Integrating Task Planning with Robust Execution for Autonomous Robotic Manipulation in Space

Emma Zemler Shaun Azimi Kevin Chang
Robotic Systems Technology Branch
NASA Johnson Space Center
2101 NASA Parkway
Houston, TX 77058
emma.zemler@nasa.gov

Robert A. Morris Jeremy Frank
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, California 94035
robert.a.morris@nasa.gov

Abstract

Future space exploration missions will include robotic assistants in human- habitable vehicles that perform complex tasks in logistics, vehicle maintenance, science experiments, and to respond to safety-threatening events. Such robotic systems will require the integration of deliberative behaviors such as task planning with reactive behaviors like perception and motion control. Such capabilities require the combination of expressive languages for generating and executing plans, with robust strategies for executing plans in a changing world. This paper describes a technology demonstration project involving robotic arm manipulation tasks using a science scenario inspired by bio-medical research performed on the International Space Station (ISS). The work described here significantly expands previous work performed by the same team (Azimi, Zemler, and Morris 2019) by adding capabilities in goal management, execution monitoring, and replanning.

Introduction

Future space missions will be enabled by a combination of human and robotic explorers. For example, a space station in Lunar orbit (Gateway 2018) will serve as a communications hub, science laboratory, habitation module, and holding area for rovers. Mobile and manipulation robots will assist crew in many different types of operational scenarios, including logistics (loading, unloading, configuring); maintenance (inspection, repair); science experiments (assisting crew, storing/retrieving samples); and ensuring vehicle safety (identifying and preventing fires or leaks). Human-robot teaming will thus ensure efficiency and safety of operations, and significantly reduce human workload. Similarly, for Lunar or Martian surface operations, robots will assist humans in construction and in-situ resource utilization (ISRU) tasks.

Although certain inter-space tasks might be best accomplished by robotic tele-operation, the complexity and size of operations will require a measured and varying degree of robotic autonomy. Robotic autonomy consists of an integrated combination of deliberative goal-directed and reactive behaviors to act efficiently and safely, and to respond effectively and in a timely manner to unexpected changes in the operational environment. AI planning offers rich expressive languages and high performance heuristic search algorithms for modeling deliberative behaviors. There is a wealth of research in combining deliberative task planning with low-level control to manage complex robot behaviors in a dynamic world.

NASA's Autonomous Systems and Operations (ASO) project develops and tests systems that assist crew members in performing science or logistic tasks in space. This paper describes a technology demonstration of autonomous planning and execution for a dexterous manipulator that is fixed in place. A mock-up of an International Space Station (ISS) science lab and an ISS-inspired bio-medical science scenario are used. This work extends previous work by the same team (Azimi, Zemler, and Morris 2019) by adding capabilities for planning and plan execution to make the system more robust to unexpected changes in the world. These capabilities include reasoning about time and goal preferences, monitoring execution, and mechanisms for plan repair and replanning. We use ROS and the ROSPlan framework to build an architecture that supports these capabilities, and to implement the layered architecture for robot behaviors.

Following a summary of related work, we describe the ISS science scenario in detail, followed by a complete overview of the technical approach. We discuss the demonstration and lessons learned, and conclude with a summary of current activity.

Related Work

Research in integrating task planning with the dynamics of a robotic system for autonomous manipulation is actively being explored by a number of research teams. A detailed review of this research is beyond the scope of this paper; here we only mention a handful of the efforts that are in specific ways related to our work.

First, this work intersects with work in developing frameworks that extend planning models to prioritize goals (Vattam et al. 2013). Second, this work overlaps with recent efforts in temporal flexibility in planning (Cashmore et al. 2017) (Huang et al. 2019), and as part of an execution system (Kim, Williams, and Abramson 2001). More generally, an execution monitoring framework has been proposed by many as a way to improve robustness in robotic systems (Pettersson 2005). Third, developing capabilities for plan repair and plan revision triggered by changes in the operational environment is an active area of research in the planning community (Fox et al. 2006), (Nebel and Koehler

1995).

Overall, the work described here follows the terminology and design framework proposed in (Ingrand and Ghallab 2017). More concretely, we view this work as consisting of developing a robotic deliberation system, in which a robot is performing actions in a goal-directed (or more broadly task-directed) manner to change the state of itself and its environment. Actions are either primitive or compound, and are organized in an abstraction hierarchy such that an action can be primitive at one level and compound at a lower (more reactive) level. A command is an action at the most primitive level, and is executed directly by the robot platform (the collection of sensors and actuators). Plans are organized collections of actions; skills are collections of actions that include commands. We will also sometimes speak of deliberative or reactive behaviors that characterize the functionality of the robot. This work broadly intersects with the vast literature on models and frameworks for integrating task planning and other high-level deliberation with motion planning, observation systems, and closed-loop actuation (Bagnell 2012), (Kaelbling and Lozano-Perez 2011), (Srivastava et al. 2014).

Operational Scenario

To illustrate diversity in handling different operational scenarios, the demonstration scenario here combines a science experiment with an emergency response. Many biological and materials experiments on the ISS utilize samples that are kept frozen until the experiment is conducted. Most experiments involve data products that include microscopic or other forms of imagery,and many of them also specify some other processing, such as mixing, separation (such as by centrifuge), heating etc.

The high level tasks for accomplishing a typical experiment are:

- Remove samples (centrifuge tubes) from freezer
- Defrost all samples;
- Expose samples to UV;
- Run some of the samples through a centrifuge; and
- Image all of the samples in the microscope (some after being spun, some not).

Furthermore, imaging tasks should not be performed during a Loss of Service (LOS) when data downlink with ground control is not possible.

The demonstration science experiment scenario reported here is an enhancement of the scenario presented in (Azimi, Zemler, and Morris 2019). In the remainder of this section we will describe two primary enhancements to the nominal scenario for improving the expressive power and robustness of the system.

First, to improve the expressive power of the task planning system a capability for prioritizing and clustering goals was added. Three priority levels, high, medium and low can be assigned to imaging tasks, and lower priority goals might not be accomplished as part of the generated plan. In addition, goal priorities are defined in terms of pairs of tubes to be processed.

Second, temporal constraints are added to the planning framework in order to model time-constrained activities such as those requiring data downlink or crew availability. PDDL planners such as LPG-td (Gerevini and Serina 2008) enforce these temporal constraints during plan generation. During plan execution, a timeline structure was used to enable monitoring of the progress of a plan in the presence of duration uncertainty. If an activity took longer than expected during execution, and the plan can no longer be executed successfully, a replanning activity was triggered, where goals can be removed or added based on priority.

Figure 1 shows a simplified illustration of a timeline and the replanning process. The top panel (A) shows a timeline with a planned sequence of tasks (with stacking to indicate that the tasks are performed concurrently). The blue dotted line indicates a deadline for the completion of all the tasks in the plan. Panel (B) illustrates the partial completion of the plan at a point at which failure of the original plan has been detected and repair actions initiated. The cause of the failure is the imaging task for Tube E taking longer than expected. Propagating the effects forward, it is determined that the deadline constraint will be violated (red bar overlapping deadline bar). As a result of repairing the plan (Panel C), certain lower priority tasks have been replaced by higher priority tasks, and the deadline constraint is again adhered to.

Finally, we studied and tested the required enhancements to a task planning system to enable an urgent response to a potential emergency. For this, we added an urgent response scenario. At any time during execution of an experiment, a fire detection alarm may be received. This indicates that a fire may be somewhere within the vicinity of the manipulator. The alarm triggers an emergency response by the system with the goal of isolating the specific location of the fire by probing fire ports with a smoke particle detector. The smoke detection data is reported to the ground station operator.

Technical Approach

We used the ROS module ROSPlan (Cashmore et al. 2015) to implement deliberative behaviors and provide the interface with lower level control of a manipulation system. Figure 2 provides a visual overview. The system is arranged in a standard layered structure, with deliberative capabilities controlling and receiving feedback from lower, more reactive layers, and the interactions among the components based on a client-server protocol. The deliberative layers consist of a ROSPlan manager (RPM), that commands and monitors the overall system. Within the RPM is a Goal Manager, that maintains a priority ordering of goals and manages the selection of goals for planning. The planning system generates and executes task plans. The plan dispatcher commands and receives feedback from the robot behavior components, which are themselves composed of sequences of simple actions. A Plan Viability Checker (PVC) serves as a monitor of the executing plan. The PVC can detect plan failure and trigger replanning. The world state is routinely updated and stored in a database.

The system components will now be described in more detail.

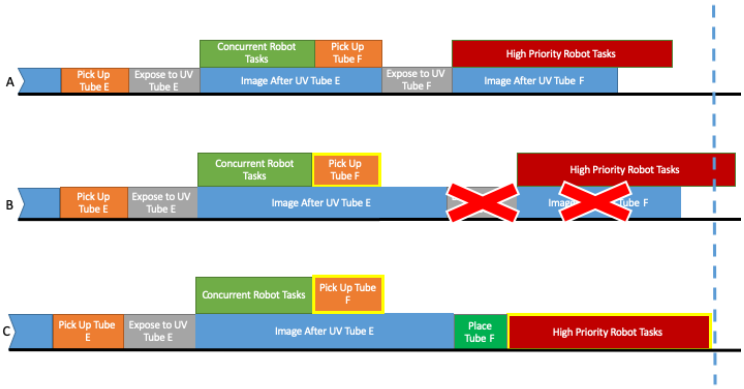

Figure 1: Illustration of Prediction of plan failure and re-planning with priority goals

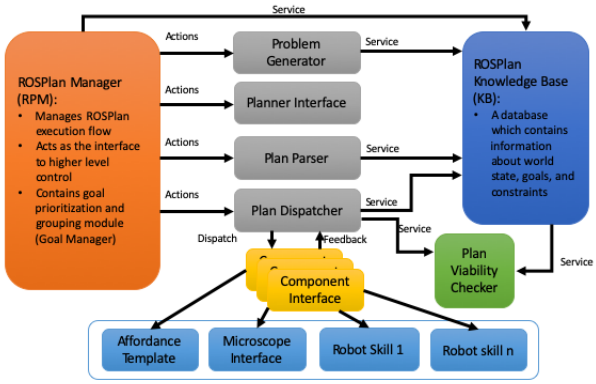

Figure 2: Integration of Task Planning with Robot Behaviors using ROSPlan

ROSPlan Manager/Goal Manager

Future intra-space activity will combine numerous science tasks with logistics and maintenance tasks that are time- and resource constrained. To maintain continuous goal formation, planning, execution, monitoring and replanning, the ROSPlan manager module (RPM) was developed as part of our effort. RPM provides a single interface for all feedback from the system. RPM enables automatic replanning upon plan failure if unachieved goals exist. It serves as a plan Executive. Finally, it provides the entry point for human users of the system for setting goals and priorities.

RPM contains a Goal Manager (GM) that can continuously manage and update activity itineraries (sets of goals). The GM follows the architectural structure of (Roberts et al. 2016), providing a goal reasoning capability both during planning time and execution time.

GM allows a user to cluster a set of related goals that need to be achieved together, or simply are considered equally important. The cluster of goals can then be assigned to a priority level (low, medium, high, or immediate) and added as a single entry to the GM queue. The GM determines which goals to add to a planning problem, and maintains a queue of uncompleted goals for a future planning problem, e.g., as a result of re-planning during execution.

The input to the GM is a set of goals, either user-generated or the result of previous processing (either during planning or execution, including the special case of the result of sensing a fire alarm). The goals are passed to the plan generator to determine the feasibility of the plan. If it is feasible, it is dispatched by the RPM. Otherwise, the GM updates the set of goals by shedding lower priority goals and invokes the planner again. This continues until a feasible plan (or no plan) is found. The overall goal ordering mechanism is loosely based on goal prioritization in PDDL3.0 (Gerevini and Long 2005), and the search mechanism for finding high priority plans resembles the hill-climbing approach for solving partial CSPs (Voudouris and Tsang 2000).

GM avoids the combinatorial complexity of solving the general optimal goal selection problem by applying a set of rules for ordering and removing goals. First, the GM will try to plan for the entire set of input goals. If this problem is infeasible, then the lower priority goals are removed, one at a time, until the planner returns a feasible plan. Finally, if the set of goals contains only goals of one level of priority, then the goals are removed using a user-specified pre-order of oldest-first or newest-first.

Finally, GM supports an 'emergency response' capability in response to alarms or other sensing of safety-threatening events by loading emergency response goals, such as isolating the location of a fire in the proximity of the ISS Express Rack (see below for a detailed description of the demonstration platform).

The GM provides the 'expertise' for re-planning performed by the RPM. The RPM can be invoked for re-planning either during planning time, as the result of failure to find a plan for an initial set of goals, or during execution, when the dispatcher notifies the RPM that an executing plan has failed. RPM formulates a new planning problem by reading the current world state and the list of goals provided by the GM. The RPM re-plan loop succeeds if there exists a

Figure 3: Robot Inspecting a Fire Port

set of goals that results in a valid plan; otherwise, the RPM fails and no plan is dispatched.

Task Planning

Planning problems are formulated by defining a PDDL 2.1 domain model and problem. For the science scenario, PDDL actions for picking and placing test tubes to and from either storage locations or from a centrifuge, for opening a closing a freezer door and centrifuge lid, and for imaging samples under a microscope, were defined (this action model is virtually the same as the one described in (Azimi, Zemler, and Morris 2019).)

To test the new desired deliberative capabilities (specifically robust execution and smart responses to potential threats to the spacecraft) the action domain was extended to include actions for inspecting a fire port using a fire probe, as well as actions that quickly transition the science work space into a safe state prior to inspecting for fire (see Figure 3).

The resulting planning problem is solved using the heuristic forward search planner LPG-td (Gerevini and Serina 2002). The ROSPlan system outputs a Esterel plan (Berry 2000). An Esterel plan is a directed acyclic graph with nodes representing the start or end of activities and the edges representing either causal or temporal orderings in the plan. The ROSPlan dispatcher, in conjunction with the Plan Viability Checker (see below) executes each enabled action in the plan.

Temporal constraints in the form of Timed Initial Literals (TILs) (Gerevini et al. 2004) add expressive power to the planning language by constraining the duration of certain actions. In space domains TILs are useful in specifying resource constraints such as crew availability or loss of signal for downlinking data.

Plan Viability Checker

Acting in an uncertain world requires a combination of expressive planning models and capabilities for robust plan execution. In this effort we selected an approach to robustness based on execution monitoring and re-planning. A viable alternative that has been explored extensively in the literature is to incorporate the property of robustness into the plan itself through the representation of temporal flexibility and uncontrollable events (Morris, Muscettola, and Vidal 2001). We chose here to limit the expressive power of the plan model in favor of robust execution for the sake of faster planning; we are aware of the potential limitations of this approach and are interested in exploring hybrid approaches to ensuring robustness in future work.

Execution monitoring is a capability for tracking the evolution of the world state while a plan is executed. Systems robust to plan failure should respond to unpredicted changes to the world state that cause a plan to fail by triggering corrective actions. The Plan Viability Checker (PVC) is an execution monitoring system that continuously checks whether a partially executed plan is viable given changes to the world state (represented in the ROSPlan Knowledge Base). It is a similar, although simplified, capability to the one described in (Muise, Beck, and McIlraith 2013).

The PVC handles two kinds of unexpected change: in the duration of events (actions takes longer than expected) and in the truth values of fluents that are preconditions for future actions. Change in duration requires an ability to propagate the effects of the change into the future to determine whether a temporal constraint will be violated. Temporal constraints can either be with respect to points in time (e.g. deadlines) or intervals of time (a period where certain actions cannot be performed), such as loss of downlink.

The PVC operates in parallel with (an enhancement of) the ROSPlan dispatcher of Esterel plans. The Esterel plan graph consists of a set of action nodes. An action node is dispatched when all the preceding nodes in the graph are completed. Furthermore, an action start node completes once the action is dispatched, while a action end node completes when the execution of the action returns success by the action components.

The PVC is called to check to see if the plan is still viable after an action succeeds. PVC requires a limited ability to simulate the effects of changes to the world state in the future to determine whether an executing plan remains valid despite unplanned changes in world state. To do this, PVC creates and maintains a 'shadow knowledge base' (shadow-KB), initialized to the current world state. It simulates the execution of the remainder of the plan, in order, updating the shadow-KB to reflect the effects of the actions, determining whether action preconditions and overall conditions of durative actions hold for future actions. A plan is viable if the effect of the simulation is that the plan succeeds. Otherwise, the plan fails and the RPM is called to replan.

A feature provided by the enhanced dispatcher and the PVC is the ability to shift the timeline due to violated temporal constraints with respect to the original dispatch time of

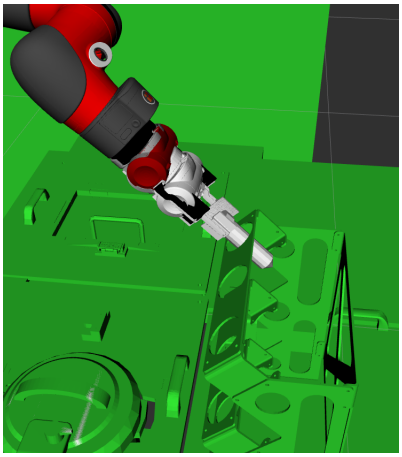

Figure 4: RVIZ visualization of robot executing the pick of a test tube.

an action as defined by the plan. The PVC will check to see if the failing precondition is a timed initial literal and will attempt to simulate the plan as if the action begins at the TIL application time, shifting all other actions in the plan as well. If the PVC is successful, RPM can avoid full replanning.

Integration with Low Level Behaviors

To integrate deliberative task-planning with low-level behaviors using ROSPlan, The ROSPlan Action Interface was designed to map actions in the PDDL domain model to low-level components. When ROSPlan dispatches an action to a component, the component will invoke the behavior required for completing the action. Upon successful completion of the behavior, the ROSPlan Knowledge Base is updated, with the effects of the action, using sensor data as verification prior to updating as applicable.

A number of different modeling languages for behaviors was employed for this domain. FlexBE (Schillinger 2017), was used as a modeling tool and behavior engine for manipulating the freezer door. At the lowest level, an action like opening the freezer door is mapped to a Finite State Machine of atomic commands like 'release gripper'. Branching in the FSM allowed for a degree of modeling failed actions or intermediate states like a partially closed door. Complimentary behaviors (opening, closing door) are arranged in a hierarchy of behavior meta-patterns that are managed by a mediator that decides which patterns to invoke.

Dispatching a fundamental manipulation action such as 'pick' or 'place' requires solving a motion planning problem. To facilitate the integration of task and motion planning, the *Affordance Template* framework was used (Hart, Dinh, and Hambuchen 2015). An affordance template is a graphical representation that exists in a 3D immersive environment (such as RViz) to provide robot task goals (including spatial end-effector way- points, contact points for interaction, force magnitudes) and parameters (object scales, locations) in object-centric coordinate frames. If a template is placed in such an environment (a location that is said to afford the task behavior) with the associated goals and parameters, they can be sent to the robot's control system to perform the task (see (Azimi, Zemler, and Morris 2019) for a more detailed description of how Affordance Templates were used in this system). Figure 4 shows an RVIZ visualization of a pick action with the Affordance Template framework.

Demonstration Platform Design

The demonstration platform used for these experiments was an enhancement of the one described in (Azimi, Zemler, and Morris 2019), and so will be only summarized here. The platform was inspired by existing scientific facilities aboard the International Space Station (ISS). The use case scenario specifies that the robot is attached to an ISS EXPRESS rack and interacts with rack payloads. Since an ISS EXPRESS rack is effectively just a cabinet frame with rack mount rails Figure 5), we opted to build an equivalent sized frame using aluminum T-slot rails rather than acquire an expensive duplicate rack.

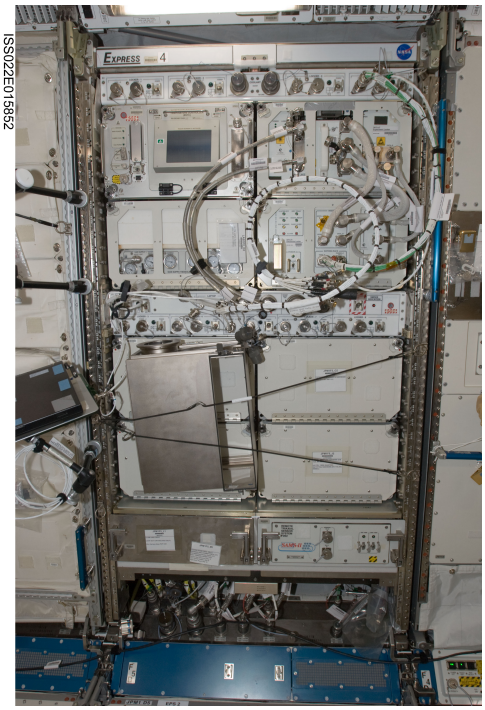

Figure 5: ISS Express Rack

Four standard size rack-mount "payloads" were then designed to fit within the frame: (1) A mechanical mock-up freezer modelled after the MERLIN freezer used on ISS, (2) a mock up centrifuge with a twist-lock lid that is simulated by a modified kitchen appliance; (3) a combination test tube rack and (functional) digital microscope; and (4) a mock-up of a set of fire ports to be used for inspecting with a fire probe. Figure 6 summarizes the platform configuration.

In order to know the state of the world without having to rely on a perception suite, the apparatus was augmented with

limit switches placed at various locations in the experiment area. These switches map to discrete on or off signals when pressed. In our scenario, switches were located at positions that allowed us to determine the state of fluents related to the states of the centrifuge and freezer. In the case of the centrifuge, one switch is triggered when the lid is at its open position, while a different switch is triggered when the lid is at its closed position. The freezer's switches were deployed in a similar way. Having these switches allowed us the additional functionality of changing the state of fluents in the domain in a way that is unexpected to both the planner and robot.

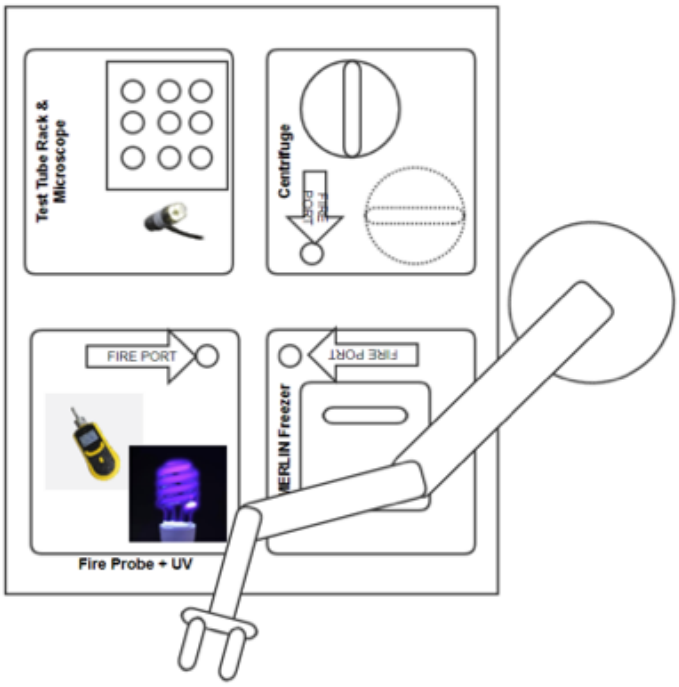

Figure 6: Drawing of ISS Express Rack Design with freezer, centrifuge,storage areas, and fire ports for inspection.

Experiments and Lessons Learned

The goals of the experiments were to enhance the capabilities of the system described in (Azimi, Zemler, and Morris 2019), using much of the same science scenario, in three ways:

- Adding goal preferences and the ability to plan based on goal priorities;
- Illustrating robustness to time delay through executing monitoring and replanning; and
- Illustrating smart autonomous response to potential threats.

In the nominal scenario, science imaging goals were defined for six test tubes using different set-up steps: all required removal from freezer and defrosted before taking

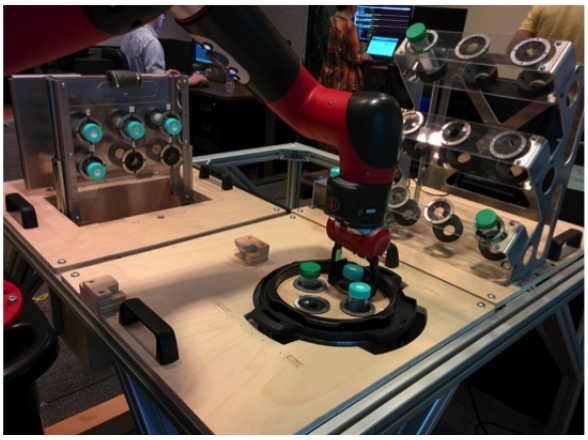

Figure 7: Sawyer Robot Executing PDDL Formulated Science Goals

an image; some were imaged without first being spun in a centrifuge; others were imaged after spinning. Finally, for added variety, still others were first exposed to UV before imaging. Each goal was assigned a priority of small, medium, or high. The nominal scenario schedules a maximum set of high-priority goals using the approach described above.

During execution, the nominal scenario was modified in one of three ways: either to increase the duration of an imaging action to delay the execution of the plan; an unexpected state change of a fluent; or to randomly trigger a fire alarm event.

Injecting delay into the executing plan required the system to determine correctly whether the remaining plan will eventually fail as a result of violation of temporal constraints; and, if failure is detected, trigger re-planning actions that adheres to goal priority ordering.

An unexpected state change was triggered by changing the state of an object that interacts with the limit switches on the ISS Express Rack mockup. This state change will get propagated to the Knowledge Base, which the PVC uses. If the unexpected state change violates any proceeding action in the plan, the PVC detects the failure, and triggers a re-planning

To test threat response, at a random time during plan execution, a fire alarm signal will trigger. The goal is for the system to cancel the plan in progress; finish the current robotic behavior to get to a predefined known state by performing a sequence of make-safe commands (using a finite state machine at the behavior level); and re-plan with immediate priority goal "find fire", where an immediate goal is one that super-cedes all other goals and is never planned for with other goals. The resulting plan will include actions for putting down any held test tube in a safe spot, and closing freezer door if it is open.

The scenario was successfully demonstrated on the robotic platform described above, using ROSPlan for task planning and execution. With proper tuning of input parameters, optimal plans (based on makespan) could be generated

by LPG in a matter of a few seconds, Plan lengths were in the range of a few dozen, and the plan model contained roughly 10 durative actions and 50 predicates.

Performance Analysis and Lessons Learned

Due to the nature of the described system, there is inherit flexibility regarding the inputs of our problem. For example, we could add multiple loss of signal (LOS) intervals that would prevent the image from transmitting from station to ground. We also varied the initial starting state of the items in our experiment, such as the centrifuge open versus closed or the initial positions of test tubes. The robot operators would adjust the input parameters to the system to test different aspects of the robotic operation, such as modifying and reducing the overall operations deadline to force the Goal Manager to reduce goals, adding a LOS during execution time (not planning time) to see if the enhanced esterel dispatcher and PVC would shift actions until the acquisition of signal (AOS), or the operator would move the centrifuge lid from open to closed during execution to see if the PVC recognizes the unexpected state change and force a replan. The emergent actions required to locate the "fire" required more complex robotic behavior modeling, as the desire was for the robot to stop execution in the quickest manner resulting in a known configuration. This required vast testing to ensure the success of the behavior model. Throughout all the testing, we ran the robotic system upwards of 100 times to the expected completion based on the varying conditions.

The operators, through the Goal Manager, would configure the task planner to solve up to six goals with a model containing 29 actions, with a potential to reduce to a total of two goals, depending on the solvability of all the goals based on the configuration of other inputs, such as an overall operations deadline. The planning time with the GM is dependent on the settings for the planner being used. Ultimately, to reduce to the highest priority two goals, the GM would have three planning cycles. For our settings with LPG-td (Gerevini and Serina 2008), we specified that the planner use a cpu time of 45 seconds, for a total of up to 2:15 minutes of planning time. For locating the "fire", the GM would invoke the immediate goals, and the robot would proceed to execute the plan, resulting in approximately 10 seconds of planning. For the GM, the ability to customize the planner command depending on whether the goal is an immediate goal versus a different priority level was necessary.

The Plan Viability Checker was invoked prior to the dispatch of every action. As the PVC creates a copy of the knowledge base, we can determine as soon as possible a failure that is expected to occur in the future, triggering a replanning cycle to attempt to recover from the event. The system can begin the recovery plan in as little as 45 seconds (based on the settings of the task planner). Future improvements can include invoking the PVC during the execution of actions to parallelize the process.

Although the tests conducted here consistently resulted in a response time for the deliberation system (planner and execution) that is consistent with possible operational requirements, more work is required to further quantify the the results on a range of realistic scenarios.

Summary and Future Work

Building an effective manipulator that combines deliberative and reactive behaviors requires a careful blend of plan modeling techniques; automated planner and plan representation; plan dispatching and execution; and goal reasoning and high level mission management. It also requires a framework for integrating deliberative models and decision-makers with lower level models and behaviors.

Future deep space assets, such as NASA's Gateway, will involve "cutting-edge robotics and computers [that] will operate experiments inside and outside the spaceship, automatically returning data back to Earth" (Gateway 2018). Future work includes generalizing the approach to integrating task planning with robotic manipulation to a multi-robot setting. We are seeking to determine whether the multi-agent planning framework (Stolba and Komenda 2017) could be use to characterize and plan for tasks that need to be coordinated for multiple agents (both human and robotic).

Future space explorers will combine smart robotic assets with automated vehicle system management and crew supervision. The RPM and Goal manager system components provide the basis for integrating human and machine activities, but much more work is needed in all aspects of human-robot collaboration, including knowledge sharing, behavior and intent prediction, goal sharing and negotiation, and trust building (Mainprice and Berenson 2013).

This paper has described a successful technology demonstration of integrated task planning for a in-space intra-vehicle activity application of a robot manipulator. The approach combines task planning and execution using PDDL planning, augmented with capabilities for enhanced optimized planning and robust execution.

Acknowlegements

This work was performed under NASA's Autonomous Systems and Operations (ASO) Project. The authors would like to thank team members Mike Goza, Alex Hansen, and Michael Park for their contributions to this effort. We also thank the anonymous reviewers for helpful feedback.

References

Azimi, S.; Zemler, E.; and Morris, R. A. 2019. Autonomous robotics manipulation for in-space intra-vehicle activity. In *Workshop on Planning and Robotics (PlanRob)*.

Bagnell, J. a. 2012. An Integrated System for Autonomous Robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2955–2962.

Berry, G. 2000. The foundations of esterel. In Plotkin, G. D.; Stirling, C.; and Tofte, M., eds., *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, 425–454. The MIT Press.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Huros, N.; and Carreras, M. 2015. Rosplan: Planning in the robot operating system. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, 333–341. ICAPS.

Cashmore, M.; Cimatti, A.; Magazzeni, D.; Micheli, A.; and Zehtabi, P. 2017. Robustness envelopes for temporal plans. *Proceedings of the AAAI Conference on Artificial Intelligence* 33:7538–3545.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *ICAPS*, volume 6, 212–221.

Gateway. 2018. *Q&A: NASA's New Spaceship*. `https://www.nasa.gov/feature/questions-nasas-new-spaceship`.

Gerevini, A., and Long, D. 2005. Plan constraints and preferences in pddl3 - the language of the fifth international planning competition. Technical report.

Gerevini, A., and Serina, I. 2002. Lpg: A planner based on local search for planning graphs with action costs. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*, AIPS'02, 13–22. AAAI Press.

Gerevini, A., and Serina, A. S. I. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence* 172(8-9):899–944.

Gerevini, A.; Saetti, A.; Serina, I.; and Toninelli, P. 2004. Lpg-td: a fully automated planner for pddl2.2 domains. In *In Proc. of the 14th Int. Conference on Automated Planning and Scheduling (ICAPS-04) International Planning Competition abstracts*.

Hart, S.; Dinh, P.; and Hambuchen, K. A. 2015. The affordance template ros package for robot task programming. *2015 IEEE International Conference on Robotics and Automation (ICRA)* 6227–6234.

Huang, X.; Hong, S.; Hofmann, A.; and Williams, B. C. 2019. Online risk-bounded motion planning for autonomous vehicles in dynamic environments.

Ingrand, F., and Ghallab, M. 2017. Deliberation for autonomous robots. *Artif. Intell.* 247(C):10–44.

Kaelbling, L. P., and Lozano-Perez, T. 2011. Hierarchical task and motion planning in the now. In *IEEE Conference on Robotics and Automation (ICRA)*. Finalist, Best Manipulation Paper Award.

Kim, P.; Williams, B. C.; and Abramson, M. 2001. Executing reactive, model-based programs through graph-based temporal planning. In *IJCAI*, 487–493.

Mainprice, J., and Berenson, D. 2013. Human-robot collaborative manipulation planning using early prediction of human motion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 299–306.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'01, 494–499. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2013. Flexible execution of partial order plans with temporal constraints. *International Joint Conference on Artificial Intelligence* 2328–2335.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence (AIJ)* 76(1-2):427–454. Special Issue on Planning and Scheduling.

Pettersson, O. 2005. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems* 53(2):73–88.

Roberts, M.; Shivashankar, V.; Alford, R.; Leece, M.; Gupta, S. K.; and Aha, D. W. 2016. Goal reasoning, planning, and acting with actorsim, the actor simulator.

Schillinger, P. 2017. *FlexBE Behavior Engine*. `http://philserver.bplaced.net/fbe/index.php`.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; and Abbeel, P. 2014. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Stolba, M., and Komenda, A. 2017. The madla planner: Multi-agent planning by combination of distributed and local heuristic search. *Artificial Intelligence* 252.

Vattam, S.; Klenk, M.; Molineaux, M.; and Aha, D. W. 2013. Breadth of approaches to goal reasoning: A research survey. Technical report, Naval Research Lab Washington DC.

Voudouris, C., and Tsang, E. 2000. Partial constraint satisfaction problems and guided local search.