IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 July 2023, accepted 20 August 2023, date of publication 25 August 2023, date of current version 30 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3308619

RESEARCH ARTICLE

Toward Autonomous Robotic Minimally Invasive Surgery: A Hybrid Framework Combining Task-Motion Planning and Dynamic Behavior Trees

KHUSNIDDIN FOZILOV[1], **JACINTO COLAN**[1], **(Member, IEEE), KOSUKE SEKIYAMA**[2], **AND YASUHISA HASEGAWA**[1], **(Member, IEEE)**
[1]Department of Micro-Nano Mechanical Science and Engineering, Nagoya University, Furo-cho, Nagoya, Aichi 464-8603, Japan
[2]Department of Mechatronics Engineering, Graduate School of Science and Engineering, Meijo University, Nagoya 468-8502, Japan

Corresponding author: Khusniddin Fozilov (khusniddin@mein.nagoya-u.ac.jp)

This work was supported by the Japan Science and Technology Agency (JST) CREST, Japan, under Grant JPMJCR20D5.

ABSTRACT The growing need for high levels of autonomy in Autonomous Robotic Surgery Systems (ARSS) calls for innovative approaches to reduce surgeons' cognitive load, optimize hospital workflows, and ensure efficient task-level reasoning and adaptation during execution. This paper presents a novel hybrid framework that synergistically combines Task-Motion Planning and Dynamic Behavior Trees for ARSS in Minimally Invasive Surgery. Our approach is designed to address the challenges of coordinating multiple surgical tools within a small workspace, thereby making complex surgical tasks like multi-throw suturing feasible and efficient. Through an extensive evaluation in simulation across diverse initial conditions and noise scenarios, the proposed method demonstrates improved success rates, reduced execution times, and fewer regrasps compared to standalone approaches. Furthermore, it showcases robustness under increased noise conditions. By applying our framework to a complex multi-throw suturing task, we illustrate its capability to seamlessly handle comprehensive suturing tasks, including needle picking, insertion, extraction, and the handover of the needle between Patient Side Manipulators. The results suggest that our hybrid approach not only enhances ARSS autonomy but also adapts effectively to unexpected environmental changes, laying the groundwork for its potential applicability in real-world surgical robotics.

INDEX TERMS Autonomous systems, behavior trees, hierarchical deliberation, minimally invasive surgery, multi-throw suturing, nonlinear optimization, robotic surgery, task and motion planning.

I. INTRODUCTION

Advancements in Robotic Surgical Systems have transformed modern healthcare, offering enhanced precision, stability, and reduced invasiveness of surgical operations. Traditionally, these systems operate in a master-slave configuration or rely heavily on human supervision. The emergence of Autonomous Robotic Surgical Systems (ARSS) holds promise for further optimizing surgical procedures by reducing cognitive load on surgeons, streamlining hospital workflows, decreasing surgical intervention times, and accelerating patient recovery [1], [2], [3], [4].

While RSS have made significant strides, they primarily operate at level 0-1 autonomy with complex task automation largely restricted to surgical subtasks such as suturing [5], [6], [7], [8], [9], [10], knot tying [10], tissue manipulation [11], [12], [13], and needle manipulation [14], [15], [16], [17], [18], [19], [20]. To ascend to level 2 autonomy and beyond, ARSS need to display deliberative capabilities, encompassing goal reasoning, environment monitoring through sensors, and

The associate editor coordinating the review of this manuscript and approving it for publication was Jingang Jiang.

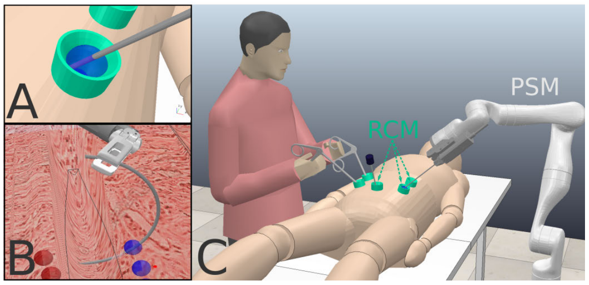


FIGURE 1. MIS task scenario depicting a PSM and a surgeon performing a suturing task: (a) represents the trocar to which the tools and PSM are typically attached; (b) corresponds to the possible view from the endoscopic camera; (c) illustrates the overall view with a potential layout demonstrating the positioning of the surgeon and PSM with respect to the patient during the surgery process.

adaptive re-planning when observed behaviors diverge from expected outcomes [3]. This requirement extends to the need for these deliberative functions to be interpretable for safer and more reliable surgeon-system interactions.

Dynamic surgical environments further complicate the design of ARSS. Current limitations of autonomous systems can be traced to the separation of planning and execution modules [1], [21], [22], [23]. As surgical conditions constantly change, the necessity for hierarchical deliberation and continual online planning and reasoning is underscored. Precomputed plans require dynamic adjustments, and surgical procedures must be monitored continuously to adapt strategies effectively at run-time [1], [22].

Moreover, specific surgical scenarios such as Minimally Invasive Surgery (MIS) tasks present additional challenges, including the operation of multiple tools in a confined workspace (Figure 1(c)). These intricate procedures, like suturing, are performed within a constrained workspace, heightened by the need to precisely control high Degree of Freedom (DOF) arms placed at the trocar or the incision point (Figure 1(a)), which is formulated as Remote Center of Motion (RCM). Furthermore, the cable-driven tools are commonly used, which necessitates fine-tuned calibration, adding another layer of complexity to precise control [24], [25], [26]. To address these complexities, ARSS must effectively combine task-level and motion-level reasoning. Task-level reasoning decides when and how to use available Patient Side Manipulators (PSMs), while motion-level reasoning generates safe trajectories for these manipulators. This coordination, essentially a hybrid search problem, can be effectively managed using Task and Motion Planning (TMP) [27], [28]. TMP has demonstrated success in tasks such as manipulation and rearrangement planning [29], [30], [31], [32], [33].

Despite the proven effectiveness of TMP, traditional approaches often treat task planning and motion planning as separate problems, which can result in suboptimal, inefficient, or unsafe plans. TMP seeks to overcome these issues by integrating task-level and motion-level constraints and objectives, resulting in safer and more efficient plans. Optimal performance is achieved through efficient sampling techniques and accurate domain descriptions [34]. Any inaccuracies can lead to suboptimal plans and safety risks, emphasizing the importance of continually refining and updating these models based on execution process feedback.

The aim of this study is to address the limitations of TMP and meet the challenges of continual online planning. We present a novel approach for achieving higher levels of autonomy in robotic surgery through a hybrid integration of TMP and Behavior Trees (BT) [35]. Our method includes considerations for a multi-arm setup in the MIS context and addresses specific MIS challenges like RCM constraints, collision and inverse kinematics (IK) samplers, for safe execution. In addition to continual action monitoring and reactivity in the case of execution noises, BTs offer readable plans, aiding solution analysis and debugging. We demonstrate the effectiveness of our approach through a case study of a knot-less suturing task in a simulated minimally invasive surgery environment. Our method allows for the inclusion and analysis of multiple components within the suturing task such as - picking, orienting, insertion, extraction, and handover of the needle given an RCM and the chosen suturing location, while ensuring that constraints are met and collisions are avoided. Our study contributes to the field of ARSS for MIS tasks with the following additions:

1) We propose a hybrid framework that combines TMP with Dynamic Behavior Trees (DBT) for continual online planning and reasoning in response to real-time changes in the surgical environment.
2) We develop a TMP approach specifically tailored for suturing tasks in MIS, that incorporates an efficient IK sampler, collision-checking, and motion planning techniques. A key feature of our approach is the ability to coordinate two PSMs to perform a full suturing task. This task includes challenging subtasks, such as needle picking, suturing, and handover between the robots. We demonstrate this methodology through a four-throw suturing task.
3) We provide an open-source simulation environment and code implementation of our framework to foster further research and development in TMP methods for surgical scenarios and enhance the community's understanding and implementation of ARSS. The code is available at https://github.com/husikl/asar_hybrid_tmp.git.

The remainder of this paper is structured as follows: Section II reviews related works in the field of RSS and ARSS. Section III introduces our system architecture, outlining the approach for accomplishing a Multi-Throw Suturing (MTS) task using TMP. Section IV presents our proposed approach, detailing TMP for PSMs in MIS and the multi-grasp trajectory generator for the suturing task. The dynamic execution framework using behavior trees is elaborated in Section V. The evaluations and results are presented in Section VI. Finally, Section VII offers our concluding remarks.

II. RELATED WORKS

A. ROBOT AUTONOMY IN MIS

While there was a great breakthrough in surgical subtask automation, when it comes to MIS, the majority of the works focus on the collaborative execution paradigms (semi-autonomous). Furthermore, coordinating the multiple-arms in the MIS setting creates additional challenges to ensure the safety and robustness of the execution.

The requirements for MIS were introduced in [36], comparisons of different MIS methods [37], and the mental load on surgeons during these procedures [38], [39]. Furthermore, several studies have explored the challenge of automating complex tasks such as suturing [7], [8], [40], [41], stitch planning [41], needle relay [20], and endoscopic camera control [42], [43], [44], [45], [46], [47]. Research on preoperative planning strategies [48], [49], [50], [51] emphasizes the influence of configuration choice in achieving optimal surgical outcomes.

B. TASK AND MOTION LEVEL REASONING

Task and motion level reasoning is crucial in automating surgical tasks. Significant advancements have been made in several research areas; some have learned transitions in surgical scenarios [52], [53], [54], [55], while others have leveraged knowledge models to enhance robot skills [56]. Some approaches have modeled surgical tasks using statecharts [57], and others have evaluated the skills of surgeons [58]. Frameworks like the one proposed in [59] aim for autonomous needle insertion by employing a supervisory controller based on a Hidden Markov Model.

However, the majority of these works focus on high-level reasoning, assuming the presence of finely-tuned controllers or execution with model predictive control. Furthermore, these approaches do not consider the challenges related to RCM in their attempts for fully autonomous execution.

C. EXECUTION MODELING AND RECOVERY

When it comes to execution modeling and recovery, both Finite State Machines (FSM) and BTs have found their applications. FSMs, utilized for suturing [9] and knot-tying tasks [60], are known for their reusability and formal verifiability. On the other hand, BTs offer the advantage of dynamic reconfigurability, making them suitable for run-time execution adjustment or failure recovery [23], [61], [62].

Despite these advancements, challenges in the field of MIS persist. Handling multi-arm systems [50], [63], [64], [65], the necessity for tissue retraction [11], and addressing ethical issues [66] remain critical areas of concern. This study builds upon these foundational works, focusing on combining task-motion planning and behavior trees to achieve higher levels of autonomy in MIS.

III. PROPOSED SYSTEM OVERVIEW

The proposed hybrid framework, illustrated in Figure 2, comprises four main modules: the Task Manager, Motion Samplers, Behavior Manager, and the Robot's Module. Each component plays a unique role in ensuring the effective execution of surgical tasks.

The Robot's Module relates to all controllers and sensors that process raw information into symbolic representations and send them to the Motion Samplers. This module receives input from the Behavior Manager, which allows it to execute commands generated from the dynamic behavior tree.

The Task Manager, which serves as the higher-level planning module, is based on the PddlStream planner that integrates TMP. It generates a sequential plan using pre-operative information, an action library, and task description. This plan, which comprises a tuple of actions and trajectories, is verified for action constraints using available samplers. The validated plan is then transmitted to the Behavior Manager.

The Behavior Manager module utilizes the plan from the Task Manager to generate a DBT. The DBT is continuously monitored to detect conflicts and respond to external disturbances. It has the capability to re-execute the plan, for instance, re-grasping an object or re-sampling a new trajectory, or request a re-plan from the Task Manager if the BT cannot be updated due to unsatisfiable preconditions or infeasible trajectories. This process repeats until all actions in the sequence are successfully executed.

Significantly, both the Behavior Manager and the Task Manager can access the Motion Samplers. This dual access facilitates two types of execution: DBT-based and TMP-based DBT hybrid execution. The DBT-only based execution can generate trajectories and acquire samples of interest, such as placements, or check for collisions when necessary. On the other hand, the TMP-based DBT hybrid execution is used to generate the initial DBT as an alternative to building the DBT from the bottom up, which integrates optimality metrics into the DBT. In ideal scenarios, the DBT constructed from the TMP should be sufficient and require no further adjustments.

In cases of unexpected events, the DBT can be updated on the fly by directly communicating with the Motion Samplers, bypassing the potentially time-consuming TMP re-planning process. However, if the DBT cannot be further adjusted to achieve the goal, a re-planning request is sent to the Task Manager module for a new solution. Moreover, the framework could request human supervisor assistance in a takeover scenario, enabling a transition to remote control.

IV. TASK AND MOTION PLANNING

A. PddlStream

Our proposed framework employs PDDLStream [67], a specific variant of TMP methodology. PDDLStream enhances TMP by allowing the effects of actions to be dynamically determined by the output of streams, providing a higher degree of flexibility during the planning process. PDDLStream, akin to PDDL [68], uses predicate logic to articulate planning problems. It incorporates literals, facts, static and fluent literals, and states. In this context, literals

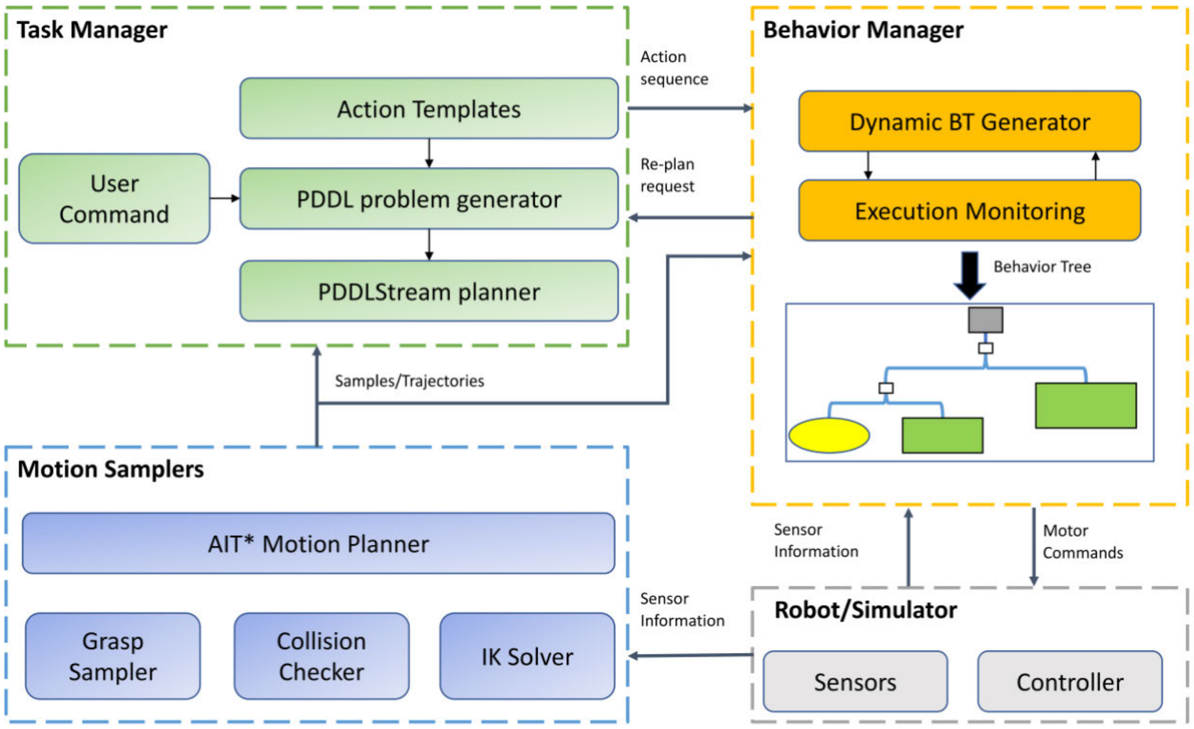


FIGURE 2. **Proposed system overview: Hybrid framework combining task-motion planning and dynamic behavior trees.**

are used to evaluate a predicate for a given set of arguments, facts employ literals to express a statement as true or false, static literals remain constant, and fluent literals change their truth value as actions are applied.

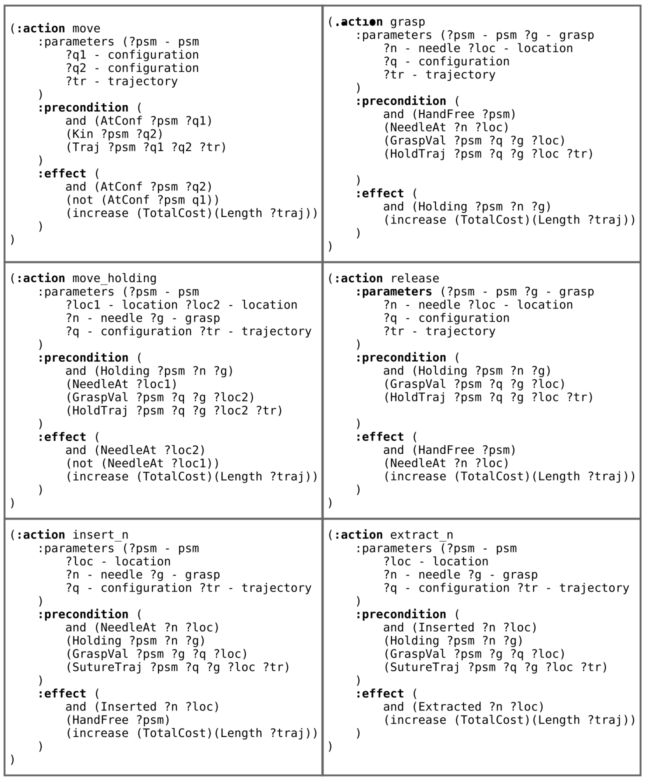
Similar to the Planning Domain Definition Language (PDDL) [68], PDDLStream uses predicate logic to articulate planning problems. It incorporates literals, facts, static and fluent literals, and states. In this context, literals are used to evaluate a predicate for a given set of arguments. Facts use literals to express a statement as true or false. Static literals remain constant, while fluent literals change their truth value as actions are applied. We adopt these conventions in our approach, defining actions with preconditions (conditions that must be satisfied for an action to be applied) and post-conditions (conditions that must hold true for an action to terminate).These conventions typically consist of a set of preconditions (**:pre**) that need to be satisfied before applying the action, an effect formula (**:eff**) that describes the transition between states, and a set of parameters (**:param**) that the action operates on. The effect formulas may alter fluent to be true, false (**not**), or increase the plan cost (**incr**).

Formally, an action a is characterized by its preconditions C^a_{pre} and post-conditions (also called as effects) C^a_{post}:

$$a := \left\{C^a_{\text{pre}}, C^a_{\text{post}}\right\}, \quad C^a_{\text{pre}} \subset C, \quad C^a_{\text{post}} \subset C$$

where C is a set of all conditions, and C_{pre} and C_{post} are sets of preconditions and post-conditions, respectively. A state s is a vector of conditions $C_i \in C$. An action $a \in A$ is applicable in a state s if it satisfies all a's preconditions. Applying a in s yields the state $s' = \{C^a_{\text{post}}\}$. The domain for the suturing task is represented using parameters such as `?psm` for an available PSM, `?n` for the 6DOF pose of a surgical needle, and `?loc` for the two entry/exit points on the tissue, among others. Fluent predicates model the evolving PSM configuration and end-effector status. The fluent predicates `AtConf`, `Holding`, `HandFree`, and `NeedleAt` model the changing robot configuration, end-effector status, and the needle positioning. `Inserted` and `Extracted` model the suturing progress given a needle and the suture location.

The static predicates `PSM`, `Conf`, `Grasp`, `GraspVal`, `Kin`, `Traj`, `HoldTraj`, and `SutureTraj` are constant facts. `PSM` declares which PSM from the available two is being used. `Conf` declares the robot's configuration. `Grasp` indicates a candidate grasp sample to apply an action with respect to the needle. `Kin` is a kinematic constraint that indicates the configuration is valid. `Traj` is a constraint that `?q1` and `?q2` are start and end configurations for the trajectory `?t`, which has to respect joint limits, keep all constraints, and avoid collisions. The `HoldTraj` and `SutureTraj` are compound trajectories for the placement of the needle in the environment and suturing motions respectively - the `HoldTraj` has an extra constraint for the grasp `?g` to be valid to achieve the movement, and the `SutureTraj` includes following the path for the needle given the grasp and suture location - that is, orientation and position constraints for suturing.



```
(:action move
    :parameters (?psm - psm
        ?q1 - configuration
        ?q2 - configuration
        ?tr - trajectory
    )
    :precondition (
        and (AtConf ?psm ?q1)
        (Kin ?psm ?q2)
        (Traj ?psm ?q1 ?q2 ?tr)
    )
    :effect (
        and (AtConf ?psm ?q2)
        (not (AtConf ?psm q1))
        (increase (TotalCost)(Length ?traj))
    )
)
```

```
(.actıon grasp
    :parameters (?psm - psm ?g - grasp
        ?n - needle ?loc - location
        ?q - configuration
        ?tr - trajectory
    )
    :precondition (
        and (HandFree ?psm)
        (NeedleAt ?n ?loc)
        (GraspVal ?psm ?q ?g ?loc)
        (HoldTraj ?psm ?q ?g ?loc ?tr)

    )
    :effect (
        and (Holding ?psm ?n ?g)
        (increase (TotalCost)(Length ?traj))
    )
)
```

```
(:action move_holding
    :parameters (?psm - psm
        ?loc1 - location ?loc2 - location
        ?n - needle ?g - grasp
        ?q - configuration ?tr - trajectory
    )
    :precondition (
        and (Holding ?psm ?n ?g)
        (NeedleAt ?loc1)
        (GraspVal ?psm ?q ?g ?loc2)
        (HoldTraj ?psm ?q ?g ?loc2 ?tr)
    )
    :effect (
        and (NeedleAt ?loc2)
        (not (NeedleAt ?loc1))
        (increase (TotalCost)(Length ?traj))
    )
)
```

```
(:action release
    :parameters (?psm - psm ?g - grasp
        ?n - needle ?loc - location
        ?q - configuration
        ?tr - trajectory
    )
    :precondition (
        and (Holding ?psm ?n ?g)
        (GraspVal ?psm ?q ?g ?loc)
        (HoldTraj ?psm ?q ?g ?loc ?tr)

    )
    :effect (
        and (HandFree ?psm)
        (NeedleAt ?n ?loc)
        (increase (TotalCost)(Length ?traj))
    )
)
```

```
(:action insert_n
    :parameters (?psm - psm
        ?loc - location
        ?n - needle ?g - grasp
        ?q - configuration ?tr - trajectory
    )
    :precondition (
        and (NeedleAt ?n ?loc)
        (Holding ?psm ?n ?g)
        (GraspVal ?psm ?g ?q ?loc)
        (SutureTraj ?psm ?q ?g ?loc ?tr)
    )
    :effect (
        and (Inserted ?n ?loc)
        (HandFree ?psm)
        (increase (TotalCost)(Length ?traj))
    )
)
```

```
(:action extract_n
    :parameters (?psm - psm
        ?loc - location
        ?n - needle ?g - grasp
        ?q - configuration ?tr - trajectory
    )
    :precondition (
        and (Inserted ?n ?loc)
        (Holding ?psm ?n ?g)
        (GraspVal ?psm ?g ?q ?loc)
        (SutureTraj ?psm ?q ?g ?loc ?tr)
    )
    :effect (
        and (Extracted ?n ?loc)
        (increase (TotalCost)(Length ?traj))
    )
)
```

FIGURE 3. PDDL based action templates definitions for suturing task.

We define six actions - move, move holding, grasp, release, insert, extract - as shown in Figure 3

PddlStream also employs *streams*, which are functions that take inputs (**:inp**) and generate a potentially infinite sequence of outputs (**:out**). For a suturing task, we sample possible configurations by checking inverse kinematics given the grasp `?g` and location `?loc` using the `s-ik` stream, and `s-grasp` to sample grasps `?g` given the needle at location `?loc`, as defined in Figure 4.

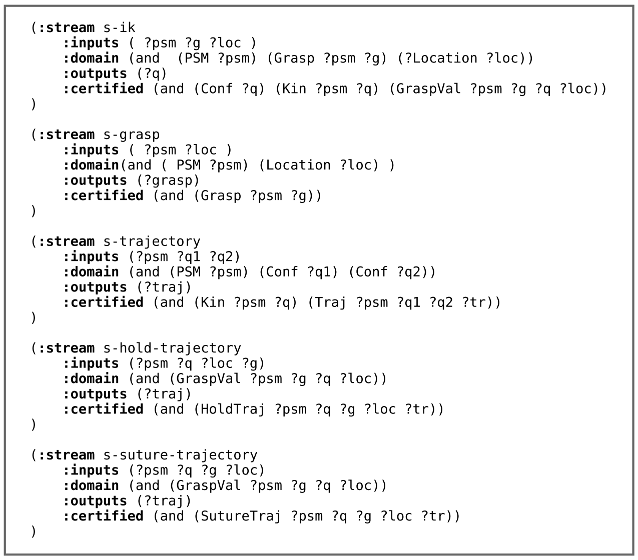
For the suturing task, we formulate a compound cost function that considers both the number of regrasps and the total trajectory length. This cost function is used within the PDDLStream framework to compute the optimal plan. The formulation is as follows:

$$\pi^* = \underset{\pi\in\Pi}{\operatorname{argmin}}\left(\sum_{a\in\pi}\alpha\cdot\mathrm{Cost}(a)+\beta\cdot\mathrm{Grasps}(a)\right) \quad (1)$$

Here, Cost(a) is the cost of applying action a (associated with the trajectory length `Length`), Grasps(a) signifies the number of regrasps involved in action a, and α and β are weighting factors balancing the two components of the cost function.

B. MOTION SAMPLERS

This section focuses on the motion samplers implemented for MTS task in TMP. The performance and scalability of TMP methods rely heavily on the quality and speed of obtaining the samples. The section describes grasp samplers



```
(:stream s-ik
    :inputs ( ?psm ?g ?loc )
    :domain (and  (PSM ?psm) (Grasp ?psm ?g) (?Location ?loc))
    :outputs (?q)
    :certified (and (Conf ?q) (Kin ?psm ?q) (GraspVal ?psm ?g ?q ?loc))
)

(:stream s-grasp
    :inputs ( ?psm ?loc )
    :domain(and ( PSM ?psm) (Location ?loc) )
    :outputs (?grasp)
    :certified (and (Grasp ?psm ?g))
)

(:stream s-trajectory
    :inputs (?psm ?q1 ?q2)
    :domain (and (PSM ?psm) (Conf ?q1) (Conf ?q2))
    :outputs (?traj)
    :certified (and (Kin ?psm ?q) (Traj ?psm ?q1 ?q2 ?tr))
)

(:stream s-hold-trajectory
    :inputs (?psm ?q ?loc ?g)
    :domain (and (GraspVal ?psm ?g ?q ?loc))
    :outputs (?traj)
    :certified (and (HoldTraj ?psm ?q ?g ?loc ?tr))
)

(:stream s-suture-trajectory
    :inputs (?psm ?q ?g ?loc)
    :domain (and (GraspVal ?psm ?g ?q ?loc))
    :outputs (?traj)
    :certified (and (SutureTraj ?psm ?q ?g ?loc ?tr))
)
```

FIGURE 4. Streams definitions for suturing task.

for surgical needle manipulation, an optimization-based IK solver that considers RCM constraints and joint limits, and a motion planner using AIT* that generates pick-up, insertion/extraction, and re-alignment trajectories.

1) GRASP SAMPLING

We consider a semicircle shape for the needle, which is the most common in MIS. The equation for the semicircle suture needle in the needle frame can be defined as:

$$\begin{cases} x_n = 0 \\ y_n = R\cos\alpha \\ z_n = R\sin\alpha \end{cases} \quad (2)$$

where R is the radius of the needle and $\alpha \in [\frac{\pi}{2}, \frac{3\pi}{2}]$ is the angle on the needle. By randomly sampling an α, a grasping point on the suture needle can be sampled. The pose of this grasping point in the needle frame is denoted as $\left[p^n_g \; q^n_g\right]$, where ${p^n}_g = [x_n \; y_n \; z_n]^T$ and is calculated by equation (1). Then $q^n_g = [1\;0\;0\;0]^T$ is the quaternion to represent initial orientation.

A spherical coordinate system is defined with the origin at the grasping point, referred to as the grasping point frame, to sample a grasping direction pointing to a grasping point for initialization. A point using Cartesian representation in this frame can be calculated as

$$\begin{cases} x_g = d\cos\phi \\ y_g = d\sin\phi\cos\theta \\ z_g = d\sin\phi\sin\theta \end{cases} \quad (3)$$

where (d, θ, ϕ) are the radius, azimuth, and inclination respectively from the grasping point frame. The depth of the grasp with regards to the gripper is defined by d, θ and ϕ give the grasping angle relative to the needle.

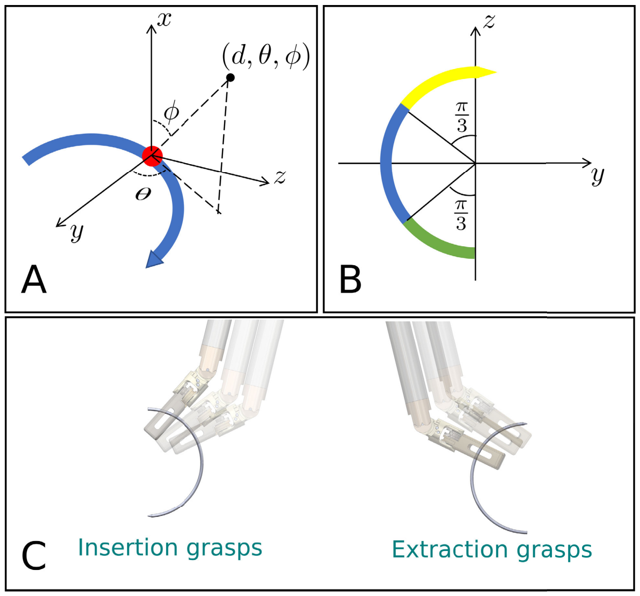


FIGURE 5. This figure provides an illustration of grasp sampling on a semi-circular needle. Subfigure (a) demonstrates the use of spherical coordinates to sample the grasping position. In subfigure (b), the sampling regions on the needle are depicted, where yellow indicates extraction samples and green denotes insertion. Subfigure (c) presents example grasps for both the insertion and extraction actions.

To divide the semicircle into three equal parts, then each part would have an angle of $\frac{\pi}{3}$. To sample grasping points from each part of the semicircle, we would need to adjust the range of α accordingly. Since α is the angle on the needle, we can adjust the range of α to cover each third of the semicircle as follows:

For the first third: $\alpha \in [\frac{\pi}{2}, \frac{5\pi}{6}]$ For the second third: $\alpha \in [\frac{5\pi}{6}, \frac{7\pi}{6}]$ For the third third: $\alpha \in [\frac{7\pi}{6}, \frac{3\pi}{2}]$ These ranges cover the entire semicircle and divide it into three equal parts, allowing us to sample grasping points from each part.

By randomly sampling a (d, θ, ϕ), the target position of the end-effector will be set to $p_e^g = [x_g, y_g, z_g]^T$, and its orientation will become q_e^g such that the gripper points from $[x_g, y_g, z_g]^T$ to the origin in the grasping point frame. Next, (p_e^g, q_e^g) are transformed to the needle frame by:

$$H_e^n(p_n^g, q_n^g) = H_g^n(p_g^n, q_g^n)H_e^g(p_e^g, q_e^g) \tag{4}$$

where $H(\cdot, \cdot) \in SE(3)$ is the homogeneous representation of a pose. Then the end-effector can be set to reach (p_e^n, q_e^n), hence grasping the needle, through inverse kinematics. The goal grasping direction can be set similarly. In the following sections, the end-effector that is initialized to hold a needle is referred to as the grasping end-effector, and the one that approaches the goal is referred to as the regrasping end-effector.

Given the sampled grasping poses, the batch of samples is evaluated based on manipulability at the grasped pose. Manipulability, represented by the determinant of the manipulability Jacobian J, measures the robot's ability to move effectively in a specific direction. Specifically, manipulability is calculated as:

$$J_m = \sqrt{\det(JJ^T)} \tag{5}$$

In the context of MIS, the manipulator's degrees of freedom are constrained due to the RCM limitations, which may prevent us from reaching certain sampled grasped poses. Therefore, by prioritizing samples with high manipulability, we can plan grasps that are potentially reachable and useful to successfully perform the stitching task.

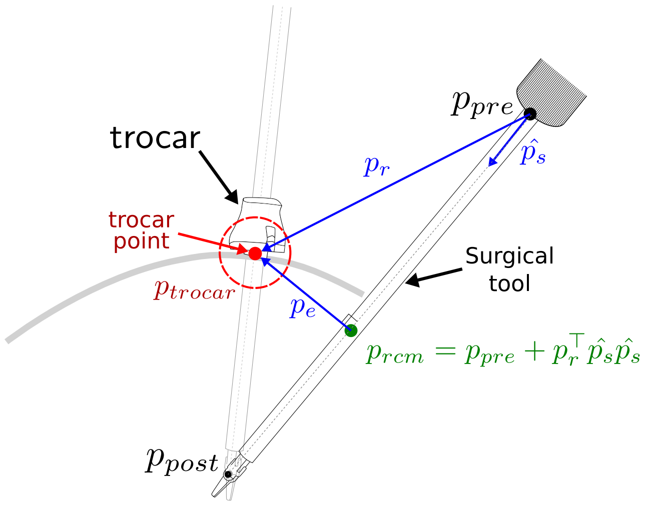


FIGURE 6. Characterization of the remote center of motion (RCM) [69].

2) NONLINEAR OPTIMIZATION IK SOLVER

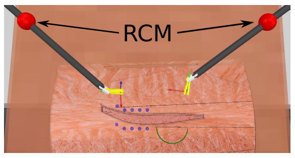
In this paper, we build upon the concept of concurrent inverse kinematic (IK) solving, which has been proposed to account for the unique constraints in MIS [8], [69], and extend its application to motion planning for PSMs.

In MIS applications, two main tasks are defined: the RCM constraint task for ensuring that the remote center of motion constraint is met, and the tool tip pose control task for accurate control of the surgical tool's position.

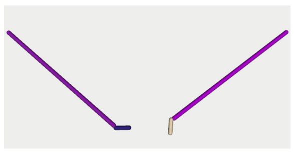
The RCM constraint is characterized in terms of the kinematic distance between the trocar point and the tool axis [8]. The positions of the joints before and after the location of the RCM can be defined as $p_{pre} \in \mathbb{R}^{3\times1}$ and $p_{post} \in \mathbb{R}^{3\times1}$, respectively. The nearest point on the tool axis to the trocar point is defined as $p_{rcm} \in \mathbb{R}^3$ and can be obtained as

$$p_{rcm} = p_{pre} + p_r^T \hat{p}_s \hat{p}_s, \tag{6}$$

where $\hat{p}_s = \frac{p_{post} - p_{pre}}{||p_{post} - p_{pre}||}$ indicates the direction of the surgical tool axis and $p_r = p_{trocar} - p_{pre}$ represents the difference between the position of the trocar point and the joint before the RCM. The vector $p_e = p_{trocar} - p_{rcm}$ denotes the vector to the trocar point p_{trocar} from its nearest point on the tool axis p_{rcm}.



(a) This figure depicts two PSMs with 3-DoF end effectors attached to the RCM.



(b) The simplified model, composed of four cylinders, is visualized in RVIZ [72].

FIGURE 7. A simplified collision model is used for collision checking: two cylinders represent the shafts, and two more represent the end effector fingers.

The RCM error e_{rcm} is defined as the minimum distance to the trocar point and can be calculated as

$$e_{rcm} = ||p_e|| = ||p_{trocar} - p_{rcm}|| \tag{7}$$

For a tool tip pose control task, we define the actual and desired end-effector pose as the transformations ${}^{B}T_{ee_{act}} \in SE(3)$ and ${}^{B}T_{ee_{des}} \in SE(3)$ of the robot end-effector frame with respect to an inertial frame B respectively. The 6D pose task error is then defined by

$$e_{ee(q)} = \log_6({}^{B}T_{ee_{des}} {}^{B}T^{T}_{ee_{act}}) \tag{8}$$

where the logarithm $\log_6 : SE(3) \rightarrow se(3)$ maps the pose from the Lie group $SE(3)$ to twists in the $se(3)$ [70].

The constrained inverse kinematics problem can be redefined with a nonlinear soft restricted optimization formulation, where a weight w_i is assigned to each ith task according to its priority. For the proposed nonlinear optimization IK solver, the RCM constraint task and the tool pose control task are given a weight coefficient w_1 and w_2 respectively, with $w_1 \gg w_2$. The optimization problem is defined as

$$\begin{aligned} \dot{q} = \min_{\dot{q}} \quad & w_1 e_{ee}^{T} e_{ee} + w_2 e_{rcm}^{T} e_{rcm} + w_3 \dot{q}^{T} \dot{q} \\ \text{s.t.} \quad & q^{-} \leq q + \dot{q}\delta t \leq q^{+} \end{aligned} \tag{9}$$

where q^{-} and q^{+} are the lower and upper joint limits, respectively, and δt represents a control cycle period. The last term $w_3 \dot{q}^{T} \dot{q}$ works as a regularizer, whereas the inequality constraint avoids exceeding the joint limits.

3) COLLISION CHECKING

Given the attachment of the PSMs to RCM constraints in our study, the workspace of each robot arm is naturally defined as a cone whose apex is at the RCM point. The mechanical constraints limit the movements of the robot arm to rotations around the RCM, delineating a conic operation space for both arms. This space can be further constrained by specifying the tissue location. To prevent potential collisions between the PSM tools, we employ a simplified model for the surgical tool, as depicted in Figure 7. It is conceived as two cylinders - the shaft, linking the end effector to the PSM, and the end effector itself, an abstraction suitable for an end effector with 3 degrees of freedom.

During the pose sampling process of the PSMs, if the IK solution is valid, we subsequently perform collision checking between the corresponding models of the surgical tools. Specifically, we inspect for collisions between the four cylinders, two representing each surgical tool. The computations for collision checking are based on the Flexible Collision Library [71]. It is important to note that the recovery process once a collision occurs, or collision avoidance with respect to dynamic elements such as the tissue itself, the surgical needle, or the human surgeon, falls outside the scope of this paper.

4) MOTION PLANNERS INTEGRATION

In this paper, we focus on two key motions in MIS: end-effector trajectory planning and needle trajectory planning. This process involves the efficient planning and execution of robotic movements to ensure precise and effective suturing during surgery. The end-effector trajectory planning component is tasked with the optimal positioning of the surgical tool for needle pickup and placement, while needle trajectory planning focuses on plotting the needle's accurate path of insertion and pull-through in the tissue. These combined elements involve several advanced techniques, such as the AIT* motion planner, and introduce a novel multi-grasp trajectory generation methodology to improve the procedure's flexibility and efficiency in various MIS scenarios.

*a: AIT**

The motion planner proposed in this study utilizes the AIT* planner [73], an almost-surely asymptotically optimal motion planner that operates on the basis of batch-sampling. The AIT* planner applies an asymmetric bidirectional search to derive and exploit an accurate heuristic for each individual problem instance. The primary focus of the AIT* planner is the movement of the PSM's end-effector. It employs a nonlinear optimization based IK solver to verify the validity of all potential states of the end-effector. If a sampled state fails to ascertain joint values for placing the end-effector within a specified timeout period, it is deemed as invalid. Every sample is subsequently inspected for potential collisions. Following a collision check, a validated motion plan undergoes a smoothing process based on B-spline techniques, finalizing the resulting trajectory.

b: MULTI-GRASP TRAJECTORY GENERATION

Traditional suturing methods typically employ a single grasp throughout the task, operating under the assumption that it permits the desired motion. However, in MIS scenarios, constraints can often limit the manipulability at the chosen grasping point, rendering the initial grasp only partially usable. Surgeons frequently adjust their grasp during the suturing process, considering the entry and exit points on the tissue, to enhance manipulability. Hence, the ability to maneuver the needle with various grasp options can

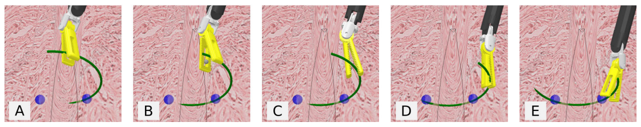


FIGURE 8. Insertion phase: (a) The needle is initially inserted, (b) a kinematic limit is reached, and the end effector opens to facilitate a new grasp, (c)-(d) a new grasp is applied, (e) the insertion is completed, and the needle is released.

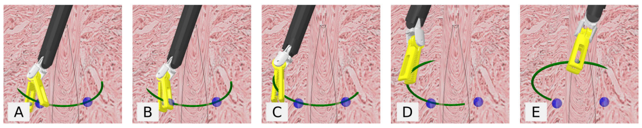


FIGURE 9. Extraction phase: (a) The needle is grasped, (b)-(d) the end effector rotates the needle in a circular motion, (e) the needle is fully extracted.

Algorithm 1 Multi-Grasp Trajectory Generation

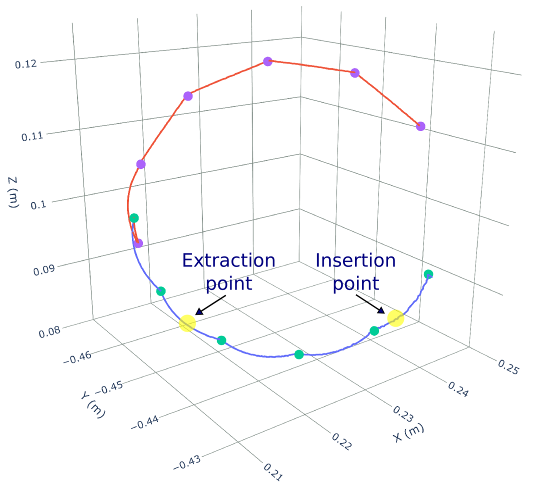
```
Input: $\mathcal{W} = \omega_1, \omega_2, \ldots, \omega_n$, $G = g_1, g_2, \ldots, g_m$
Output: $\mathcal{T} = (g_i, \tau_i)_{i=1}^{k}$
$\mathcal{T} \leftarrow \emptyset$
for $\omega_j \in \mathcal{W}$ do
    for $g_i \in G$ do
        if $Kin(g_i)$ then
            $\tau_i \leftarrow$ AIT*$(g_i, \omega_j)$
            if $\tau_i \neq fail$ then
                if $g_i \in \mathcal{T}$ then
                    $(g_i, \tau_i) \leftarrow \tau_k$
                else
                    $(g_k, \tau_k) \leftarrow \mathcal{T}$
if $|\mathcal{T}| = |\mathcal{W}|$ then
    return $\mathcal{T}$
return failure
```

significantly enhance the feasibility and efficiency of the suturing task in MIS scenarios.

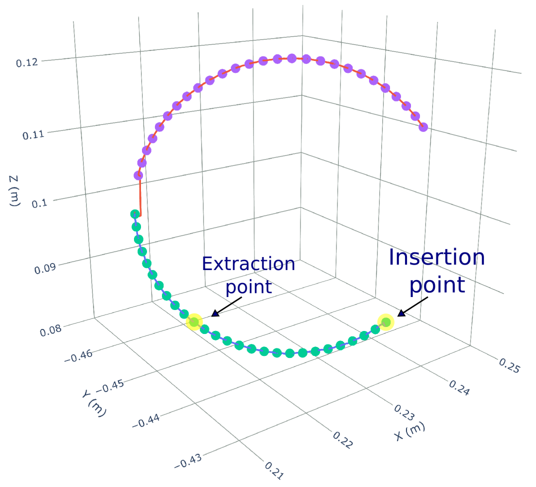
The multi-grasp trajectory generation method, outlined in Algorithm 1, uses the AIT* method to tackle these challenges. The algorithm starts with waypoints as input, which are determined based on the needle's insertion and extraction points, as well as its size. The waypoints are pose vectors representing continuous pose and orientation, assuming a circular path for the needle. These waypoints for Algorithm 1 can be computed similarly to the method presented in [5], which handles curvature-constrained motion planning issues using sequential convex optimization, or by estimating an optimal center point for the needle and generating a circular needle trajectory as in [6]. The larger the number of waypoints, the more constrained the PSM's motion is likely to be. Additionally, using waypoints as input also allows for replication of other needle motion patterns, such as elliptical curves.

Algorithm 1 initiates with a set of waypoints, denoted as $\mathcal{W}$, and a set of grasps, G. The objective is to generate a tuple of grasps and trajectories, denoted as $\mathcal{T} = (g_i, \tau_i)_{i=1}^{k}$, as its output. Iterating through each waypoint ω_j in $\mathcal{W}$, the algorithm processes each grasp g_i in G. It verifies the kinematic reachability of each grasp using the Kin function. If a grasp g_i is deemed reachable, the AIT* function is invoked to create a collision-free trajectory τ_i from this grasp to the waypoint ω_j. Upon successful trajectory generation τ_i, the algorithm determines whether this grasp g_i is associated with a pre-existing tuple t_k in $\mathcal{T}$. If so, the newly generated trajectory τ_i is appended to the existing trajectory associated with this grasp. If not, a new tuple $t_k = (g_i, \tau_i)$ is formed and integrated into $\mathcal{T}$. This process is repeated until a trajectory for each waypoint is generated, or all grasps within G have been assessed. The algorithm then returns the set of tuples, $\mathcal{T}$, each comprising a grasp and its corresponding trajectory. In case it fails to generate a trajectory for each waypoint, the algorithm returns a failure state. The general idea behind the multi-grasp based suturing is illustrated in Figure 8 and Figure 9. The robot initially attempts complete insertion using the first grasp as depicted in Figure 8 (a). Due to kinematic limits the regrapsing is initiated as shown in Figure 8 (b-c), following by the insertion motion by Figure 8 (d-e). Two crucial assumptions underscore this approach. Firstly, this approach's effectiveness hinges on the suturing depth - a deeper suturing depth might restrict to only one feasible grasp. Secondly, it presumes the tissue retains the needle to some degree, enabling the PSM to release and regrasp it.

Figure 10 illustrates the needle tip's motion (blue and red lines) once the multi-grasp suturing motion is generated by Algorithm 1. The blue lines represent the insertion phase, and the red lines depict the extraction phase. A discontinuity is noticeable at the junction where the two lines intersect, arising from the grasp alteration during the transition from insertion to extraction. In particular, the grasp location at the extraction phase is closer to the needle tip, the position of which is plotted in this figure. This leads to straight lines when the waypoints are distanced from each other, while the trajectories are more curved for the insertion phase since the grasp is nearly on the opposite side of the circumference. The green and purple points correspond to the target waypoints during the insertion and extraction phases, respectively. Separating the insertion and extraction phase in such a manner is designed to facilitate the interaction between the PSMs in cases of limited manipulability regions, i.e., only one PSM might be able to perform the insertion while the second one can only perform the extraction. The quality of these trajectories substantially depends on the number of waypoints used in their creation, as clearly demonstrated by comparing Figure 10a and Figure 10b. To assess the impact of varying waypoint numbers on trajectory quality, we evaluated trajectories constructed with waypoint sets ranging from as few as 8 to 44. These numbers represent the total count of waypoints, encompassing insertion and extraction phases. They indicate the points on the circular path that the needle tip must follow to guarantee secure suturing. Figure 11 reveals the tracking error change as the waypoint number increases. This error is computed based on the RMSE error between



(a) Trajectories generated for suturing task with 12 waypoints.



(b) Trajectories generated for suturing task with 46 waypoints.

FIGURE 10. Comparison of trajectories for suturing task with different numbers of waypoints.

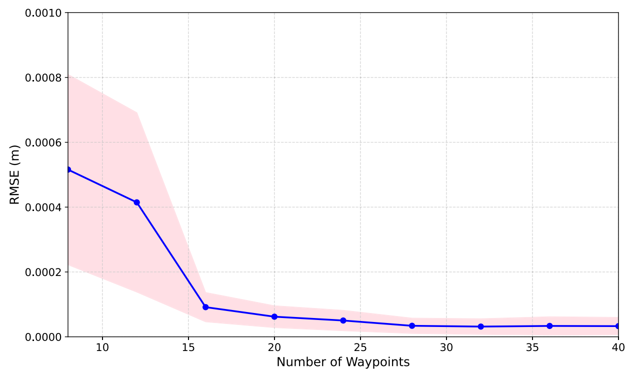


FIGURE 11. Root Mean Square Error (RMSE) error illustrating the trajectory quality generated by AIT* in following the target waypoints.

the expected circular path specified by the waypoint and the actual trajectory for the tip generated by Algorithm 1. Interestingly, even with the smallest set of waypoints, eight, the mean error was minimal at 0.5 mm. This error further decreased to 0.1 mm when the number of waypoints increased to 24, indicating a substantial improvement in trajectory accuracy. This is correlated to the maximum End Effector (EE) task error, which was set at 1×10^{-4} for the IK solver.

V. INTEGRATING TMP WITH DBT

In traditional Task and Motion Planning (TMP), there is a primary focus on the planning process, often assuming the existence of finely-tuned controllers or considering the execution as a separate problem. However, this assumption of perfect plan execution can lead to failure when unexpected errors or disturbances arise. Moreover, since TMP can be time-consuming due to the iteration over numerous samples, re-planning can become prohibitively expensive, especially when dealing with continuous parameters.

Contrarily, BTs offer an execution framework that monitors the execution, enables action re-application if necessary, and adjusts execution in real-time, significantly speeding up the planning-acting loop. DBT was introduced by Colledanchise et al. [23], allowing us to handle unexpected disturbances, such as deviations in a needle's pose after a grasp during task execution, ensuring adaptive and robust completion of tasks.

In this section, we will present an overview of DBTs, followed by the introduction of a hybrid framework that builds on DBT synthesis to encode the TMP solution.

A. DYNAMIC BEHAVIOR TREE

A BT is a directed acyclic graph (DAG) that is structurally represented as a tree, exhibiting control flow from top to bottom (parent to child) among various node types. We adhere to the common definitions of root, control, and leaf nodes within the BT structure: the root node has no parents, while all other nodes have a single parent.

We utilize three types of control nodes: fallback, sequence, and retry decorator [35]. A sequence node only returns success if all of its child nodes succeed. In contrast, a fallback node continues to the next child node if the previous one fails, returning success if any child succeeds and failure otherwise. The retry decorator node facilitates multiple attempts of a behavior, only returning failure once the maximum retry limit is reached.

The concept of DBT synthesis and update is outlined in Algorithm 2. The *CreateReactiveSubtree* function assumes the availability of action templates defined in the PDDL. Given an action $a = (C_{\text{pre}}, C_{\text{post}})$, we create a reactive subtree that continuously checks for necessary conditions and applies the action as needed until the desired outcome is achieved. For DBT execution, specifying the post-effect C_{post} is sufficient, representing the transition we are interested in. Given the goal condition encompassing the C_{pre}, the *UpdateDBT* function

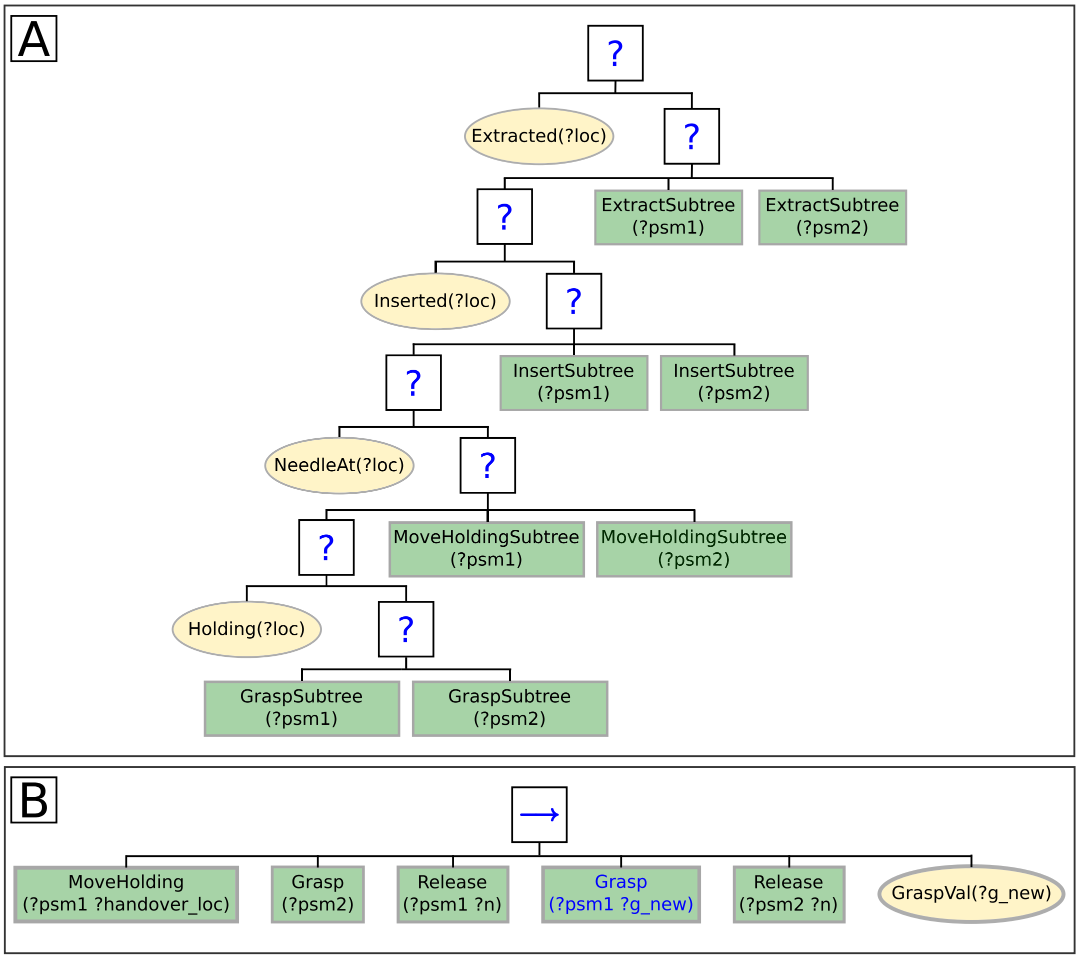


FIGURE 12. **DBT for suturing task using two PSMs, based on Algorithm 2. Subfigure (a) shows a sub-optimal reactive subtree built bottom-up given the `Extracted(?loc)` condition node. Subfigure (b) shows a handover subtree using two PSMs, to update the grasps via handover and regrasping.**

searches for a feasible set of actions iteratively through backchaining. This approach finds fast feasible actions at the cost of optimality as it does not consider the cost or priority of the actions. Sub-optimal DBT solutions for suturing tasks are shown in Figure 12(a).

One of the sub-optimality's reasons lies in the inefficient handling of the available PSMs. Although the DBT structure robustly accommodates both PSMs within a Fallback node to ensure an alternative action route, it does not consider the unique capabilities of each PSM. For instance, if one PSM lacks the necessary dexterity for a specific task, the DBT will still attempt the action with both PSMs, increasing execution times. Furthermore, once an action has been initiated, the state of the environment changes. If an action is unfeasible with the selected PSM, the action must be undone or an additional step must be taken to return to the desired state, thereby extending motion lengths and further increasing execution times. For instance, consider a scenario where PSM1 picks up the needle, but only PSM2 has the dexterity necessary for suturing. Instead of simply dropping the needle for regrasping with the partner robot - an unfavorable action - a handover routine must be triggered as depicted in Figure 12 (b). This process necessitates at least two more additional grasps, resulting in a longer motion trajectory and extended execution time.

B. HYBRID EXECUTION

In contrast, we aim to utilize TMP plans that thoroughly inspect discrete and continuous constraints to obtain a solution. The TMP solution, structured similarly to a PDDL-based plan, specifies additional pose and trajectory-related information. This solution is converted into a DBT following the procedure in the *CreateInitialDBT* function as illustrated in Algorithm 3. Since the TMP plan is a sequence of actions, we reverse their order to ensure the final state transition of interest is monitored.

Consider an example of converting a sequential plan comprising *Grasp*, *MoveHolding*, *Insert*, and *Extract* for a single suturing task. As shown in Figure 13, individual reactive subtrees for each action are combined into a single DBT. This DBT, compared to one obtained via a simple backchaining routine, depicts the solution obtained from

Algorithm 2 Dynamic Behavior Tree Synthesis and Update

```
Function CreateRS ($a = (C_{pre}, C_{post})$):
    Fallback
    Sequence$_{post}$ ← Sequence()
    for $C$ in $C_{post}$ do
        Sequence$_{post}$.addChild(CondNode($C$))
    Fallback.addChild(Sequence$_{post}$)
    Sequence$_{pre}$ ← Sequence()
    for $C$ in $C_{pre}$ do
        Sequence$_{pre}$.addChild(CondNode($C$))
    $counter$ ← Counter($maxRetryCount$)
    Sequence$_{pre}$.addChild(Retry($counter$,
     ActNode($a$)))
    Fallback.addChild(Sequence$_{post}$)
    return Fallback
Function UpdateDBT(DBT):
    $updated$ ← false
    while $hasFailure$(DBT) do
        $CondNode(C_{failed})$ ←
         getFailedCondition(DBT)
        $\mathcal{A}$ ← getSatisfyingActions($C_{failed}$)
        if $|\mathcal{A}| = 0$ then
            return failure
        else if $|\mathcal{A}| = 1$ then
            $a \leftarrow \mathcal{A}[0]$
            $RS$ ← CreateRS($a$)
            replaceNode($CondNode(C_{failed})$, $RS$)
        else
            Fallback ← Fallback()
            for $a$ in $\mathcal{A}$ do
                $RS$ ← CreateDBT($a$)
                Fallback.addChild($RS$)
            replaceNode($CondNode(C_{failed})$,
             Fallback)
        $updated$ ← true
    return $updated$
```

Algorithm 3 Hybrid Execution: Converting the PDDLStream Plan Into DBT

```
Function CreateInitialDBT($PDDL\ plan$):
    Sequence ← Sequence()
    for $a$ in Reverse($\pi$) do
        Sequence.addChild(CreateRS($a$))
    return Sequence
Function Main($\pi$):
    DBT$_{initial}$ ← CreateInitialDBT($\pi$)
    ExecuteDBT(DBT$_{initial}$)
    while UpdateDBT(DBT$_{initial}$) do
        ExecuteDBT(DBT$_{initial}$)
```

TMP, which is probabilistically complete. Hence, our hybrid framework aims to leverage the advantages of both TMP and DBT: a static plan obtained by TMP, albeit based on strong assumptions, is used to generate a probabilistically optimal DBT. This solution in our hybrid execution paradigm can be further adjusted in case of disturbances or unexpected changes in the environment by simply invoking the *UpdateDBT* function, providing flexibility in task execution while reducing the cost of TMP re-planning, which can be prohibitively expensive in surgical scenarios.

VI. SIMULATION BASED EXPERIMENTS

This section discusses an initial performance assessment conducted for the method we propose, within a simulated environment. We employed three primary evaluation metrics: the solve rate, the average execution time, and the count of regrasps. In MIS situations, each additional regrasp by the PSM extends the total path length, possibly inducing deviations in the needle pose. Additionally, multiple regrasps during the insertion or extraction phases could potentially cause tissue damage. Therefore, the chief objective of our planning algorithm was to limit the number of regrasps, thus improving both the efficiency and safety of the procedure.

A. SIMULATION ENVIRONMENT

The system proposed in this study was subjected to testing through simulations on a Linux Ubuntu 20.04 workstation, equipped with an Intel Core i7-8700K processor and 64 GB of RAM. We designed the implementation to be compatible with the Robotics Operating System (ROS) framework. The Pinocchio library (v. 2.6.10) [74] was utilized for kinematic computations, transformations, and kinematic chain parsing. We employed CASadi (v. 3.5.5) [75] as a backend for the nonlinear solver, using IPOPT (v. 3.14.5) [76]. A custom simulation environment was created in Coppeliasim [77], featuring a tissue model for suturing and a circular suturing needle with a radius of 20 mm.

The task planner module was implemented using the PddlStream open library [67]. We used the OMPL library [78] for the implementation of AIT* motion planning. The synthesis of Dynamic Behavior Tree and the behavior manager modules were implemented using the BehaviorTree.CPP library [79]. For the suturing task, two PSMs were employed, each represented by a kinematic chain based on a 7-DOF Robot manipulator (Figure 14 (a)) with an attached 3-DOF robotic surgical tool [80]. The degree of freedom dedicated to operating the gripper was excluded from the IK solving process. It is noteworthy that the positioning of the arms, selection of the RCM, placement of the tissue, pose of suturing points, and the initial location of the needle were not optimized specifically for the task. Instead, the proposed method aims to perform optimally under a variety of conditions. Furthermore, the choice of different kinematic chains can lead to varying levels of performance, demonstrating the robustness and adaptability of our approach to a broad range of surgical scenarios.

In our experiments, we recorded the solve rate, average execution time, and the number of regrasps. Our system autonomously selects the optimal PSM for needle insertion

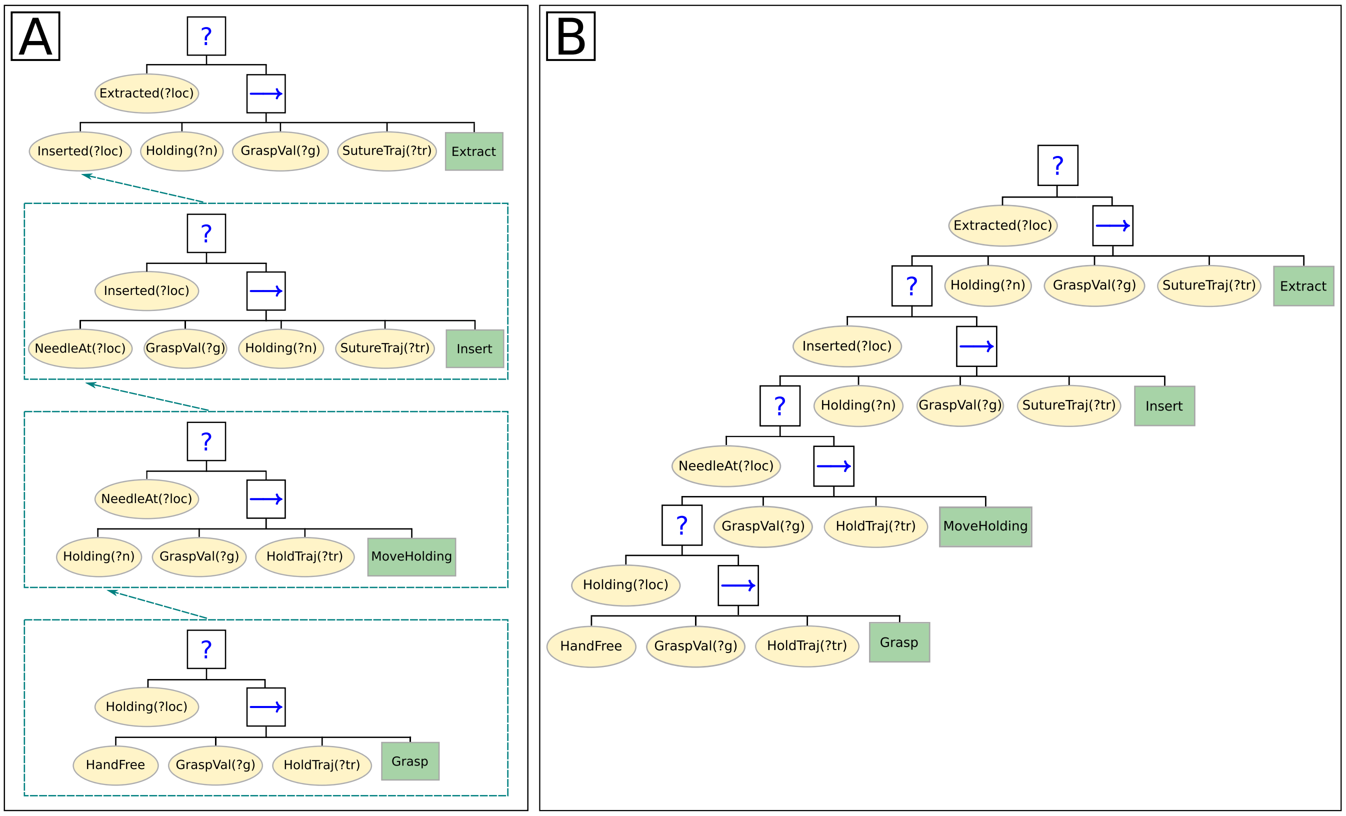


FIGURE 13. DBT for stitching using two PSMs. Subfigure (a) represents the reactive subtrees for the 'grasp', 'move holding', 'insert', and 'extract' actions that are obtained from the TMP planner according to Algorithm 2. Subfigure (b) represents the encoding of the sequential plan from TMP into a single DBT.

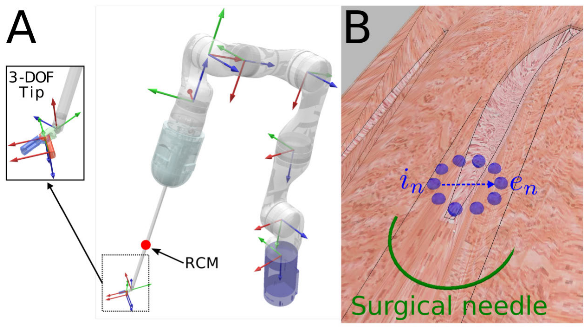


FIGURE 14. (a) The Patient Side Manipulator (PSM) is depicted as a kinematic chain based on a 7-DOF robot manipulator (Kinova Gen3) with a 3-DOF end effector [80]. (b) This represents the possible candidate points for insertion and extraction from a set of blue points for the suturing task.

TABLE 1. Parameters used for the suturing task scenarios.

Parameter	Value
Semi-circle needle radius	0.02 m
Suture depth	0.005 m
Maximum EE speed	$0.005\,\mathrm{m\,s^{-1}}$
Maximum EE task error type	log6
Maximum EE task error	1×10^{-4}
RCM error	1×10^{-4}
Maximum IK computation time (s)	10×10^{-3} s
EE task weight for NLO (w_1)	20
RCM task weight for NLO (w_2)	100
Δq weight for NLO (w_3)	1×10^{-4}
Batch size for grasp samples	150
Number of waypoints	24
AIT* timeout (s)	7 s
DBT action node timeout (s)	15 s
Max. TMP plan computation time (s)	300 s
TMP planner	Adaptive [67]
TMP best-first heuristic search	hmax A* [67]

or extraction based on suture points and needle location. Detailed parameters related to the IK solver, samplers, and planners are provided in Table 1.

B. EVALUATION WITH VARIABLE RCM LOCATIONS

We first examined the TMP in handling different RCM configurations, a key aspect in MIS. The PSMs were attached to four distinct RCM locations, as depicted in Figure 15. The task involved a single throw suturing operation, which required the needle to be picked up from its initial position, oriented based on the suture location, and then inserted and extracted. The task was considered complete upon successful needle extraction, and the execution time, including both planning and solution reproduction times, was subsequently recorded.

Candidate suturing locations were represented by points situated on the circumference of a circle with a radius of 0.012 m. Any two points could be selected as a suturing tuple,

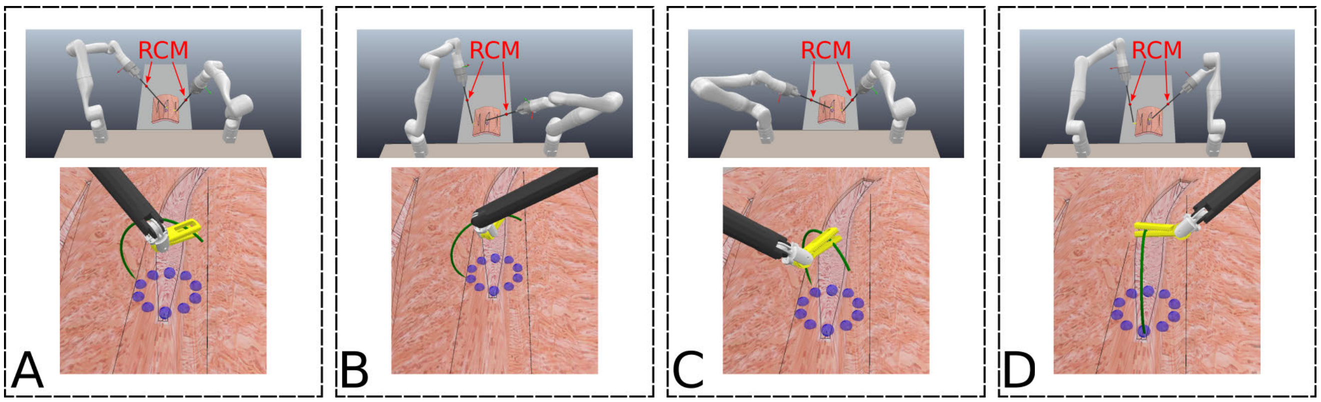


FIGURE 15. RCM configurations for varying RCM task scenario.

indicating the insertion and extraction points, as shown in Figure 14(b). The suturing task was repeated ten times at each RCM location, resulting in a total of 100 attempts per RCM location. Under the assumption of ideal task execution, the offline plan generated by the TMP was executed, and the execution time was recorded. The trajectories were interpolated such that the EE moved at a speed of 5 mm/s. This speed was chosen to closely emulate real-life autonomous stitching scenarios, where the maximum speed is typically limited to prevent unnecessary needle rotations and to ensure proper tissue penetration. As observed in Table 2, all locations exhibited similar cost metrics, averaging 2-4 regrasps with an execution time of around 120 seconds. It's important to note that multiple regrasps here imply that the needle is partially inserted into the tissue, allowing the tissue to hold the needle and thereby enabling the PSM to update the grasp.However, the success rate varied across configurations, ranging from 0.85 - 0.92 for Configuration A-C, with Configuration D recording the lowest success rate of 0.61. This suggests that a well-chosen RCM, leading to high manipulability for the PSMs, results in consistent system performance. On the other hand, a poorly chosen RCM can limit performance. TMP allows examining multiple potential configurations. Its flexibility enables the evaluation of a broad range of parameters and positions, enabling us to identify optimal configurations for a given surgical task. Therefore, TMP not only adapts to different conditions but also offers a potential pathway for optimizing and refining surgical procedures. Furthermore, identifying when an RCM is not suitable for suturing at a particular location or orientation can aid surgeon interaction or tool management, such as requesting takeover or prompting better tissue positioning.

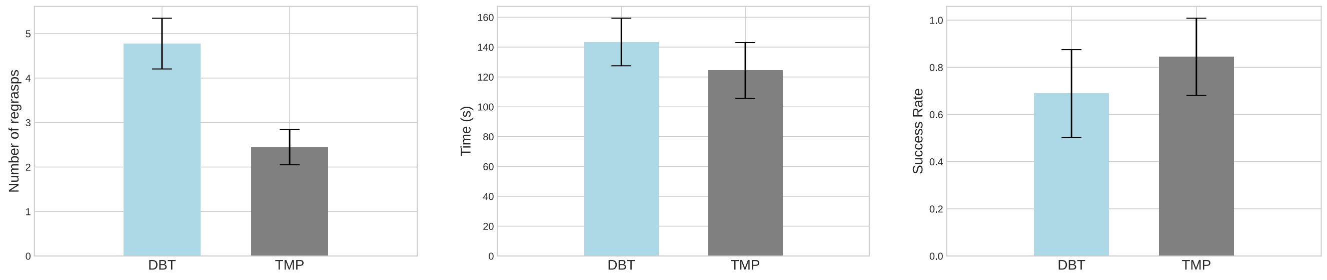
C. COMPARISON BETWEEN OFFLINE TMP AND ONLINE DBT-BASED SOLUTION

The proposed system seeks to combine the benefits of both Task-Motion Planner (TMP) and Dynamic Behavior Tree (DBT). The key difference between these two lies in the fact that TMP is an offline planner with optimality guarantees, while DBT offers suboptimal online planning and execution methods. As both methods can handle the suturing task, we compared their performance to determine if spending additional time on offline grasp and trajectory refinement with TMP is beneficial for a single throw suturing task. It's worth noting that DBT uses the same grasp sampler and motion planners as TMP, but TMP further refines these samples to minimize the total trajectory length and number of regrasps.

The task involved a single throw suturing operation, which required the needle to be picked up from its initial position, oriented based on the suture location, and then inserted and extracted. As in the variable RCMs task, the suture locations were selected from two points on the circumference of a circle with a radius of 0.012 m. In total, ten different locations were evaluated, with ten attempts made for each set of suture points, amounting to a total of one hundred attempts. The RCM location remained consistent for both cases. The grasp attachment and trajectory execution were assumed to be perfect. The task was considered complete once the extraction was achieved. For the TMP-based execution method, the time was recorded once the planning started, which resulted in a total time that included both the planning and execution of the solution. The execution was aborted and considered a failure in the case of collisions, or if the time limit was reached. Specifically for DBT, if the tree returned Failure and no updates were feasible, the execution was aborted and considered as a failure. In scenarios where the grasp was not suitable for orienting and inserting at the respective suture location after picking up the needle, the grasp had to be updated via a handover routine. This was particularly relevant for the DBT-based execution, as it focuses on finding a feasible grasp or trajectory and does not evaluate if it will be valid for other actions in the sequence. The results for both cases are summarized in Figure 16. The main difference between the two methods is evident in the number of regrasps, with TMP resulting in an average of 2-3 regrasps and DBT averaging 4-6 regrasps. This led to similar average execution

TABLE 2. The performance metrics of various RCM configurations are illustrated in the figure below. The top image displays a PSM attached to different RCMs. The bottom image represents the suturing task for a selected set of insertion and extraction points, which are indicated by blue points.

	Success rate	Average execution time (s)	Average number of regrasps
Configuration A	0.85	114.31±69.51	2.78±1.40
Configuration B	0.92	105.84±61.12	2.64±0.95
Configuration C	0.91	120.75±69.08	2.71±0.97
Configuration D	0.61	140.49±73.93	2.38±0.78



(a) Average Number of Regrasps. (b) Average Execution Time. (c) Success Rates.

FIGURE 16. Performance metrics for TMP and DBT-based execution in suturing task with variable suture point locations. The execution time for TMP encompasses both offline planning and execution phases.

times for both cases, recording a mean time of approximately 120s for TMP and 140s for DBT. The success rate for DBT averaged at 0.68 and 0.82 for TMP.

The similarity in execution time is primarily due to the high number of regrasps in the DBT method, most of which were related to unsuitable grasps that led to time-consuming handover routines. Another contributing factor is that DBT has to explore the usefulness of both PSMs given a suturing location. In contrast, TMP chooses one of the two PSMs based on the availability of the grasp and suturing trajectory, which requires checking if the PSM can achieve all the picking, orienting, insertion, and extraction actions. However, DBT evaluates the feasibility of only one action at a time, using the choice of the second PSM as a backup, which leads to handover to continue the execution.

Although the confidence range for both TMP and DBT was high, making it challenging to state a significant difference between the two methods, the difference in the number of regrasps suggests that a high number of regrasps under realistic conditions would inevitably lead to poor performance, underscoring the advantage of TMP.

D. COMPARISON BETWEEN the HYBRID APPROACH AND STANDALONE DBT

We employed BTs for their reactivity, and to handle unexpected environmental changes through action reapplication. While TMP assumes perfect state transitions once actions are applied, making long-horizon solutions useless under unexpected environmental changes, our hybrid approach utilizes TMP to obtain the initial solution and encodes the TMP plan as a DBT to handle unexpected changes. To evaluate the performance of the hybrid method, we introduced noise during execution.

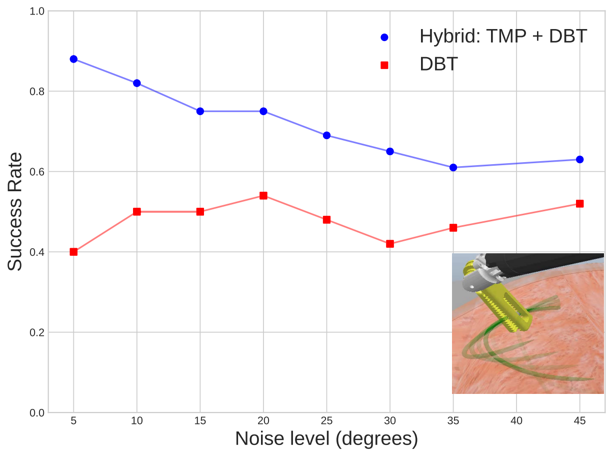
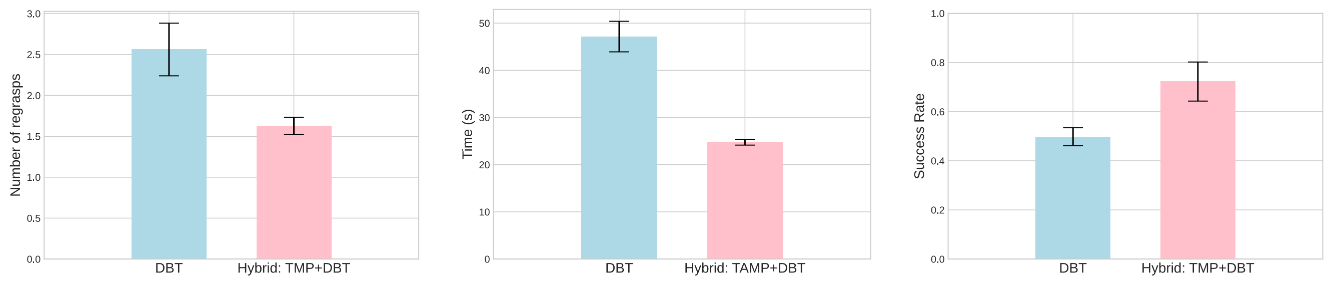


FIGURE 17. The success rate changes as the noise level increases for the needle insertion task. The panel in the bottom right corner illustrates example changes in pose of the needle when a grasp is applied.

The perfect grasp of the surgical needle is often unfeasible due to its small size, as the interaction between the EE and the needle usually results in a change in the grasp pose that deviates from the expected one. The surgical needle rotates around the contact point, leading to a new needle pose. As most suture path planners need to consider the transform between the EE and the tip of the needle, reproducing the computed motions becomes challenging, or the system is forced to limit the maximum speed of suturing to minimize potential deviations. To evaluate this, we introduced random noise, representing a level of rotation around the contact point.

This noise was applied whenever a grasp was needed, such as during needle picking or handover, with an exception made



(a) Average Number of Regrasps. (b) Average Execution Time. (c) Success Rates.

FIGURE 18. **Comparison of performance metrics under noise conditions.**

for regrasping during insertion, where we assumed the tissue held the needle, causing minimal deviation in needle pose. We introduced Gaussian noise for a rotation angle with a mean ranging from 5 to 45 degrees, and the standard deviation for each Gaussian was assumed to be half of the mean. Each noise level was tested a hundred times, with any collisions deemed a failure.

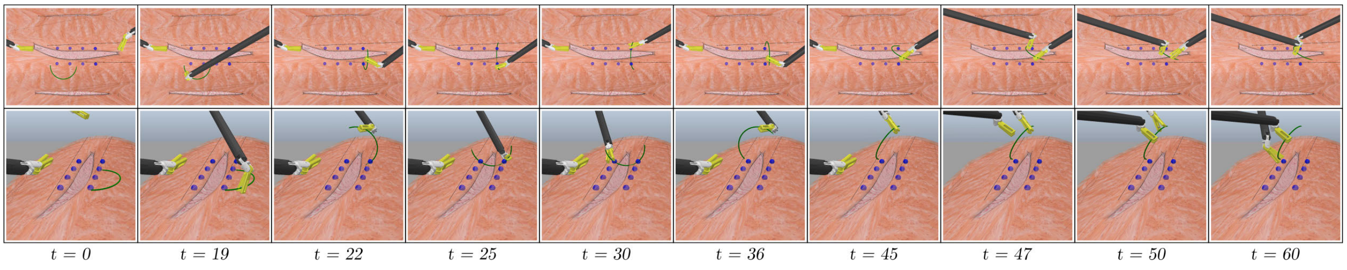
To compare the performance of the hybrid method and the DBT method, we fixed the RCM locations and chose a suture location where both TMP and DBT methods had a 100 percent success rate. The task included needle picking, alignment for suturing, and insertion, and was deemed complete upon successful insertion. Both methods required the use of the handover routine to update the grasp to compensate for the needle pose change, a process that can be time-consuming as noise levels rise. Therefore, we imposed a 60-second timeout, with executions taking longer deemed a failure.

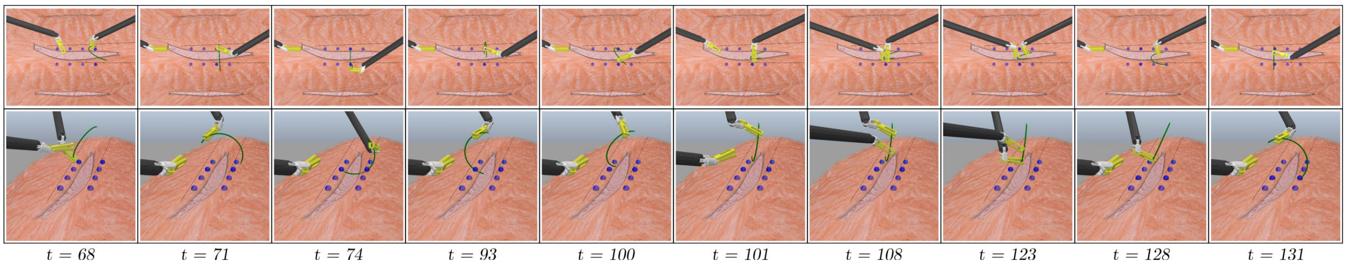
Figure 17 depicts the impact on performance as the noise level increases. The DBT-based execution had a success rate fluctuating between 0.4 and 0.58 as the noise level increased. In contrast, the proposed method recorded a success rate above 0.8 when the noise level was between 5-10 degrees, gradually dropping to 0.6 as the noise level increased. Additional evaluation metrics, including execution time, number of regrasps, and average success rates for both methods, are illustrated in Figure 18. The proposed hybrid approach outperforms the DBT method across all metrics. The average success rate is notably higher for the hybrid approach, approximately 0.72 as compared to 0.51 for DBT (see Figure 18c). Furthermore, both the execution time and the number of regrasps are lower for the hybrid method. Specifically, the mean execution time is approximately 24s for the hybrid method as opposed to 47s for the DBT method (Figure 18b), and the average number of regrasps is between 1 and 2 for the hybrid method, compared to 2 to 3 for the DBT method (Figure 18a). While both methods utilize the same DBT's algorithm to handle noise, the additional evaluations used by TMP in the hybrid approach to choose the most suitable PSM and grasp pose lead to overall minimization of the number of regrasps. This results in a higher success rate, as each additional regrasp contributes to the noise and thereby increases the chance of failure.

E. APPLICATION TO MULTI-THROW SUTURING TASK

Finally, we demonstrated how the proposed framework can be applied to a more complex multi-throw suturing task. In previous experiments, we focused on single-throw suturing, varying the orientation for a given suture location. Here, we considered a four-throw suture scenario under noise-free conditions: four suture locations were distributed around a wound model at a distance of 0.02m apart. The task followed this sequence: picking the needle from its initial location, orienting the needle, suturing, and handing over to the partner PSM to achieve a suitable grasp for repeating the process at the next suture location. Prior to execution, the most suitable PSM for each suture point was selected using the proposed framework, with handover-related grasps and trajectories computed on the fly. Figure 19 illustrates snapshots of the multi-throw suturing task. Each frame in the sequence has a top image showing the top view and a bottom image showing the side view, with t representing the time since the start of execution. At $t = 0$, we see the initial configuration of the two PSMs. We refer to the PSM on the left as $psm1$ and the one on the right as $psm2$. At $t = 19$, $psm2$ picks up the needle and orients it at the first suture location by $t = 22$. The first throw is completed by $t = 36$. Next, between $t = 35$ and $t = 68$, we see the first handover routine: the needle is passed to $psm1$ at $t = 50$ and $psm2$ regrasps the needle with a new grasp at $t = 68$. The orientation and subsequent second throw occur between $t = 71$ and $t = 93$. After that, the second handover and third throw are completed by $t = 145$. Finally, the third handover is performed, and the last fourth insertion is captured at $t = 211$. The four-throw task is completed by $t = 226$, taking a total of 3 minutes and 46 seconds. Full videos of several four-throw suturing tasks can be viewed at the project's link.

This demonstration aims to illustrate how the proposed combination of TMP and DBT allows for seamless execution of the complete suturing task, encompassing all main steps such as needle picking, suturing motions, and the handover of the needle between PSMs. Moreover, as shown in Figure 19,


(a) Initial pick-up and suturing of the first point, followed by a subsequent handover to PSM1


(b) Subsequent suturing at the second location, followed by a handover and the initiation of insertion at the third point.

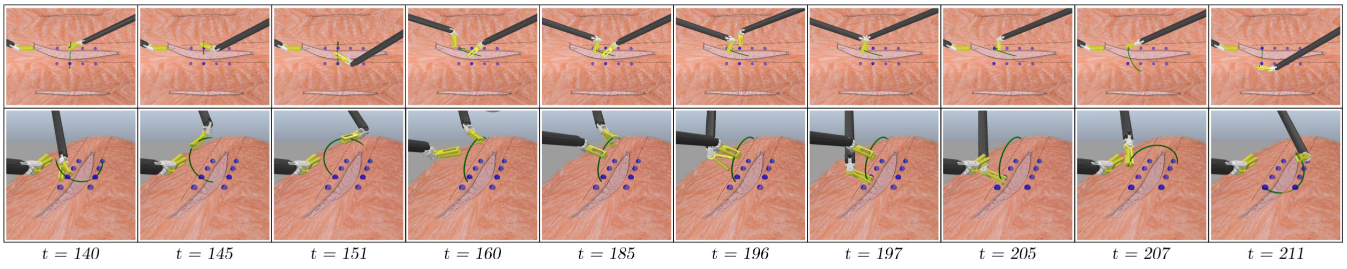

(c) Suturing at the third location, followed by a handover and the beginning of the fourth throw.

FIGURE 19. **Multi-throw suturing out of 4 throws.**

the execution results in collision-free motions that respect the RCM constraint across various suture locations.

VII. CONCLUSION AND FUTURE WORKS

In this study, we developed a hybrid system that combines TMP and DBT for autonomous robotic surgery, with a particular focus on suturing tasks in MIS. By bringing together the strengths of both TMP and DBT, our method demonstrates high performance across a variety of conditions.

Our evaluations confirmed the effectiveness of our approach in different starting conditions and noisy environments. The hybrid method outperformed the standalone DBT technique in terms of success rates and execution times, illustrating resilience even under increased noise conditions. Notably, the motion sampling strategies enabled us to acquire collision-free trajectories promptly, showing substantial promise for real-world applications. The TMP component allowed us to establish and incorporate a range of initial conditions and constraints and obtain a probabillistically optimal solution that minimizes cost. Furthermore, the proposed framework generated the final solution in the form of a behavior tree, enhancing the system's explainability by graphically depicting the execution flow and indicating failure points. This attribute can facilitate the communication between the ARSS and the human supervisor.

Addressing the limitations of this work, we initially assumed an idealized needle-tissue interaction. Real-world surgical settings present complexities arising from varied tissue attributes, i.e. stiffness, rigidity, and elasticity, potentially causing unpredictable tool deviations at the end effector. Perception-related challenges also exist: accurate discernment of crucial elements like the needle, tissue, suturing points, and the associated manipulator joint states is paramount. Although behavior trees can mitigate sensing inaccuracies through action re-sampling, establishing the correct decision thresholds and behaviors is essential. Furthermore, our methodology prioritizes generating smooth trajectories via b-spline interpolation. Yet, for effective hardware deployment, it is crucial to also maintain consistent velocity and acceleration profiles for generated trajectories. This ensures oscillations are minimized, potential motor noise is addressed and a safe tool-tissue interaction is achieved.

Despite the encouraging outcomes, there is potential for further refinement of the TMP-DBT hybrid technique to tackle more complex and realistic surgical scenarios. Future endeavors could include extending the framework to evaluate the system under sensor noise and partial-observability scenarios. It's also crucial to explore the system's capability to handle unforeseen events, such as tissue deformations, loss of the needle, or managing time constraints in planning scenarios. Additionally, research should focus on the potential coordination between more than two PSMs or surgical robots, for instance, multiple PSMs and an endoscope. Progress in these areas is essential for deploying ARSS in real-world medical scenarios in the future.

REFERENCES

[1] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artif. Intell.*, vol. 247, pp. 10–44, Jun. 2017.

[2] A. Attanasio, B. Scaglioni, E. De Momi, P. Fiorini, and P. Valdastri, "Autonomy in surgical robotics," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 4, no. 1, pp. 651–679, May 2021.

[3] M. Yip and N. Das, "Robot autonomy for surgery," in *The Encyclopedia of MEDICAL ROBOTICS: Volume 1 Minimally Invasive Surgical Robotics*. Singapore: World Scientific, 2019, pp. 281–313.

[4] S. O'Sullivan, N. Nevejans, C. Allen, A. Blyth, S. Leonard, U. Pagallo, K. Holzinger, A. Holzinger, M. I. Sajid, and H. Ashrafian, "Legal, regulatory, and ethical frameworks for development of standards in artificial intelligence (AI) and autonomous robotic surgery," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 15, no. 1, p. e1968, Feb. 2019.

[5] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen, and K. Goldberg, "Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 4178–4185.

[6] S. Iyer, T. Looi, and J. Drake, "A single arm, single camera system for automated suturing," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 239–244.

[7] J. R. U. Roldan and D. Milutinovic, "Suture looping task pose planner in a constrained surgical environment," *J. Intell. Robotic Syst.*, vol. 106, no. 4, p. 78, Dec. 2022.

[8] J. Colan, J. Nakanishi, T. Aoyama, and Y. Hasegawa, "Optimization-based constrained trajectory generation for robot-assisted stitching in endonasal surgery," *Robotics*, vol. 10, no. 1, p. 27, Feb. 2021.

[9] S. A. Pedram, C. Shin, P. W. Ferguson, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous suturing framework and quantification using a cable-driven surgical robot," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 404–417, Apr. 2021.

[10] A. Knoll, H. Mayer, C. Staub, and R. Bauernschmitt, "Selective automation and skill transfer in medical robotics: A demonstration on surgical knot-tying: Selective automation and skill transfer in medical robotics," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 8, no. 4, pp. 384–397, Dec. 2012.

[11] A. Attanasio, B. Scaglioni, M. Leonetti, A. F. Frangi, W. Cross, C. S. Biyani, and P. Valdastri, "Autonomous tissue retraction in robotic assisted minimally invasive surgery—A feasibility study," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6528–6535, Oct. 2020.

[12] D. Meli, E. Tagliabue, D. Dall'Alba, and P. Fiorini, "Autonomous tissue retraction with a biomechanically informed logic based framework," in *Proc. Int. Symp. Med. Robot. (ISMR)*, Nov. 2021, pp. 1–7.

[13] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. K. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Visuospatial foresight for multi-step, multi-task fabric manipulation," 2020, *arXiv:2003.09044*.

[14] K. L. Schwaner, D. Dall'Alba, P. T. Jensen, P. Fiorini, and T. R. Savarimuthu, "Autonomous needle manipulation for robotic surgical suturing based on skills learned from demonstration," in *Proc. IEEE 17th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2021, pp. 235–241.

[15] V. M. Varier, D. K. Rajamani, N. Goldfarb, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "Collaborative suturing: A reinforcement learning approach to automate hand-off task in suturing for surgical robots," in *Proc. 29th IEEE Int. Conf. Robot Human Interact. Commun. (RO-MAN)*, Aug. 2020, pp. 1380–1386.

[16] C. D'Ettorre, G. Dwyer, X. Du, F. Chadebecq, F. Vasconcelos, E. De Momi, and D. Stoyanov, "Automated pick-up of suturing needles for robotic surgical assistance," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1370–1377.

[17] A. Wilcox, J. Kerr, B. Thananjeyan, J. Ichnowski, M. Hwang, S. Paradis, D. Fer, and K. Goldberg, "Learning to localize, grasp, and hand over unmodified surgical needles," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 9637–9643.

[18] P. Sundaresan, B. Thananjeyan, J. Chiu, D. Fer, and K. Goldberg, "Automated extraction of surgical needles from tissue phantoms," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 170–177.

[19] Z.-Y. Chiu, F. Richter, E. K. Funk, R. K. Orosco, and M. C. Yip, "Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 7737–7743.

[20] K. Watanabe, T. Kanno, K. Ito, and K. Kawashima, "Single-master dual-slave surgical robot with automated relay of suture needle," *IEEE Trans. Ind. Electron.*, vol. 65, no. 8, pp. 6343–6351, Aug. 2018.

[21] M. Ghallab, D. Nau, and P. Traverso, "The actor's view of automated planning and acting: A position paper," *Artif. Intell.*, vol. 208, pp. 1–17, Mar. 2014.

[22] D. Nau, M. Ghallab, and P. Traverso, "Blended planning and acting: Preliminary approach, research challenges," in *Proc. AAAI Conf. Artif. Intell.*, 2015, vol. 29, no. 1, pp. 1–5.

[23] M. Colledanchise, D. Almeida, and P. Ögren, "Towards blended reactive planning and acting using behavior trees," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8839–8845.

[24] J. Sandoval, H. Su, P. Vieyres, G. Poisson, G. Ferrigno, and E. De Momi, "Collaborative framework for robot-assisted minimally invasive surgery using a 7-DoF anthropomorphic robot," *Robot. Auto. Syst.*, vol. 106, pp. 95–106, Aug. 2018.

[25] M. Hwang, B. Thananjeyan, S. Paradis, D. Seita, J. Ichnowski, D. Fer, T. Low, and K. Goldberg, "Efficiently calibrating cable-driven surgical robots with RGBD fiducial sensing and recurrent neural networks," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5937–5944, Oct. 2020.

[26] M. Haghighipanah, Y. Li, M. Miyasaka, and B. Hannaford, "Improving position precision of a servo-controlled elastic cable driven surgical robot using unscented Kalman filter," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 2030–2036.

[27] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, pp. 265–293, May 2021.

[28] A. Suárez-Hernández, G. Alenyà, and C. Torras, "Interleaving hierarchical task planning and motion constraint testing for dual-arm manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4061–4066.

[29] C. R. Garrett, C. Paxton, T. Lozano-Perez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2020, pp. 5678–5684.

[30] C. V. Braun, J. Ortiz-Haro, M. Toussaint, and O. S. Oguz, "RHH-LGP: Receding horizon and heuristics-based logic-geometric programming for task and motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 13761–13768.

[31] W. Thomason, M. P. Strub, and J. D. Gammell, "Task and motion informed trees (TMIT): Almost-surely asymptotically optimal integrated task and motion planning," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 11370–11377, Oct. 2022.

[32] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *Int. J. Robot. Res.*, vol. 40, nos. 6–7, pp. 866–894, Jun. 2021.

[33] J. Jeon, H.-R. Jung, F. Yumbla, T. A. Luong, and H. Moon, "Primitive action based combined task and motion planning for the service robot," *Frontiers Robot. AI*, vol. 9, Feb. 2022, Art. no. 713470.

[34] M. Mansouri, F. Pecora, and P. Schüller, "Combining task and motion planning: Challenges and guidelines," *Frontiers Robot. AI*, vol. 8, May 2021, Art. no. 637888.

[35] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. Boca Raton, FL, USA: CRC Press, 2018.

[36] H. Nakawala, E. De Momi, A. Tzemanaki, S. Dogramadzi, A. Russo, M. Catellani, R. Bianchi, O. De Cobelli, A. Sideridis, E. Papacostas, A. Koupparis, E. Rowe, R. Persad, R. Ascione, and G. Ferrigno, "Requirements elicitation for robotic and computer-assisted minimally invasive surgery," *Int. J. Adv. Robotic Syst.*, vol. 16, no. 4, Jul. 2019, Art. no. 172988141986580.

[37] G. A. Fontanelli, G.-Z. Yang, and B. Siciliano, "A comparison of assistive methods for suturing in MIRS," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4389–4395.

[38] R. Nagyné Elek and T. Haidegger, "Non-technical skill assessment and mental load evaluation in robot-assisted minimally invasive surgery," *Sensors*, vol. 21, no. 8, p. 2666, Apr. 2021.

[39] A. De Benedictis, A. Trezza, A. Carai, E. Genovese, E. Procaccini, R. Messina, F. Randi, S. Cossu, G. Esposito, P. Palma, P. Amante, M. Rizzi, and C. E. Marras, "Robot-assisted procedures in pediatric neurosurgery," *Neurosurgical Focus*, vol. 42, no. 5, p. E7, May 2017.

[40] T. Liu and M. C. Cavusoglu, "Needle grasp and entry port selection for automatic execution of suturing tasks in robotic minimally invasive surgery," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 552–563, Apr. 2016.

[41] F. Nageotte, P. Zanne, C. Doignon, and M. de Mathelin, "Stitching planning in laparoscopic surgery: Towards robot-assisted suturing," *Int. J. Robot. Res.*, vol. 28, no. 10, pp. 1303–1321, Oct. 2009.

[42] C. Gruijthuijsen, L. C. Garcia-Peraza-Herrera, G. Borghesan, D. Reynaerts, J. Deprest, S. Ourselin, T. Vercauteren, and E. V. Poorten, "Robotic endoscope control via autonomous instrument tracking," *Frontiers Robot. AI*, vol. 9, Apr. 2022, Art. no. 832208.

[43] X. Ma, C. Song, L. Qian, W. Liu, P. W. Chiu, and Z. Li, "Augmented reality-assisted autonomous view adjustment of a 6-DOF robotic stereo flexible endoscope," *IEEE Trans. Med. Robot. Bionics*, vol. 4, no. 2, pp. 356–367, May 2022.

[44] S. Paradis, M. Hwang, B. Thananjeyan, J. Ichnowski, D. Seita, D. Fer, T. Low, J. E. Gonzalez, and K. Goldberg, "Intermittent visual servoing: Efficiently learning policies robust to instrument changes for high-precision surgical manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 7166–7173.

[45] T. Da Col, G. Caccianiga, M. Catellani, A. Mariani, M. Ferro, G. Cordima, E. De Momi, G. Ferrigno, and O. de Cobelli, "Automating endoscope motion in robotic surgery: A usability study on da vinci-assisted ex vivo neobladder reconstruction," *Frontiers Robot. AI*, vol. 8, Nov. 2021, Art. no. 707704.

[46] X. Ma, C. Song, P. W. Chiu, and Z. Li, "Autonomous flexible endoscope for minimally invasive surgery with enhanced safety," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2607–2613, Jul. 2019.

[47] A. M. Jarc and M. J. Curet, "Viewpoint matters: Objective performance metrics for surgeon endoscope control during robot-assisted surgery," *Surgical Endoscopy*, vol. 31, no. 3, pp. 1192–1202, Mar. 2017.

[48] M. Hayashibe, N. Suzuki, M. Hashizume, K. Konishi, and A. Hattori, "Robotic surgery setup simulation with the integration of inverse-kinematics computation and medical imaging," *Comput. Methods Programs Biomed.*, vol. 83, no. 1, pp. 63–72, Jul. 2006.

[49] J. Yang, Y. Zhao, Z. Han, Y. Zhang, and W. Wang, "Preoperative planning of three axis intersection surgical laparoscopic arm system based on characteristic parameters," *Int. J. Comput. Assist. Radiol. Surg.*, pp. 1–11, Mar. 2023.

[50] Z. Du, W. Wang, W. Wang, and W. Dong, "Preoperative planning for a multi-arm robot-assisted minimally invasive surgery system," *Simulation*, vol. 93, no. 10, pp. 853–867, Oct. 2017.

[51] P. Su, J. Li, C. Yue, T. Liu, B. Liu, and J. Li, "Preoperative positioning planning for a robot-assisted minimally invasive surgical system based on accuracy and safety," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 18, no. 4, p. e2405, Aug. 2022.

[52] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *Int. J. Robot. Res.*, vol. 36, nos. 13–14, pp. 1595–1618, Dec. 2017.

[53] D. Meli and P. Fiorini, "Unsupervised identification of surgical robotic actions from small non-homogeneous datasets," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 8205–8212, Oct. 2021.

[54] G. De Rossi, M. Minelli, S. Roin, F. Falezza, A. Sozzi, F. Ferraguti, F. Setti, M. Bonfè, C. Secchi, and R. Muradore, "A first evaluation of a multi-modal learning system to control surgical assistant robots via action segmentation," *IEEE Trans. Med. Robot. Bionics*, vol. 3, no. 3, pp. 714–724, Aug. 2021.

[55] Y. Sharon and I. Nisky, "Expertise, teleoperation, and task constraints affect the speed–curvature–torsion power law in RAMIS," *J. Med. Robot. Res.*, vol. 3, Sep. 2018, Art. no. 1841008.

[56] M. Ginesi, D. Meli, H. Nakawala, A. Roberti, and P. Fiorini, "A knowledge-based framework for task automation in surgery," in *Proc. 19th Int. Conf. Adv. Robot. (ICAR)*, Dec. 2019, pp. 37–42.

[57] F. Falezza, N. Piccinelli, G. De Rossi, A. Roberti, G. Kronreif, F. Setti, P. Fiorini, and R. Muradore, "Modeling of surgical procedures using statecharts for semi-autonomous robotic surgery," *IEEE Trans. Med. Robot. Bionics*, vol. 3, no. 4, pp. 888–899, Nov. 2021.

[58] M. J. Fard, S. Ameri, R. Darin Ellis, R. B. Chinnam, A. K. Pandya, and M. D. Klein, "Automated robot-assisted surgical skill evaluation: Predictive analytics approach," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 14, no. 1, p. e1850, Feb. 2018.

[59] R. Muradore, P. Fiorini, G. Akgun, D. E. Barkana, M. Bonfe, F. Boriero, A. Caprara, G. D. Rossi, R. Dodi, O. J. Elle, and F. Ferraguti, "Development of a cognitive robotic system for simple surgical tasks," *Int. J. Adv. Robotic Syst.*, vol. 12, no. 4, p. 37, Apr. 2015.

[60] T. D. Nagy and T. Haidegger, "A DVRK-based framework for surgical subtask automation," *Acta Polytechnica Hungarica*, vol. 16, no. 8, pp. 61–78, Oct. 2019.

[61] D. Hu, Y. Gong, B. Hannaford, and E. J. Seibel, "Semi-autonomous simulated brain tumor ablation with RAVENII surgical robot using behavior tree," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 3868–3875.

[62] C. Pezzato, C. H. Corbato, S. Bonhof, and M. Wisse, "Active inference and behavior trees for reactive action planning and execution in robotics," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 1050–1069, Apr. 2023.

[63] A. Leporini, E. Oleari, C. Landolfo, A. Sanna, A. Larcher, G. Gandaglia, N. Fossati, F. Muttin, U. Capitanio, and F. Montorsi, "Technical and functional validation of a teleoperated multirobots platform for minimally invasive surgery," *IEEE Trans. Med. Robot. Bionics*, vol. 2, no. 2, pp. 148–156, May 2020.

[64] F. Setti, E. Oleari, A. Leporini, D. Trojaniello, A. Sanna, U. Capitanio, F. Montorsi, A. Salonia, and R. Muradore, "A multirobots teleoperated platform for artificial intelligence training data collection in minimally invasive surgery," in *Proc. Int. Symp. Med. Robot. (ISMR)*, Apr. 2019, pp. 1–7.

[65] Z. Wang, Z. Liu, Q. Ma, A. Cheng, Y.-H. Liu, S. Kim, A. Deguet, A. Reiter, P. Kazanzides, and R. H. Taylor, "Vision-based calibration of dual RCM-based robot arms in human–robot collaborative minimally invasive surgery," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 672–679, Apr. 2018.

[66] F. Ficuciello, G. Tamburrini, A. Arezzo, L. Villani, and B. Siciliano, "Autonomy in surgical robots and its meaningful human control," *J. Paladyn Behav. Robot.*, vol. 10, no. 1, pp. 30–43, Jan. 2019.

[67] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 30, 2020, pp. 440–448.

[68] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, and D. Christianson, "PDDL—The planning domain definition language," Yale Center Comput. Vis. Control, New Haven, CC, USA, Tech. Rep. CVC TR-98-003/DCS TR-1165, 1998.

[69] J. Colan, A. Davila, K. Fozilov, and Y. Hasegawa, "A concurrent framework for constrained inverse kinematics of minimally invasive surgical robots," *Sensors*, vol. 23, no. 6, p. 3328, Mar. 2023.

[70] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018, *arXiv:1812.01537*.

[71] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3859–3866.

[72] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "RViz: A toolkit for real domain data visualization," *Telecommun. Syst.*, vol. 60, no. 2, pp. 337–345, Oct. 2015.

[73] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3191–3198.

[74] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2019, pp. 614–619.

[75] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

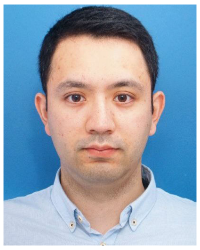
[76] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[77] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1321–1326.

[78] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

[79] D. Faconti and M. Colledanchise. (2019). *Behaviortree.cpp*. Accessed: Jul. 2023. [Online]. Available: https://www.behaviortree.dev

[80] J. Colan, A. Davila, Y. Zhu, T. Aoyama, and Y. Hasegawa, "OpenRST: An open platform for customizable 3D printed cable-driven robotic surgical tools," *IEEE Access*, vol. 11, pp. 6092–6105, 2023.



KHUSNIDDIN FOZILOV received the B.E. and M.E. degrees from Nagoya University, Nagoya, Japan, in 2017 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the Department of Micro-Nano Mechanical Science and Engineering. His main research interests include information theory, decision-making, multi-agent systems, intelligent assistive systems, and medical robotics.

JACINTO COLAN (Member, IEEE) received the B.S. degree from the National University of Engineering, Lima, Peru, in 2010, and the M.E. and Ph.D. degrees in micro-nano mechanical science and engineering from Nagoya University, Nagoya, Japan, in 2018 and 2021, respectively. He is currently a Postdoctoral Researcher with Nagoya University. His main research interests include medical robotics, human–robot interfaces, and intelligent assistive systems. He received the Best Paper Award from the IEEE International Symposium on Micro-Nano Mechatronics and Human Science (MHS 2019 and MHS 2022).

KOSUKE SEKIYAMA received the B.E., M.E., and Dr.Eng. degrees from Nagoya University, Nagoya, Japan, in 1992, 1994, and 1997, respectively. From 1998 to 2001, he was a Lecturer with the Tokyo University of Science. From 2001 to 2006, he was an Associate Professor with the University of Fukui. From 2006 to 2019, he was an Associate Professor with the Department of Micro-Nano Mechanical Science and Engineering, Nagoya University. Since 2019, he has been with Meijo University, where he is currently a Professor with the Department of Mechatronics Engineering. His research interests include distributed autonomous systems, in particular, self-organizing systems at various system levels, multi-agent systems, and cyber-physical systems.

YASUHISA HASEGAWA (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in robotics from Nagoya University, Nagoya, Japan, in 1994, 1996, and 2001, respectively. From 1996 to 1998, he was with Mitsubishi Heavy Industries Ltd., Japan. He joined Nagoya University, in 1998. He moved to Gifu University, in 2003. From 2004 to 2014, he was with the University of Tsukuba. Since 2014, he has been with Nagoya University, where he is currently a Professor with the Department of Micro-Nano Mechanical Science and Engineering. His main research interests include the research fields of motion-assistive systems, teleoperation for manipulation, and surgical support robot.