliding with the seafloor) is a much greater concern than in the asteroid flyby scenario. This is because the undersea environment contains more external disturbances (such as unknown currents), the AUV must typically operate in close proximity to the seafloor, and navigation errors are compounded without absolute reference points such as stars. For that reason, this scenario strongly highlights the risk-bounded aspect of our planning and execution architecture. Furthermore, it serves as an example for other mission scenarios, such as those involving rovers or aerial vehicles, that must also deal with a high chance of mission failure due to loss of the vehicle.

The focus of the remainder of this paper is on the asteroid flyby scenario. For more information on the unmanned underwater exploration scenario, including a demonstration in simulation, see Ref. [6].

III. Current Practice and Related Work

In this section, we describe how missions similar to the asteroid flybys would be attempted under current practice, along with the restrictions that current practice would place on those scenarios. Additionally, we briefly overview related work in both moving reasoning onboard autonomous space agents and information-driven planning.

In the current state of practice, as described in Section I, a sequence of science observations would be pre-scheduled by the operations team on Earth, as the flyby event is sufficiently short that iterating with the ops team during the flyby would be precluded. Uncertainties would be mitigated by (1) adding margin in the observation timeline, resulting in a conservative measurement sequence; (2) taking images at multiple exposures to guarantee that features of interest meet quality requirements; (3) using a wide field angle camera instead of a narrow angle camera to capture a large scene instead of targeted regions. (1) and (2) would result in allocating precious resources to a dataset that is later disregarded, and (3) would result in compromising on spatial resolution, potentially leading to science loss if the regions of scientific interest are very localized. Clearly, providing a spacecraft with the onboard ability to make trusted decisions in real time about the flyby observations, based on precursor measurements from the Mothercraft and any Daughtercraft, should result in significantly better science return and decreased risk of mission failure (not meeting high-level science requirements).

Adaptive information-theoretic planning has been previously explored from a multitude of perspectives, whether with an explicit objective to maximize the information between observations and the environment at large [9,10,11] or utilizing information to maximize the value of observations over long horizons [12,13]. A common approach to modeling such problems, including in this work, is to represent the prior information about the environment using a Gaussian Process (GP) [14,15] and assume that the actual environment is consistent with that model. Prior work [10,13] has shown that greedy strategies (i.e., iteratively choosing the single best observation to take at a particular instant) produce solutions that are within a multiplicative factor of being optimal.

However, a common shortcoming of these approaches is that they do not consider any complex operational constraints that would be imposed on a spacecraft mission. In non-terrestrial, or even terrestrial but dangerous, environments, exploration is routinely subject to restrictions on communication, state (e.g., obstacles must be avoided), and resources (e.g., a maximum amount of energy can be expended during a day of operations). Additionally,

uncertainty in the environment and navigation introduces the possibility that constraints may be violated and the mission lost unless the agent is overly conservative. To handle this particular aspect, two broad approaches have been taken. One approach is to move reasoning onboard the spacecraft, as exemplified by the Remote Agent experiment on Deep Space One [16,17] and, more recently, the AEGIS autonomous science capability on the Curiosity rover [18]. The key upside to this approach is that if a novel and/or dangerous situation is encountered, the onboard autonomy can produce a new plan (that also respects operational constraints) to deal with it before remote operators may even learn about the situation. However, one limitation of this approach is that it only produces plans, not policies; this results in a delay between a novel state being observed and a new plan being generated to handle it.

The second broad approach to handle dynamic or dangerous environments is to explicitly reason about risk. When reasoning about risk, it is common to take a risk-bounded approach; that is, the agent can take potentially fatal actions, so long as the probability of the mission ending due to a constraint violation is bounded by some threshold Δ. Our approach builds off existing techniques for risk-bounded information-based planning [3,19], and risk-bounded motion planning [4]. Underpinning these planners are algorithms such as weighted penalty methods [20], extensions to heuristic forward search [21], or Monte Carlo Tree Search [22]. While these techniques typically focus on violation of state constraints, another type of constraint where violation must be considered is temporal. Temporal constraint violation results in missed deadlines which can result in mission failures such as the inability to use a communications window or rendezvous with another agent. This type of constraint violation is handled by risk-bounded schedulers [23]. Another approach, called chance-constrained optimization-based planning, solves a large-scale optimization problem that explicitly takes uncertainty of robot position and obstacle uncertainty into account [24, 25, 26]. To make these non-linear optimization problems tractable, the problem is often solved as a sequence of convexified problems around a nominal trajectory, which can lead to a conservative approach or even empty convex free-space [27].

IV. Information-Seeking Planning and Execution

The end goal of our information seeking planning and execution system is to execute a series of commands that maximize some information objective Q, while respecting operational constraints. In this section, we discuss the information objective for each of our example problems, how the planning problem is modeled, how the solution generated by the planner is represented, and how the solution is executed.

A. Information Objectives

The goal of the simplified asteroid flyby scenario used in this paper's demonstration is to image craters that will help us learn the relationship between some inputs (crater morphology, albedo, and location) and an output (ice cover). In the AUV scenario, we want to find the locations with the highest temperature. Underlying these scenarios is a physical process, represented as the function f, that we wish to learn more about. In the asteroid flyby scenario, learning that process is the direct goal; whereas in the AUV scenario, it is an indirect goal: if we learn the process which causes temperature fluctuations, we can better predict where there might be hot spots and then directly observe them to confirm it.

We represent our knowledge about these processes using a GP with output noise. GPs represent a probability distribution over an infinite dimensional vector of random variables, in which each random variable is indexed by a label. The use of infinite vectors allows GPs to describe probability distributions over functions where the input to the function is an index value $x \in X$ and the output y of the function is drawn from the random variable Y corresponding to that index ($y \sim Y$). The addition of index-dependent output noise $\nu(x)$ allows us to model the possibility that the exact same inputs to a process may produce multiple outputs. This may be the case if there are unmodeled inputs to the function, the underlying physical process is actually stochastic, or our observation of the output is itself noisy. In this work, we model the output noise as Gaussian with a mean of zero and a known variance σ^2. Formally, this means that given some index x, we observe some output y as described by Eqs. (1) and (2), where $N(0, \sigma^2)$ represents a Gaussian distribution with zero mean and variance σ^2.

$$y = f(x) + \nu(x) \tag{1}$$

$$\nu(x) \sim N(0, \sigma^2) \tag{2}$$

A GP is characterized largely by a kernel function K. This kernel function provides a measure of similarity between two separate indices x_1 and x_2. This similarity is represented as the covariance between the random output variables associated with each index, Y_1 and Y_2, respectively. One common kernel function is the squared exponential kernel,

K_{SE}, shown in Eq. (3). This kernel says that two outputs are strongly related the closer their input indices are to each other. The correlation falls off based on the characteristic length scale of the kernel, l.

$$K_{SE}(x_1, x_2) = e^{-\frac{|x_1 - x_2|^2}{2l^2}} \tag{3}$$

There are two primary operations that can be performed on a GP: assimilation of new data and prediction. When data is assimilated, the pair (x_i, y_i) representing an input and its corresponding output are saved for later use. Prediction takes a yet unseen index x_i and computes a probability distribution for its associated random variable Y_j. This is accomplished by using all saved data, along with the kernel, to compute how correlated Y_j is with the saved observations. The resulting distribution is normally distributed with a mean and covariance that depends on the data and kernel function. Additionally, for GPs with output noise, the variance also depends on the variance of $v(x_j)$. Prediction is a somewhat expensive operation that grows with the number of data points assimilated, cubed.

We use GPs to represent our knowledge of f for three primary reasons. First, they model smooth functions and we do not typically expect the output of the physical processes in our examples to change radically for small variations in location or morphology. Second, when used to predict the output of an unobserved input, the degree of uncertainty in that prediction is also produced. Last, the degree of smoothness in the function can be tuned by choosing an appropriate kernel. A benefit of this is it allows us to incorporate expert knowledge. For instance, we can encode that small changes in the longitude of a crater will affect the ice content less than small changes in latitude.

Let us first focus on the asteroid flyby scenario. In this scenario, the output of the process is the percentage of the crater which is covered in ice; the input is the crater location, its radius, and observed albedo; and the output noise represents the uncertainty in our ability to process the image and determine the amount of ice cover. Furthermore, we will use $\mathcal{E}$ to represent the infinite dimensional random vector that corresponds to f. In this scenario, our goal is plainly stated as choosing craters to observe that will maximize our knowledge of f. Formally, our objective is to choose a set of random variables to observe $Y_1, \dots, Y_N$ (corresponding to the ice cover for specific craters), such that the information objective shown in Eq. (4) is maximized, where $I(\mathcal{E}; Y_1, \dots, Y_N)$ represents the mutual information between the random variables we observe and the entire infinite dimensional vector.

$$Q = I(\mathcal{E}; Y_1, \dots, Y_N) \tag{4}$$

Mutual information can be defined in terms of entropy H, as shown in Eq. (5). Therefore, we can rewrite the information objective as shown in Eq. (6) to gain an intuitive understanding of what is being maximized. In the model we have described, $H(Y_1, \dots, Y_N|\mathcal{E})$ is purely a function of our observation uncertainty $v(x) \sim N(0, \sigma^2)$. This is because if we are given $\mathcal{E}$, then we know with certainty the output of $f(x_1), \dots, f(x_N)$; which leaves the value of $v(x_1), \dots, v(x_N)$ as the only source of uncertainty. If the variance were zero (i.e., there is no noise in our observations and the underlying process is completely deterministic), then $H(Y_1, \dots, Y_N|\mathcal{E})$ would be zero, because the observed random variables are necessarily a subset of $\mathcal{E}$. However, if the variance is non-zero, that entropy term is computed from the underlying normal distribution. With this framing, we can intuitively see that Q is maximized if we both choose craters where our uncertainty in interpreting the data is minimized ($H(Y_1, \dots, Y_N|\mathcal{E})$ is minimized) and those craters have little in common with each other, with respect to the GP kernel ($H(Y_1, \dots, Y_N)$ is maximized). This latter point can be seen, as the more correlated two random variables are (i.e., the GP kernel evaluates to a high value), the lower the entropy of their joint distribution.

$$I(X; Y) = H(Y) - H(Y|X) \tag{5}$$

$$Q = I(\mathcal{E}_c; Y_1, \dots, Y_N) = H(Y_1, \dots, Y_N) - H(Y_1, \dots, Y_N|\mathcal{E}) \tag{6}$$

We now turn our attention to the AUV scenario. The information objective is to determine where in the environment some quantity (temperature) is highest. There are two methods by which these maximums can be found: they can be directly observed (preferred), or their location can be inferred by learning the underlying process and predicting where they will be. This is encapsulated by the information objective shown in Eq. (7). The first term states that we want to choose locations to observe where the expected value of the quantity of interest is high. The second term controls how much we prefer learning the underlying process, where α is a constant that lets us balance exploration (learn the process) with exploitation (observe the interesting locations directly)

$$Q = \sum_{i=1}^{N} E[Y_i] + \sqrt{\alpha\, I(\mathcal{E}; Y_1, \dots, Y_N)} \tag{7}$$

We will focus on the asteroid flyby scenario for the remainder of this paper.

B. Planning Problem

We model the underlying physical process of making observations as a finite-horizon, discrete time, chance-constrained Markov decision process (CCMDP). A CCMDP represents the possible states of the world as the set S. A subset of these states, $B \subseteq S$, are considered safe. In every state at time t, $s_t \in S$, there is a set of actions that are available to the agent $A(s_t)$. At every point in time, $t \in [0, T]$, the agent chooses an action to perform $a_t \in A(s_t)$. This action results in the state transitioning to another state s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$. After every transition between states, the agent accumulates a reward $R(s_t, a_t, s_{t+1})$.

The goal of a CCMDP is to execute a set of actions that results in the greatest expected reward, while bounding the probability that an unsafe state (not in B) is entered to a user-defined threshold Δ.

In the asteroid flyby scenario, the CCMDP state at time t consists of three elements. First is the position of the spacecraft along its trajectory, s_t^x. Second is the orientation of the camera, s_t^c. Last is the set of observations the spacecraft has made s_t^y.

An action represents the crater that the spacecraft will attempt to observe next. Performing an action takes time (to slew the camera and potentially wait for the crater to enter the frame), however the time to perform the actions are not identical and no concurrent actions are allowed. The set of actions available at time t is computed by determining which craters are still observable from s_t. That is, the combination of s_t^x and s_t^c is such that the camera can slew to the crater before the spacecraft passes it and the crater has not already been successfully observed.

Given the state s_t and chosen action a_t, the spacecraft can transition to a number of states, depending on how long the spacecraft takes to image the crater and the results of that imaging. In the simplest case, this can be modeled using two successor states, one where the crater was successfully imaged under ideal conditions, and one where the crater could not be imaged at all, due to camera faults or uncertainty in spacecraft and/or camera orientation. In both of these states, the new position of the spacecraft is determined based on its velocity and the time needed to slew the camera from the previous position to the new crater and take images. Additionally, the new position of the camera is such that it is pointing at the expected location of the crater. Of course, an entire spectrum of successor states could be modeled that incorporates various uncertainties in velocity (including the possibility of failures of the camera gimbal) and/or uncertainties in the observation, and this spectrum will be investigated and enumerated in future work.

C. Policies

The output of a CCMDP planner is a policy π that maps states to actions. Intuitively, the policy encodes a program as a series of "if" statements. If the output of a CCMDP planner is viewed as such a program, the outcome of each action creates a branching point. At any point, the state trajectory (outcome of every action performed so far) can be used to determine the next action to perform. However, a policy has the potential to be more compact than a series of "if" statements. The reason for this is because it maps states to actions instead of state trajectories to actions. In problems where the same state can be reached multiple ways, this can result in an exponentially smaller representation of the solution. The AUV scenario displays this property more readily than the asteroid flyby scenario. This is because in the AUV scenario, the agent has a large degree of control over its complete state; however, in the asteroid flyby scenario the spacecraft has little control over its trajectory as it cannot reverse its course and backtrack to an earlier position. Given this information, a policy is rather straightforward to execute. If the initial state of the world is determined to be s_0, first execute the action returned by $\pi(s_0)$. Then, make the observations, estimate the current state of the world to be s_1, and then execute the action returned by $\pi(s_1)$. Then repeat until the mission is complete.

V. Information-Driven Risk-Bounded Planning and Execution Architecture

In this section, we describe the various components of our planning and execution framework, with a focus on the pieces actually demonstrated in the following sections.

A. Overview

The overall information-driven and risk-bounded autonomy framework that we are developing is shown in Figure 2. The user inputs a high-level specification of the agents being controlled, the environmental model, mission constraints, a control program specifying the desired behavior of the agent, and an information objective. The control program can be specified as actions the agents must perform (and when), time-evolved states (goals) the lower-level

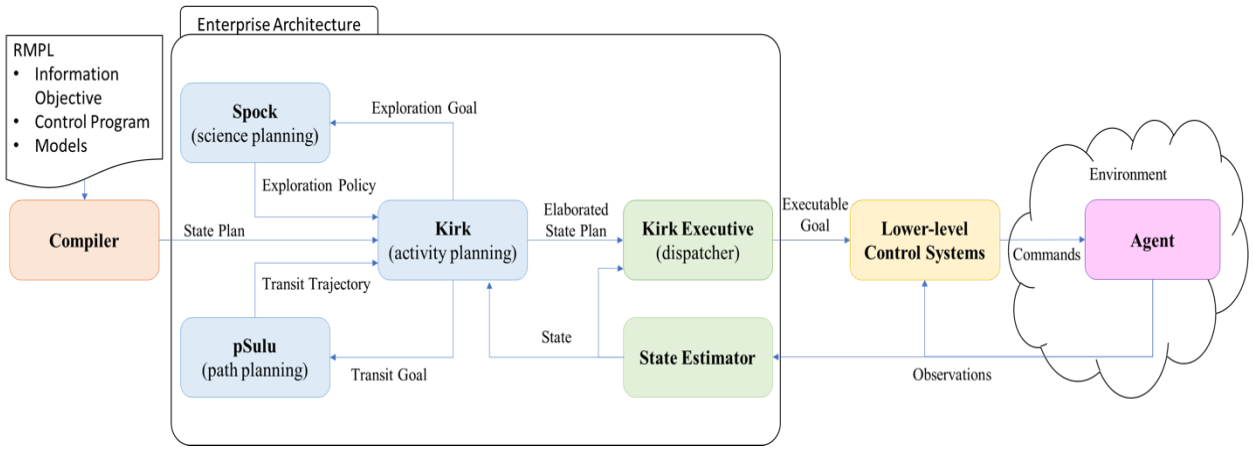

Fig. 2. Information-driven enterprise architecture.

control system must achieve, or some combination thereof (allowing for a form of adjustable autonomy). This input is currently largely expressed in RMPL [1], with the exception of the information objective; this will be folded into RMPL during the next year of this project. The user's input is then compiled into a state plan SP, a graph-like representation that captures the desired goals and state evolution of the system over time, including all temporal, state, and risk constraints.

On the other end, the planning and execution framework interacts with an agent through a lower-level control system. This control system has a set of goals that it is able to achieve, called *executable goals*. Examples of executable goals for the asteroid scenario include "image crater at coordinates (θ, ϕ)" and "launch Daughtercraft." The planning and execution framework sends these executable goals to the lower-level control system to be performed at the appropriate times.

Once the user's state plan has been compiled, it is handed off to the Kirk activity planner [2]. Kirk works with the Spock [3] and pSulu [4] planners to incrementally elaborate that state plan. A state plan SP_2 is a more elaborated version of a state plan SP_1, if every state trajectory that satisfies SP_2 also satisfies SP_1 and SP_2 either has fewer solutions than SP_1 or some goal episodes have been replaced by executable goals. A goal is considered executable if it cannot be further elaborated by the planners and is a goal that the underlying control system understands. For instance, the input state plan for the asteroid flyby scenario contains a goal that the asteroid is explored such that the information objective is maximized. Kirk will see that this goal is not elaborated (the lower-level control system does not know how to achieve an exploration goal), so it will ask Spock, the science planner, to elaborate that part of the plan. Spock will reply with a policy that, as previously discussed, specifies which actions should be taken under different states the agent may find itself in. This process continues until the plan is fully elaborated.

Once a fully elaborated plan has been computed, the Kirk executive begins dispatching it. Dispatching consists of sending executable goals to the agent's control system. Meanwhile, a state estimator is continuously monitoring the various sensors on the agent to determine the true state of the agent and environment. The executive uses both direct feedback from the executable goals and the output of the state estimator to determine what further executable goals need to be dispatched per the state plan and when. Additionally, the executive monitors the state for any sign that it is deviating beyond what the elaborated state plan can handle. If such a situation is detected, a new planning phase starts, using the estimated state to determine the initial conditions.

In the asteroid flyby scenario demonstrated in this paper, the pSulu path planner is not needed as the agent has very little control over its own trajectory during the flyby. As such, the remainder of this section will focus on the Spock and Kirk planners and how they work. For further information on pSulu or RMPL, see Ref. [6].

B. Spock

Spock is a risk-bounded adaptive sampling planner that looks for science targets to maximize measured science information [3,6,19]. When given a state plan to expand with an information objective, Spock extracts the potential observations from the environment model and state estimate. It then formulates a CCMDP problem as described above. Spock is able to formulate a CCMDP that guarantees a dynamically feasible control policy, so long as it is provided a description of the system dynamics and a solver that can compute a control trajectory to reach a desired

agent state. Spock then generates a policy such that the expected value of the information objective is maximized, while constraining the probability of entering an unsafe state.

Spock produces its policy using Monte Carlo Tree Search (MCTS). Intuitively, this process involves incrementally building up a search tree through the state space of the CCMDP. Instead of performing an exhaustive search and explicitly building the tree all at once, it alternates between expanding the tree and using simulation to refine the expected value for the information objective under a subtree. First, it chooses a node (representing a CCMDP state) that has unexpanded children. Then, it adds one of its children to the tree explicitly and estimates the information gain on that path by simulating the rest of the mission. During the simulation phase, it produces successor states by sampling a single successor from the probability distribution $p(s_{t+1}|s_t, a_t)$ instead of exhaustively examining all possible successors. Once the search has finished, a policy is extracted by walking the tree and collecting all the actions in every state that have the highest expected value of the information objective. MCTS is an anytime algorithm, meaning a feasible policy can be quickly computed, and after a feasible policy is generated, the agent can devote any remaining planning time available to computing improvements to the policy (i.e., changes to the policy that increase the expected value of its information objective function). MCTS accomplishes this by continuously balancing exploration and exploitation, meaning that Spock will always spend some time refining paths that look promising and some time refining paths that may not look promising, but have not been explored enough.

Spock further extends MCTS with the ability to stop exploring branches of the policy if they are deemed too risky [21]. For every trajectory through the state and action space of the problem ($< s_0, a_0, s_1, a_1, \dots, s_T >$), Spock computes the probability of leaving the set of safe states. If that probability ever exceeds Δ, the series of actions leading to it is pruned from Spock's search tree. This guarantees that any policy produced by Spock will stay in B with at least probability $1 - \Delta$.

C. Kirk

Kirk is an optimal activity planner that finds temporally feasible plans that minimize defined costs [2,6,28]. The Kirk activity planner is responsible for making discrete choices and ensuring temporal consistency within risk bounds. As mentioned above, Kirk receives the temporal and state constraints in the form of a state plan graph. State plans symbolically represent points in time called "events." Events can be constrained temporally, such that the difference in the execution time of two events must lie within a specified interval. Executable goals to be executed are represented using *episodes*. An episode has a start event and an end event. An episode that represents an executable goal additionally has a state constraint associated with it; this represents a constraint that must be satisfied by the system throughout the extent of the episode. Last, state plans can encode discrete choice by specifying a set of episodes and stating that only one of them must be chosen for execution.

Kirk operates internally by reformulating the input state plan as a Constraint Satisfaction Problem (CSP). All of the discrete choices between episodes are extracted as decision variables in the CSP. Then the set of constraints is built up from the temporal constraints and episodes' state constraints. Additionally, Kirk inserts constraints into the CSP that requires (i) any preconditions for an action to be met before the action is executed and (ii) two actions do not interfere (if they consume the same resource, for example). This CSP is solved using the OpSat (Optimal Satisfiability) solver. OpSat chooses the optimal full assignment to the decision variables, such that a solution to the remaining variables (such as temporal events or continuous state) exists, given the decision variable assignments. Temporal consistency is checked using efficient incremental solvers such as ITC [29] and Rubato [23]. A unique feature of Kirk is that it determines if a solution exists to the remaining (non-temporal, non-decision) variables by integrating with sub-planners. As a concrete example, if the goal episode to explore an asteroid is active, Kirk will ensure that a solution exists by invoking Spock to generate a concrete plan to achieve that goal.

The output of the Kirk activity planner is an executable state plan. A state plan is executable if it is conflict-free and fully composed of executable goals. The Kirk executive then executes this plan by dynamically assigning execution times to events and dispatching the necessary actions to the agent.

VI. Demonstration and Results

In this section of the paper, we describe the software environment we have developed for demonstration of our information-seeking planning and execution system and present the results from initial testing of the capability. We first discuss the simulation system that we have selected and adapted for our testing purposes, then we describe the overall software architecture for our demonstration testing. Finally, we present the results from running our information-seeking autonomy capability in the context of the asteroid flyby scenario.

A. Simulation Environment

In order to test and demonstrate our information-seeking autonomy capabilities, we are developing a comprehensive simulation pipeline that will enable us to execute a variety of interesting scenarios that showcase the planner's capability to handle uncertainty, while optimizing for the mission objectives. We considered a broad set of simulation tools, including several that were developed at JPL, as well as other tools developed outside JPL, e.g. open source and commercial off-the-shelf products. The functionality of the simulation tools fall into three classes: spacecraft simulation, trajectory simulation, and visualization. Spacecraft simulators are generally larger scale software frameworks that model the dynamics of the spacecraft (and usually celestial bodies) as well as the hardware and software of the spacecraft. Trajectory simulators are usually standalone tools used to calculate and show the trajectory of the spacecraft as it is traveling through space. Finally, visualization tools are used for visualizing celestial bodies, such as an asteroid with craters, and provide realistic rendering of the bodies. Ultimately, the open-source tool Basilisk [30] was chosen for its ease of use and reconfigurability, which allows us to perform rapid prototyping and scenario setup. An entire mission can be simulated in a single script. Custom components that interact with the simulation directly, such as spacecraft control laws, are easily integrated as Python modules. Additionally, Basilisk comes with a visualization tool called Vizard, built using the Unity game engine, that provides us with an immediate visual representation of the simulation, and relatively realistic 3D scene rendering out of the box.

B. Demonstration Architecture

1. The Processes and their Interactions

To guide the development and integration of our autonomy and simulation software, we started by identifying the processes (software modules) in the system and how they should nominally interact. For this activity, message sequence diagrams [31] were used. Figures 3 and 4 show the resulting message sequence diagrams for the Mothercraft and Daughtercraft, respectively. In general, a message sequence diagram consists of a collection of processes (actors) on the top, underneath each of which is a vertical timeline. Each process corresponds to a specific software module to be implemented. Horizontal arrows between the time lines show messages being sent from one process to another (or alternatively, function calls), in the order in which they occur. Such a diagram is read top to bottom, indicating the passage of time.

Figure 3 shows the sequence diagram of the Mothercraft. The processes are: the planner (the Kirk activity planner and Spock), the executive that executes the plans (the Kirk executive), the flight software (FSW), and finally the simulator for the Mothercraft. A process timeline for the Daughtercraft is also shown, since the Mothercraft interacts with the Daughtercraft. Our asteroid flyby scenario starts with the planner sending a plan to the executive, which is the initial approach procedure. The FSW is then instructed to start the approach procedure. The FSW enters a loop where it repeatedly monitors how close it is to the asteroid, until it is close "enough" to take context images of the full asteroid with its gimballed narrow-angle camera. At this point it slews the camera to point at the asteroid, waiting for the right camera pointing to be achieved in another loop. It takes three pictures. The Mothercraft then waits until it has reached a distance from the asteroid where the Daughtercraft can be deployed toward the asteroid, at which point a launch command is sent to the Daughtercraft. The Daughtercraft will then eventually reach the asteroid and take its pictures (see sequence diagram for Daughtercraft, Figure 4) and send a target crater list (of locations, diameters and albedo measurements) back to the Mothercraft. The target crater list is sent back to the executive, which sends it to the planner. Based on this list and the original context images, the planner creates a policy for the Mothercraft flyby's crater imaging sequence, which results in sending maneuvering commands (for a latitude adjustment burn) to FSW and thruster commands to the spacecraft. Once the Mothercraft has arrived at the asteroid, science imaging begins.

In the sequence diagram for the Daughtercraft (Figure 4), the Daughtercraft receives the launch command, and enters a loop waiting until it is close enough to the asteroid. When close enough, it repeatedly takes pictures of craters, and finally sends the target crater list back to the Mothercraft.

2. Software Setup

Figure 5 illustrates the software processes making up the implementation of the demonstration pipeline. The pipeline consists of three main processes (reading from left to right): the planner/executive (Kirk), the FSW, and the simulation process, which captures the modeling of the spacecraft and environment. The fourth (rightmost) process handles visualization. These processes use the Robot Operating System (ROS) middleware framework [32] as the method of communication. The benefit of this architecture is that it is very similar to the one that would be required to integrate a more fully-featured simulation environment like JPL's DARTS, which we are still considering incorporating in the future, should the need arise to run more complex and high fidelity simulations. We would be able to retain the logic in the FSW process, but we have the option to swap out the underlying simulation framework and visualization.

3. Current State of Simulation

At this stage of development, the purpose of the simulation focuses on identifying the components and their interfaces. In particular, we have implemented skeleton versions of the FSW and simulator components that correctly send messages according to the interface in the sequence diagrams, and are able to properly interact with the planner. However, the full simulator state (spacecraft and asteroid trajectory) has not yet been fully integrated as work on the software architecture and scenario/trajectory setup in the Basilisk has occurred in parallel. Specific behaviors such as pointing the camera and taking a picture are currently stubbed out; future work would involve developing the Basilisk modules that implement these behaviors. Furthermore, modeling specific spacecraft sensors would provide the capability to introduce noise, for instance, which would allow us to test the planner in more interesting and complex ways.

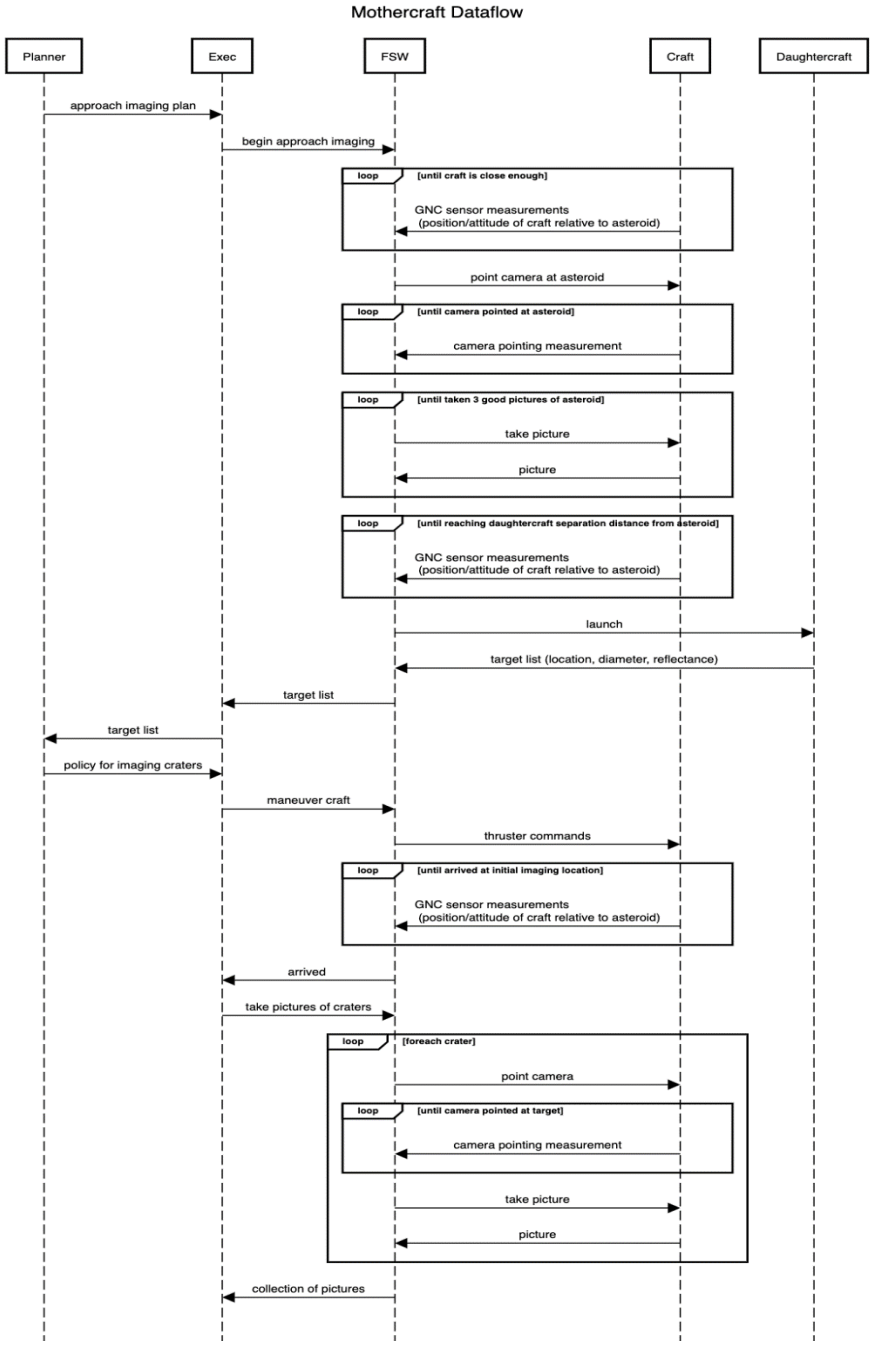

Fig. 3. Message sequence diagram for the Mothercraft.

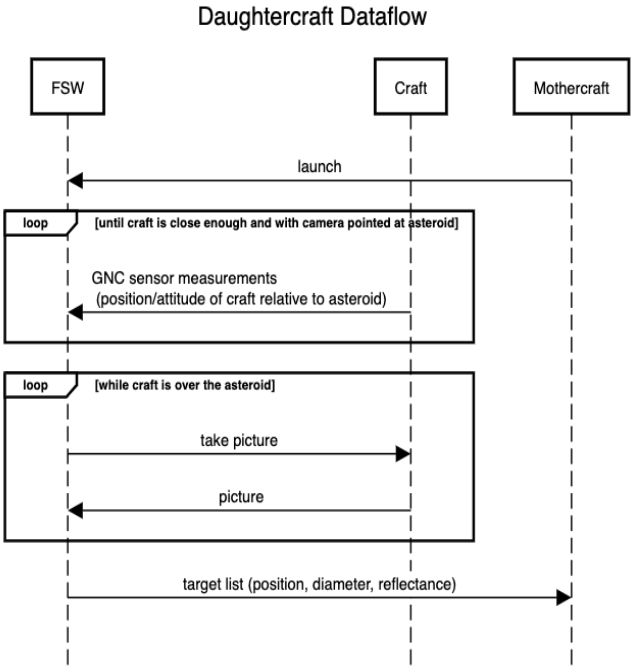

Fig. 4. Message sequence diagram for the Daughtercraft.

Fig. 5. Demonstration pipeline showing interactions between components of the software prototype.

```python
def handle_image_craters(self, msg):

    self.send_activity_accepted_msg(msg.id)

    rospy.loginfo("FSW: handle_image_craters with msg: {}".format(msg))

    for crater_id_polyvalue in msg.parameters[0].value.values:
        crater_id = crater_id_polyvalue.object_reference_value

        curr_crater_info = self.crater_id_object_map[crater_id]
        for slot in curr_crater_info.slots:
            if slot.name == "lat":
                crater_lat = slot.value.value.float_value
            elif slot.name == "lon":
                crater_lon = slot.value.value.float_value

        rospy.loginfo("FSW: processing crater id: {} with lat: {} lon: {}".format(
            crater_id, crater_lat, crater_lon))

        # point camera
        target_slew = crater_lat
        point_camera_msg = PointCamera()
        point_camera_msg.desired_camera_slew = target_slew
        self.point_cam_pub.publish(point_camera_msg)
        rospy.loginfo("FSW: sent PointCamera message")
```

Fig. 6. Python script.

C. Demonstration Results

In this subsection, we present figures that illustrate the output of our capability demonstration and provide a view into different parts of the implementation. Figure 6 depicts a function that encodes some of the interactions in the sequence diagram for the Mothercraft (Fig.), in this case the FSW receiving the policy from the executive, which outlines the sequence of craters to image. The function represents a callback in the ROS communication model, which we have implemented for each of the interactions shown in the sequence diagram.

A benefit of using the ROS framework is that it supports multiple communication paradigms. For example, the interactions between the executive and FSW are implemented using the client-service communications, while the FSW waiting for the spacecraft sensors to reach a particular state is better represented as a passive listener in the publish-subscribe model. Furthermore, by representing each interaction in the sequence diagram as a ROS message communication, we can easily log and trace the interactions and compare them to our designed scenario. Figure 7 shows part of a sequence diagram automatically generated (logged) from an execution using PlantUML [33]. The arrows represent actual ROS messages sent between the processes and allows us to compare to our original design. This also provides us a qualitative way to verify the correctness of our simulation behavior compared to the original designed sequence diagram.

Fig. 7. Part of a message sequence diagram generated from an execution of our asteroid flyby scenario.

Figure 8 illustrates Vizard (the rightmost process in Figure 5), the visualization tool that is distributed with the Basilisk framework. Vizard and the Basilisk process interact via the socket-based ZeroMQ communication library. The Basilisk simulation process pushes the state of all the bodies in the simulation over to Vizard at each time step, and Vizard renders each of the bodies. Vizard allows the loading of custom 3D models as well. Currently, we have chosen to use a scaled model of the dwarf planet Ceres as our stand-in for 24 Themis, as high-quality models of 24 Themis are not yet available. This is because Ceres has been extensively studied, which provides us with quality data to use for the basis of our simulation.

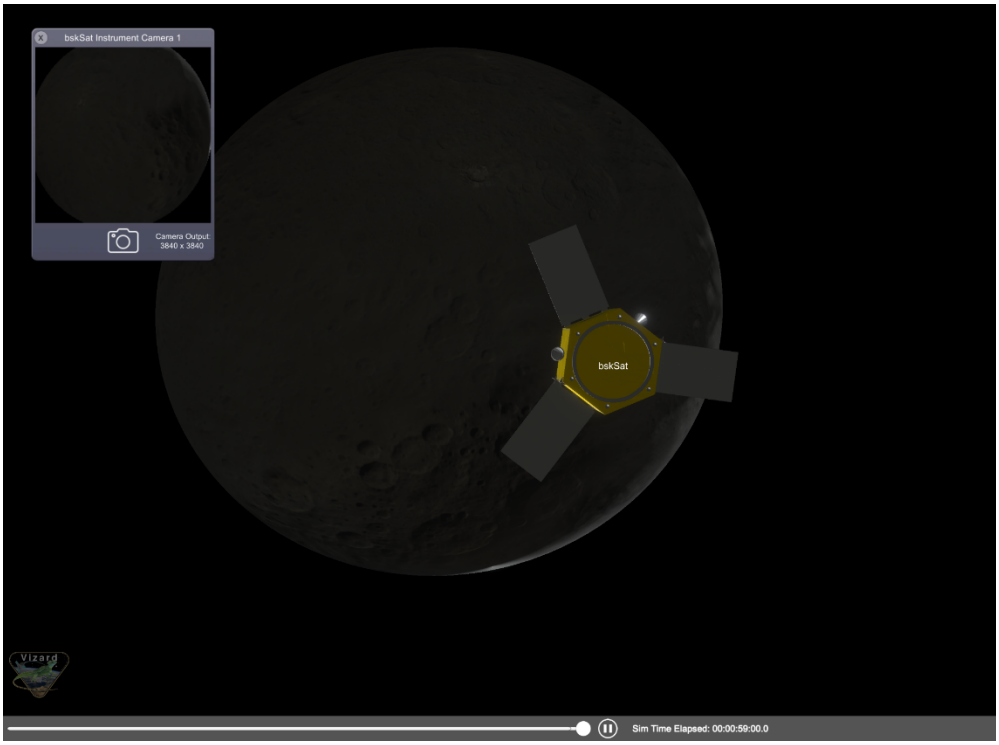

Fig. 8. The visualization Vizard.

Besides providing nice graphics for the autonomy demonstration, a visualization tool which can generate photo-realistic images is a secondary requirement for our project, as image-based observation is the primary form of science measurement in our mission. As shown in Fig. , we can create a simulated camera on the satellite, and using the mapped crater locations of Ceres, we can simulate which craters are observable in any image taken from a particular viewpoint. This information would be provided by the FSW to the planner for its decision-making. Future work would involve performing actual image processing on the images to get us closer to a fully autonomous pipeline.

We have demonstrated the complete pipeline from the Kirk and Spock planners, all the way through the simulated Mothercraft and Daughtercraft on a spherical asteroid with randomly generated craters. The craters were placed uniformly over the asteroid's surface and their physical properties were sampled from uniform distributions. In order to generate the "true" ice cover as a percentage, the function f that takes asteroid properties as input and outputs ice cover was sampled from a Gaussian Process. The mean of the GP was chosen such that ice cover increased with depth, decreased with radius and increased with albedo. The kernel function used was a squared exponential kernel with characteristic lengths of 5m for depth and radius, 1000 km for position, and 0.01 for albedo. These characteristic lengths mean that the ice cover of a single crater is closely correlated to the ice cover of craters with similar depths, radii, and albedos at practically any other location on the asteroid. This map of craters and their properties was provided to the Daughtercraft simulation, which then fed the results back to the Mothercraft and the planning and execution system after the action to image the crater during approach was performed.

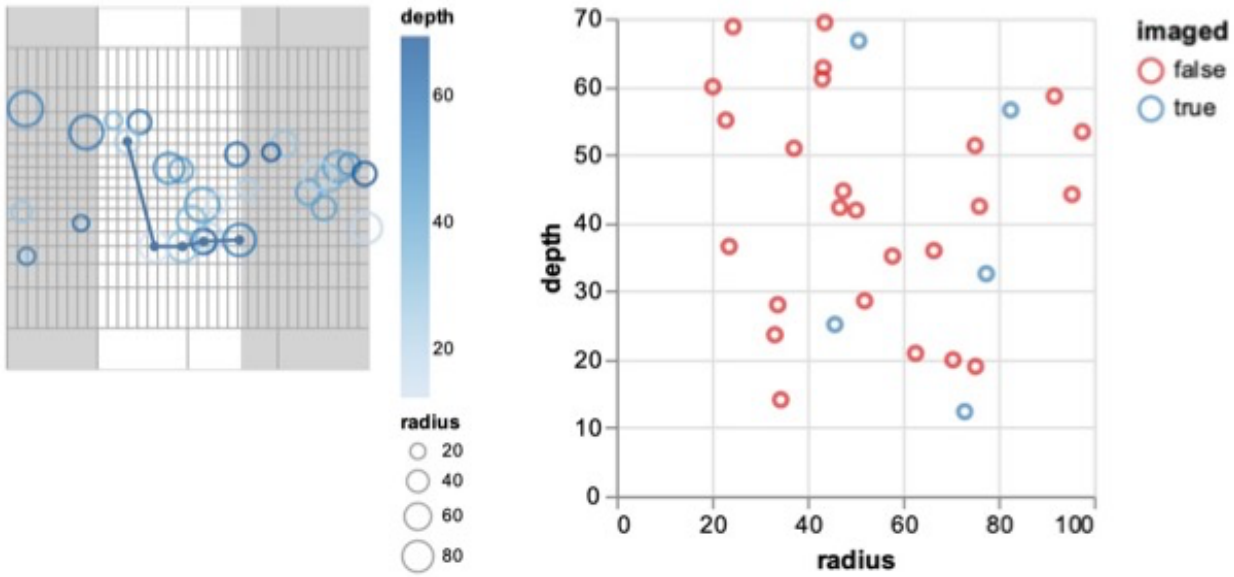

Fig. 9. (L) Sequence of craters imaged during the flyby, constrained by the flyby trajectory (traversing left-to-right) and the camera gimbal slew rate limit. (R) Our info-seeking planner selects the craters to optimize coverage of the feature space (Radius, Depth).

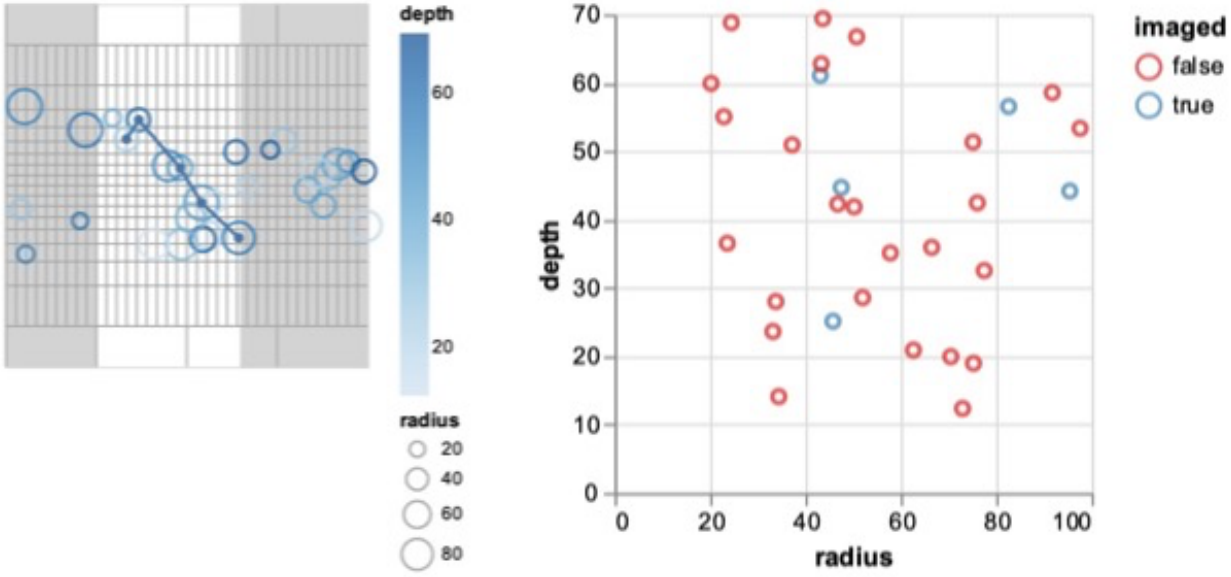

Fig. 10. When the camera slew rate limit is decreased, different craters are selected to satisfy this constraint, while still maximizing for feature coverage.

We show two representative examples of the craters that the information-driven and risk-bounded autonomy system chose to image on close approach with the Mothercraft in Figures 9 and 10. The same crater map is used in both figures and for ease of presentation, the albedo of each crater has been assigned the same value, and the craters vary only in location, depth, and radius. Additionally, the uncertainty in the measurements taken by the Mothercraft is assumed to be constant and independent of which crater is being imaged. The left side of each figure shows the position of craters on the asteroid's surface. The Mothercraft moves from left to right across the surface of the asteroid along its equator, and any crater in the shadowed area is not visible during the flyby. The right side of each figure plots the craters with respect to their depth and radius. In this image, a crater is colored blue if the Mothercraft images

it and red otherwise. The only difference between Figures 9 and 10 is that the camera slew rate has been reduced in Figure 10.

In Figure 9, we observe that the Mothercraft images its first crater at a high latitude north of the equator, then quickly slews the camera to image craters at roughly the same latitude south of the equator. In the right image, we can clearly see that the Mothercraft has imaged craters that are spread throughout the input space to the function being learned. This is because the chosen kernel says that ice cover between craters with similar depths and radii are strongly correlated. Therefore, observing a crater provides significant information about the craters that neighbor it in the depth-radius coordinate system. As the goal is to learn the relationship between crater properties and ice cover over the entire input space, the information objective is maximized by spreading out the observations across the input space. This is a visual representation of the intuition discussed regarding Eq. 6: the entropy of the joint distribution is maximized by choosing to observe craters that have little in common with each other. The same behavior is observed in Figure 10 after accounting for the reduced camera slew rate. The Mothercraft starts by observing the same first crater, but as the camera moves slower, it cannot image the same second crater before flying past it. Instead, a more gradual slewing rate is chosen to image craters at a variety of latitudes. We observe from the right-hand image that the craters are still spread out through the depth-radius coordinate system, but not quite as much as in Figure 9.

VII. Conclusion

In this paper, we have presented an overview of an information-seeking risk-bounded planning and execution system to enable future spacecraft to make decisions and adapt their behavior to seek out the most high-value scientific information, while bounding risk of failure. Information-seeking autonomy is an innovation with potential for transformational impact on the way our spacecraft enable scientific discoveries, by complementing traditional scientist-in-the-loop operations with appropriately-conservative onboard direction of science measurement activities (based on scientist-specified models).

By combining science operator-directed activities with autonomous science-driven exploration, our approach addresses biases associated with manually selecting measurements and measurement sites that are thought to be most scientifically valuable, resulting in a measurable increase in science return, compared to a spacecraft that does not have this autonomous execution capability. That is, we are endowing the spacecraft with the common sense to avoid the temptation of only visiting sites that are known to scientists and thus only add incremental science value (analogy to only looking for lost keys under the street lamp). Furthermore, by enabling spacecraft to respond to dynamic and uncertain environments autonomously, we seek to decrease the average time needed to respond to off-nominal scenarios (by potentially as much as an order of magnitude), by significantly reducing the frequency of required downlink-uplink iterations. That is, we are endowing the spacecraft with greater self-reliance.

We are developing our information-seeking executive in the context of a science-driven asteroid family mission responsive to decadal science. We have presented a mission architecture description and driving scenarios for our novel autonomy capability. We have developed an information-driven risk-bounded executive capable of reasoning over science information value while considering risk. To demonstrate the capabilities of the provided executive, we have implemented basic spacecraft simulation capabilities, leveraging the Robot Operating System and the Basilisk open-source spacecraft simulation framework.

The key innovative features of our executive are as follows:

(i) It chooses an ***information-maximizing measurement strategy*** based on the environment and initial measurements, e.g., based on lower-resolution images taken by the Mothercraft on approach to the asteroid, we choose a set of crater observations taken by a gimbaled science camera during its flyby, to provide the highest value from a scientific perspective.

(ii) It ***takes various mission risks into account*** in planning and executing its activities, such as the risk of missing important science due to pointing uncertainty, particularly for more oblique off-nadir observational geometries.

(iii) It can ***flexibly adapt its strategy on-the-fly*** based on information from collected measurements, e.g., modifying the Mothercraft flyby observation plan to target the most promising craters based on additional information communicated by the Daughtercraft.

(iv) Finally, it ***demonstrates resilience in accomplishing its high-level science objectives***, even in the presence of failures/degradations, e.g., by using the policy to quickly modify the plan in response to missed observations of craters during the flyby, to optimize the scientific information return as much as possible for the remaining observations.

Our prototype capability has already demonstrated the first of these four features. The other three are planned to be demonstrated in future work.

Also planned for future work, we will extend RMPL to consider the information-theoretic goals discussed in this paper. We will also extend Spock to fully incorporate the state uncertainties of the spacecraft. Moreover, as the asteroid

scenario has no obstacles that need to be avoided (unlike the AUV scenario), we will model other pertinent sources of risk (e.g., risk of missing important science due to pointing uncertainty for off-nadir measurements, as mentioned above), and we will investigate the use of chance constraints to ensure a minimum level of science return. We will integrate more realism and complexity into the simulation capability, so more challenging scenarios can be considered. Finally, we will make more progress toward an approach to architecting information-seeking autonomous systems. Based on the experience gleaned from prototype development in the first year of our effort, we will develop a trade space for the allocation of information and risk to different parts of the architecture, and we will analyze how different choices impact the system's decision-making performance.

Acknowledgments

The work described in this paper was performed at the Massachusetts Institute of Technology, the Jet Propulsion Laboratory, California Institute of Technology, and the Woods Hole Oceanographic Institution, under contracts with the National Aeronautics and Space Administration and the National Science Foundation.

References

[1] Williams, B. C., Ingham, M. D., Chung, S. H., and Elliott, P. H., "Model-Based Programming of Intelligent Embedded Systems and Robotic Space Explorers," *Proceedings of the IEEE*, Vol. 91, No. 1, pp. 212-237, 2003.

[2] Kim, P., Williams, B. C., and Abramson, M., "Executing Reactive, Model-based Programs through Graph-based Temporal Planning," *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 487-493, 2001.

[3] Ayton, B., Williams, B., and Camilli, R., "Measurement Maximizing Adaptive Sampling with Risk Bounding Functions," *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, Honolulu, pp. 7511-7519, 2019.

[4] Ono, M., Williams, B. C., "Iterative Risk Allocation: A New Approach to Robust Model Predictive Control with a Joint Chance Constraint," *47th IEEE Conference on Decision and Control*, Cancun, pp. 3427-3432, 2008.

[5] Timmons, E., Ayton, B., Wang, A., Pascucci, N., Zhang, Y., Bhargava, N., Reeves, M., Duguid, Z., Strawser, D., Fang, C. and Camilli, R., "Risk-bounded, Goal-directed Mission Planning and Execution for Autonomous Ocean Exploration," *Astrobiology Science Conference*, AGU, 2019.

[6] Ayton B., Reeves, M., Timmons, E., Williams, B. C., and Ingham, M. D.,"Toward Information-Driven and Risk-Bounded Autonomy for Adaptive Science and Exploration", *ASCEND*, pp. 4149. 2020.

[7] William, F., Bearden, D., Mitchell, K. L., Lam, T., Prockter, L., and Dissly, R., "Trident: The Path to Triton on a Discovery Budget", In *2020 IEEE Aerospace Conference*, pp. 1-12. 2020.

[8] Meech, K., Raymond, S. N., Meadows, V., Arney, G., Schmidt, B., and Des Marais, D. G., "Origin of Earth's water: sources and constraints", *Planetary Astrobiology*, 325, 2020.

[9] Binney, J., Krause, A., and Sukhatme, G. S., "Informative Path Planning for an Autonomous Underwater Vehicle", *IEEE International Conference on Robotics and Automation*, Anchorage, pp. 4791-4796, 2010.

[10] Krause, A., Singh, A., and Guestrin, C., "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies," *Journal of Machine Learning Research*, Vol. 9, pp. 235-284, 2008.

[11] Yilmaz, N. K., Evangelinos, C., Lermusiaux, P. F., and Patrikalakis, N. M.,"Path Planning of Autonomous Underwater Vehicles for Adaptive Sampling Using Mixed Integer Linear Programming," *IEEE Journal of Oceanic Engineering*, Vol. 33, No. 4, pp.522-537, 2008.

[12] Garnett, R., Krishnamurthy, Y., Xiong, X., Schneider, J., and Mann, R., "Bayesian Optimal Active Search and Surveying," *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, pp. 843-850, 2012.

[13] Srinivas, N., Krause, A., Kakade, S., and Seeger, M., "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design," *Proceedings of the 27th International Conference on Machine Learning*, Haifa, pp. 1015-1022, 2010.

[14] Low, K. H., Dolan, J. M., and Khosla, P., "Information-Theoretic Approach to Efficient Adaptive Path Planning for Mobile Robotic Environmental Sensing," *Proceedings of the Nineteenth International Conference on International Conference on Automated Planning and Scheduling*, AAAI, Thessaloniki, pp. 233-240, 2009.

[15] Marchant, R., Ramos, F., and Sanner, S., "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring," *Proceedings of the 30th Uncertainty in Artificial Intelligence Conference*, Quebec, pp. 553-562, 2014.

[16] Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. C., "Remote agent: To boldly go where no AI system has gone before." *Artificial intelligence* 103, no. 1-2, pp.5-47, 1998.

[17] Bernard, D., Dorais, G., Gamble, E., Kanefsky, B., Kurien, J., Man, G., Millar, W., Muscettola, N., Nayak, P., Rajan, K. and Rouquette, N., "Spacecraft autonomy flight experience: The DS1 Remote Agent experiment", *American Institute of Aeronautics and Astronautics*, AIAA, 1999.

[18] Francis, R., Estlin, T., Doran, G., Gaines, D., Johnstone, S., Montaño, S., Peret, L., Mousset, V., Gasnault, O., Frydenvang, J., Wiens, R., Schaffer, S., Pavri, B., Verma, V., Chattopadhyay, D., Bornstein, B., Mittal, N., and Deflores, L., "Incorporating AEGIS autonomous science into Mars Science Laboratory rover mission operations", *Proceedings of the 15th International Conference on Space Operations,* 2018.

[19] Ayton, B. J., "Risk-bounded Autonomous Information Gathering for Localization of Phenomena in Hazardous Environments," Master's dissertation, Aeronautics and Astronautics Dept., Massachusetts Institute of Technology, Cambridge, MA, 2017.

[20] Geibel, P., and Wysotzki, F., "Risk-Sensitive Reinforcement Learning Applied to Control Under Constraints," *Journal of Artificial Intelligence Research*, Vol. 24, pp. 81-108, 2005.
[21] Santana, P., Thiébaux, S., and Williams, B., "RAO*: An Algorithm for Chance-Constrained POMDP's," *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI, Phoenix, pp. 3308-3314, 2016.
[22] Ayton, B. J., and Williams, B. C., "Vulcan: A Monte Carlo Algorithm for Large Chance Constrained MDPs with Risk Bounding Functions," arXiv:1809.01220, 2018.
[23] Wang, A. J., and Williams, B. C., "Chance-constrained Scheduling via Conflict-directed Risk Allocation," *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3620-3627, 2015.
[24] Blackmore, L., Ono, M., and Williams, B. C. "Chance-constrained optimal path planning with obstacles", *IEEE Transactions on Robotics*, *27*(6), 1080-1094, 2011.
[25] Jasour, A., Aybat, N. S., and Lagoa, C. M, "Semidefinite programming for chance constrained optimization over semialgebraic sets", *SIAM Journal on Optimization*, *25*(3), 1411-1440, 2015.
[26] Jasour, A., Han, W., and Williams, B. C., "Convex Risk Bounded Continuous-Time Trajectory Planning in Uncertain Nonconvex Environments", *Robotics: Science and Systems (RSS),* 2021.
[27] Nakka, Y. and Chung, S. J., "Trajectory Optimization of Chance-Constrained Nonlinear Stochastic Systems for Motion Planning and Control," arXiv:2106.02801, 2021.
[28] McGhan, C., Murray, R. M., Vaquero, T., Williams, B. C., Ingham, M. D., Ono, M., Estlin, T., Lanka, R., Arsla, O., and Elaasar, M., "The Resilient Spacecraft Executive: An Architecture for Risk-Aware Operations in Uncertain Environments," *The AIAA Space and Astronautics Forum and Exposition*, pp. 5541-5561, 2016.
[29] Shu, I., Effinger, R., and Williams, B. C., "Enabling Fast Flexible Planning Through Incremental Temporal Reasoning with Conflict Extraction," *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, pp. 252-261, 2005.
[30] http://hanspeterschaub.info/basilisk.
[31] https://en.wikipedia.org/wiki/Sequence_diagram.
[32] https://www.ros.org.
[33] https://plantuml.com.
[34] Landis, M. E., Castillo-Rogez, J. C., Hayne, P. O. , Hsieh, H. H., Hughson, K. H. G. , Miller K. E., Kubitschek, D. et al. "Why We Should Study the Themis Asteroid Family in the 2023–2032 Decade." *Bulletin of the American Astronomical Society* 53, no. 4 , 2021.