2020 International Symposium on Medical Robotics (ISMR)
Atlanta, GA, USA, November 18-20, 2020

Dual-Arm Needle Manipulation with the da Vinci®Surgical Robot

Su Lu[1], Thomas Shkurti[2] and M. Cenk Çavuşoğlu[1]

2020 International Symposium on Medical Robotics (ISMR) 978-1-7281-5488-6/20/$31.00 ©2020 IEEE DOI: 10.1109/ISMR48331.2020.9312930

***Abstract*— This paper presents a method of planning and performing dual-arm needle grasp and regrasp manipulations using the da Vinci®Robotic Surgical System. This method implements a hybrid-object-based state-space Rapidly-Exploring Random Tree** (RRT) **algorithm to manage needle regrasps. Previous methods have attempted to pick up a loose surgical needle in the environment and then use the same grasp to begin suturing; however, the initial needle grasp is not always ideal in terms of the needle pose with respect to the gripper or the workspace of the arm. Thus, it is advantageous to regrasp the needle in a more optimal manner before suturing proper begins. To solve the problem, a hybrid-object state space (*HySS*) planner is implemented under the** RRT – Connect **motion planning framework. This planner is able to find a collision-free handoff manipulation path connected by a series of intermediate hybrid states. We show that this algorithm produces trajectories which can be successfully executed by a simulated da Vinci® robot and result in grasps that are optimal as defined by [1].**

***Index Terms*— surgical robotics, motion planning and control, object regrasping**

I. INTRODUCTION

In order to improve object-manipulation efficiency, dual-arm robots have been introduced into industrial environments. Two arms allow pick-and-place tasks to make use of handoff movements to transition from a grasp ideal for picking to a different grasp ideal for placing; and objects can to be transferred through a workspace larger than that of a single arm [2].

This approach is particularly appealing in surgical robotics for the suturing task: small, semicircular needles are introduced into the body cavity at arbitrary locations and must be grasped near the base before being driven through a different section of tissue in a predetermined arc; it is not always possible to grasp the needle and drive it with the same gripper, necessitating one or more re-grasps. In the suturing cycle, the first motion will be the grasping of the needle by one of the patient-side manipulators (*PSM*s). However, the initial grasp may not always be ideal in terms of the needle pose, arm selection, or grasp transformation from needle to gripper. In order to address this issue, needle regrasping needs to be performed between the two *PSM*s such that after a small number handoffs, the needle can be grasped in a way that is optimal for insertion into tissue. Repeated needle regrasp manipulations are then needed to perform a running suture.

This paper proposes a hybrid-object-based Rapidly-exploring Random Tree (RRT) [3] motion-planning algorithm to calculate a collision-free manipulation path which accommodates handoffs, for the dual *PSM*s of the da Vinci®Surgical Robot. A manipulation path is divided into a sequence of actions sorted into two different categories: a needle transfer by single arm in Cartesian space, and a handoff between arms which does not move the needle. A planner specification exploiting the RRT – Connect [4] planning framework is illustrated in Section IV. Section IV-C defines the hybrid-object state space (*HySS*) which is composed of the needle pose, arm index and a grasp index selected from a pre-defined grasp transformation list. To grow the RRT tree, a heuristic distance function is defined in Section IV-D. The state interpolation implementation necessary to interpolate between two different hybrid states is described in Section IV-E. To fully connect these states a collision checker and local planner are also explained in Section IV-F.

Though much research involving dual-arm robot manipulation has been done [5], this study is the first within a surgical-robotics context to fit a novel *HySS* into a standard sampling-based planning scheme, while taking into account state sampling and interpolation between continuous and discrete variables.

II. RELATED WORK

Extensive studies have been done on dual-arm manipulation path planning (refer to [5] for a recent survey). This class of problem becomes distinct from single-arm manipulation planning when regrasping by a handoff action is considered, instead of pick-and-place with a single arm. In these scenarios, an object is (for instance) unreachable by robot's right arm but reachable by its left arm, and the objective is for the right arm to put the object in a box nearby.

The naive solution is to pick up the object with the left arm first, then place it on a supporting surface close enough for right arm to pick up again. This "pick-place-pick" solution is inefficient because it requires the offline storage of data including different workspace dimensions, end-effector configurations, and grasp transforms. Marino et al. [6] pre-generate and store in a database all possible grasps between the start state and goal state; to connect the two a Dijkstra shortest path algorithm is used to search through the database. This database requires storing not only a feasible object transition table, but also geometric information about the object and supporting surface, approximation of

This work was supported in part by the National Science Foundation under grants CISE IIS-1524363, CISE IIS-1563805, and DUE-1928485.

[1]Su Lu and M. Cenk Çavuşoğlu is with the Department of Electrical, Computer, and Systems Engineering, Case Western Reserve University, Cleveland, OH, 44106, USA sxl924,mcc14case.edu

[2]Thomas Shkurti is with the Department of Computer and Data Sciences, Case Western Reserve University, Cleveland, OH, 44106, USA tes77case.edu

reachable region of each arm, and grasp transformations. Additionally, this method does not guarantee a collision-free path because collision checking is postponed until after planning.

Probabilistic Roadmap (PRM) sampling path-planning algorithms have also been employed in regrasping-based manipulation scenarios. Saut et al. [7] offer an offline decomposition technique for a kinematically-independent dual-arm robot, which generates a "SuperGrasp" ("SG") by merging two graphs which form a collision-free roadmap for the left arm and right arm, respectively. After this, the collision-free path is realized by a sequential online query on SG to find desired motions, i.e. object grasping, carrying to an exchange location, object placement, and going to the rest configuration. Though this method is well-suited to resolving multiple manipulation problems in the same environment, the nature of the PRM remains a disadvantage in that it requires offline computation of roadmaps specific to each environment.

Branicky et al. [8] present the original RRT evolved to a hybrid version which can solve motion problems within a continuous-and-discrete state space. A "stair climber" experiment demonstrated that the hybrid RRT could solve a problem where the initial state was located on the first floor of a building and the goal state was located on the fourth floor. In this example, the switching points (the location of stairs on each floor) were given, the planner was tailored to find a path from each switching point to another, and eventually found a solution path connecting the first floor to the fourth floor goal point.

The *HySS* introduced in this study is inspired by the above work, although in the proposed approach switching points are not explicitly defined. This means that for the needle regrasping problem, the location in Cartesian space where the handoff occurs remains unknown before planning starts. Our research also differs from the above in that the planning does not depend on an external database; only the object's geometric model and grasp transformations are stored for planning. Additionally, thanks to the probabilistic completeness of RRT – Connect, a collision free path is always findable given enough planning time. The offline tree-building of RRT is faster than PRM, because rather than exploring the entire $\mathcal{Q}_{free}$ space, tree growth is guided by sampling towards the goal. Finally, in the querying stage there is only one solution path connecting the start and goal state, which is easier to find by simple back-tracing.

III. Problem Definition

This planner is designed to solve the surgical-needle regrasping problem, where a semicircular needle is initially grasped by one of the patient-side manipulators of the da Vinci® robot with a known grasp transformation between the needle frame and the gripper's tool tip frame $\boldsymbol{G}_{tn}$; the robot must either pass the needle to another arm with an arbitrary final grasp configuration, or change the grasp configuration while leaving the needle in the same arm in which it was held originally. Additionally, the needle must be moved to an arbitrary Cartesian position in the workspace after the completion of one or more such handoffs.

IV. Planner Specification

The planning framework is inherited from RRT – Connect [4]; its algorithm is briefly explained in Algorithm 1.

Algorithm 1 RRT – Connect(hq_{init}, hq_{goal})

1: $\mathcal{T}_a.init(hq_{init})$; $\mathcal{T}_b.init(hq_{goal})$
2: **for** $k = 1$ *to* K **do**
3: $hq_{rand} \leftarrow$ RANDOM_CONFIG()
4: SWAP($\mathcal{T}_a$, $\mathcal{T}_b$)
5: **if** (EXTEND($\mathcal{T}_a$, hq_{rand}) $\neq Trapped$) **then**
6: **if** (CONNECT($\mathcal{T}_b$, hq_{new}) $= Reached$) **then**
7: **return** PATH($\mathcal{T}_a, \mathcal{T}_b$)
8: **end if**
9: **end if**
10: SWAP($\mathcal{T}_a, \mathcal{T}_b$)
11: **end for**
12: **return** failure

Initially, 2 trees are rooted at start state and goal state respectively. Without loss of generality, the start-state tree is designated $\mathcal{T}_a$ and the end-state tree is designated $\mathcal{T}_b$. In the first iteration, a random state $\mathbf{q_{rand}}$ is created, then $\mathcal{T}_a$ is extended by one step-length towards $\mathbf{q_{rand}}$. As long as it is not trapped, $\mathcal{T}_b$ attempts to connect its nearest vertex to $\mathcal{T}_a$'s new vertex. Afterwards, the two tree roles are swapped to make $\mathcal{T}_a$ grow towards $\mathcal{T}_b$ in the next iteration.

A. Notation

In order to reduce the dimensionality of the problem space (similarly to Cohen et al. [9]), two arms are not permitted to move objects simultaneously. This precludes the formation of closed kinematic loops. The arm not holding the object is asssumed to be in the home position whenever the arm that is holding the object moves. However, the resting arm is not excluded from collision checking- this ensures that the moving arm and the needle it holds are guaranteed not to collide with the resting arm.

Some notation related to the planning algorithm is as follows:

- $\mathcal{T}_a$ represents the start-state tree, $\mathcal{T}_b$ represents the end-state tree, and $\mathcal{T}$ represents the combined state tree in the RRT – Connect algorithm.
- $\boldsymbol{hq}$ represents a generic hybrid object state.
- $\boldsymbol{hq}_{rand}$ is a *random* hybrid object state.
- $\boldsymbol{hq}_{start}$ is the *initial* hybrid object state.
- $\boldsymbol{hq}_{goal}$ is the *goal* hybrid object state.
- $\boldsymbol{d}(\boldsymbol{hq})$ is a heuristic distance function between hybrid object states.

B. Grasp

The *grasp transformation* describes how the surgical needle is held by the gripper. It is the position and orientation of the needle frame with respect to the gripper tool tip frame. This relationship is illustrated in greater depth in Liu

[1], which describes criteria by which different grasp transforms can be compared and identifies optimal transforms. A nominal grasp configuration is depicted below, along with the 4 parameters ω_1, ω_2, ω_3, and v_0 which can together completely describe any such configuration. Our planner generates multiple grasp options by iterating exhaustively through a discretized space of these parameters, and stores the resulting rigid-body transforms between the needle and gripper tip in an indexed list.

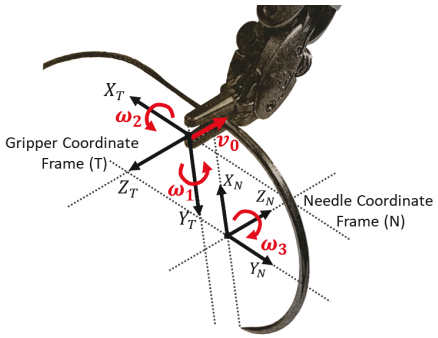


Fig. 1: A nominal needle grasping configuration which describes the relationship between needle frame and gripper frame in the form of four parameters.

C. Definition of Hybrid Object State Space

The hybrid object state space $\boldsymbol{HySS}$ is defined by the combination of the Cartesian-space needle pose $\boldsymbol{S}_{ndl} \in \boldsymbol{SE(3)}$, and 2 discrete state spaces: the grasping arm index space $\boldsymbol{D}_{arm}$ (i.e. whether the needle is grasped by Arm 1 or Arm 2), and the grasp index space $\boldsymbol{D}_{grasp}$ of rigid grasp transforms:

$$\boldsymbol{HySS} = \boldsymbol{S}_{ndl} \times \boldsymbol{D}_{arm} \times \boldsymbol{D}_{grasp} \tag{1}$$

Thus any hybrid object state $hq \in HySS$ can be fully represented as a list composed of 9 parameters:

- The needle's Cartesian position $(x_{ndl},\ y_{ndl},\ z_{ndl})$, by convention measured in meters.
- The needle's Cartesian angular orientation $(qx_{ndl},\ qy_{ndl},\ qz_{ndl},\ qw_{ndl})$, stored as a quaternion.
- The index of the arm in which the needle is grasped, arm_{id}. In the case of the da Vinci®IS1200 surgical robot, the arm index space is $D_{arm} = \{1,2\}$.
- The index of the grasp transform, $grasp_{id}$, corresponding to a rigid grasp transform within the list of generated transforms.

Given an inverse kinematics solver, like TRAC-IK [10], each hq physically reachable by the gripper can be mapped to a robot joint configuration.

Based on this definition, the edge $(hq_i,\ hq_{i+1})$ connecting 2 states is described by one of two different kinds of abstract motions. The first is *object-transfer*, which involves moving the needle to a new position while keeping it held by the same arm with the same grasp. The second is *object-transit*, which involves changing the holding arm and/or the grasp transform without moving the needle in space [11][1]. Object transit requires keeping the needle stationary in Cartesian space *without* assuming any supporting surfaces to be present (rendering the elementary pick-place-pick approach impossible). This is realized by having the needle steadily grasped by (for instance) $PSM1$, then controlling $PSM2$ to grasp the needle, and then releasing the needle from $PSM1$ and sending $PSM1$ back to its home configuration. This is referred to as one "handoff" action. A local planner which can accommodate both types of motion is explained in section IV-F.

D. Heuristic

When the RRT – Connect planner is expanding its trees, the nearest state hq_{near} to the random state hq_{rand} is found by the function $EXTEND(\mathcal{T},\ hq_{rand})$. If the two states are within a step-length of each other, the trees are connected; if not, $interpolate(hq_{near},\ hq_{rand},\ hq_{new})$ generates a new state hq_{new} for hq_{near} to grow towards. As the calculation of distance between hq states is not straightforward, a heuristic distance function is required. The heuristic distance function must be easy to compute and must reflect the fact that object-transit costs more than object-transfer. One such heuristic function is defined as:

$$\boldsymbol{d}_{SE(3)}(\boldsymbol{hq}) = \boldsymbol{d}_{SE(3)} + \boldsymbol{\mu} \cdot \boldsymbol{N} \tag{2}$$

where $\boldsymbol{d}_{SE(3)}$ is the Euclidean distance travelled by the needle in space, and $\boldsymbol{N}$ is the number of handoffs required to connect 2 states that only differ in arm and grasp. As shown by an exhaustive decomposition of the problem in Table I, $\boldsymbol{N}$ will always be either 0, 1, 2, or 3.

TABLE I: Description of Needle Handoff Scenarios

Grasping Arm	Grasped Sector	Num. Handoffs
Goal	**Goal**	0
Non-Goal	*Non-Goal*	1
Goal	*Non-Goal*	2
Non-Goal	**Goal**	3

Note: Derivation of the number of handoffs required to resolve each scenario is shown in Fig. 2.

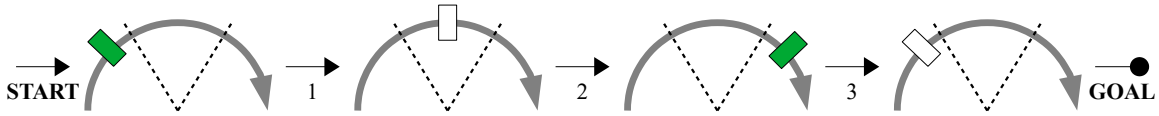
Since $\boldsymbol{N} \in \mathbb{Z}_{\geq 0}$, the parameter $\boldsymbol{\mu}$ has to be larger than the maximum distance (in 6DOF) across the robot workspace in order to indicate the long "distance" involved in performing a handoff motion.

E. Interpolation

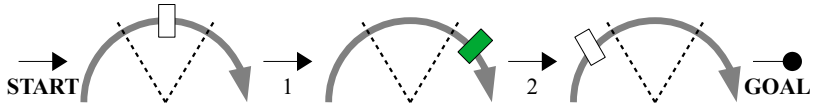
Interpolation must be performed when attempting to connect states $(hq_{near},\ hq_{rand})$ that cannot directly connect, to determine the final position to reach along the path to hq_{rand}. Depending on the needle pose and needle grasp configuration discrepancies between the state pair $(hq_{near},\ hq_{rand})$, a call to $interpolate(hq_{near},\ hq_{rand},\ hq_{new})$ could switch to any of the nine different actions as given in Table II:

Cases where the object pose is changed but the grasp is the same require a geometric interpolation in $SE(3)$; the resulting

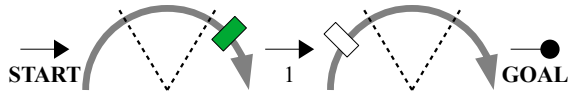
[1]While this terminology seems counterintuitive, it is used in previous works.



(a) 3-Handoff Case: Needle starts out grasped by the non-goal *PSM* in the goal sector. The goal *PSM* must first grasp a non-goal sector so that the non-goal *PSM* can move to the other non-goal sector (and get out of the way), then the goal *PSM* may move back to the goal sector that has just been vacated.



(b) 2-Handoff Case: Needle starts out grasped by the goal *PSM*, but in a different sector from the goal. The non-goal *PSM* must grasp a third sector for the goal *PSM* to move from its current sector to the goal sector.



(c) 1-Handoff Case: Needle starts out grasped by a different *PSM* in a different sector from the goal. The goal *PSM* can grasp the goal sector immediately.

Fig. 2: This figure outlines the conditional logic of the heuristic distance function's handoff count, displaying under what conditions a motion will require 1, 2, or 3 handoffs. Not shown is the trivial case where the needle is already grasped by the goal *PSM* in the goal sector, which requires 0 handoffs. When interpolating between grasps, 2 *PSM*s are forbidden from grasping the same sector of the needle simultaneously. Without loss of generality, the white square represents the gripper of the goal *PSM* and the leftmost sector the goal sector.

TABLE II: Action Depending Upon State Transition Type

Pose	Arm	Grasp	Action
With uniform 40% probability...			Change object pose, keep grasp and arm.
Same	*Same*	*Same*	No action taken.
Same	**Changed**	*Same*	Keep object pose, change grasp and arm.
Same	*Same*	**Changed**	Keep object pose, change grasp and arm.
Changed	*Same*	*Same*	Change object pose, keep grasp and arm.
Same	**Changed**	**Changed**	Keep object pose, change grasp and arm.
Changed	**Changed**	*Same*	Keep object pose, change grasp and arm.
Changed	*Same*	**Changed**	Keep object pose, change grasp and arm.
Changed	**Changed**	**Changed**	Keep object pose, change grasp and arm.

state hq_{new} differs from hq_{near} merely in its pose. In order to connect hq_{new} to hq_{near} in this case, an object-transfer motion will be selected from the local planner. The other cases, except the "No Action" case which simply copies hq_{rand} to hq_{new}, have to interpolate in space $(D_{arm} \times D_{grasp})$, but keep the needle pose the same. To change the grasp and arm values, the program first examines the number of handoffs needed to connect (hq_{near}, hq_{rand}). If this number is greater than 1, the function will pick an intermediate state hq_{interm} which is only one handoff further from hq_{near}. In implementation, the needle is divided into three sectors along its semicircular arc, and two grippers are permitted to grasp different sectors simultaneously but disallowed from grasping the same sector. For example, if hq_{near} has $arm_{id} = 1$ and has grasped Sector 1, the hq_{interm} will have $arm_{id} = 2$ (given only 2 arms), and it can adopt any grasp transform that puts it in contact with Sector 2 or 3, but *not* Sector 1. Only 3 sectors are used in order to reduce the complexity of the handoff calculation, and to guarantee that the robot will indeed grasp the needle in the desired sector and not a neighboring one (in consideration of uncertainties in needle localization and grasping [12] [13]). A comprehensive listing of handoff sequences and their motivation is depicted and described in Fig. 2.

During experimental testing, we observed that branch of a constructed tree may get needlessly trapped in states unfavorable for needle handoff despite it could make progress towards the goal by performing *object-transfer* motions. For instance, the tree could grow into a collision-free state hq_{new} where the needle is held by a *PSM* which is not in collision with any object but is very close to an object, preventing access by the other *PSM* for a needle handoff, while the needle-holding *PSM* is free to move towards the goal by performing an *object-transfer* motion (Fig. 3a). In such a scenario, the naive algorithm would only be able to extend such a branch with very low probability, because, by default, the interpolation algorithm is biased towards a needle hand-off (*object-transit*). This is because probability of randomly generating a hq_{rand} which would lead to an *object-transfer* motion would be 1 / (2·148), given 2 *PSMs* and 148 grasps, while probability of randomly generating a hq_{rand} that would lead to a hand-off would be (2·148 - 1) / (2·148). In order to balance the extension of the tree through *object-transfer* and *object-transit*, the algorithm is modified to assign a distinct, 40%, probability to perform *object-transfer*.

F. State Sampling, Validity Checker and Local Planner

In order to improve performance of the state generator, a random state hq_{rand} is generated by sampling uniformly from the robot joint space (within the joint limits in order to guarantee to a tool-tip-reachable pose). Then, values are sampled uniformly for $arm_{id} \in D_{arm}$ and $grasp_{id} \in D_{arm}$ respectively. Since each grasp index $grasp_{id}$ corresponds to a specific grasp transformation, a random needle pose S_{ndl} with respect to reference frame is:

$$\boldsymbol{G}_{rn} = \boldsymbol{G}_{rt} \cdot \boldsymbol{G}_{tn} \tag{3}$$

G_{rt} and G_{tn} represent the transformation from tool tip to reference frame, and from needle to tool tip frame, respectively. Instead of directly sampling in object space and then using inverse kinematics to get a corresponding joint position, the forward method relies on forward kinematics, which takes less time to compute. This also avoids having the state validity checker repeatedly examine IK solutions from many hq_{rand}.

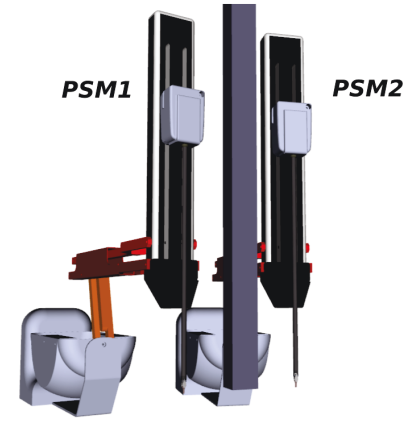
To create collision-free regrasping paths, a validity checker and local planner are implemented. The validity checker checks if sample state hq_{rand} causes any collisions, and if the state hq_{new} from the *interpolate* function is IK-solvable and collision-free. Upon completion, the local planner decides which motion is used to connect (hq_{near}, hq_{new}), and checks collisions for each incremental motion along the resulting path.

V. Validation

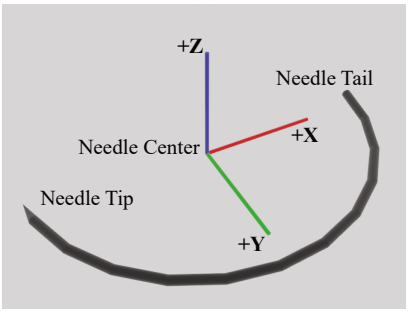
A simulation-based validation of the proposed method was carried out in a simulated environment [14] which included a semicircular needle model with radius $r = 12mm$, 2 *PSM*s, and an simplified column endoscopic camera, shown as Fig. 3a. The hybrid-state-space RRT – Connect motion planner was implemented using the Open Motion Planning Library (OMPL) [15].

A. Example Needle Regrasp Plan

In order to demonstrate the operation of the algorithm, an example multi-arm grasping task is presented in this section. When given a start state hq_{start} with *PSM*2 holding the needle tail as shown in Fig. 4a, and goal state hq_{goal} with *PSM*1 holding the needle tail optimally as shown in Fig. 4h, *PSM*2 successfully passed the needle to *PSM*1 after three handoffs. From the *PSM*2-grasped initial state, the first handoff motion was executed as shown in Fig. 4b after two object-transfer steps (corresponding to motion 2 and 3 in Table III). The needle was grasped by both arms simultaneously, (*PSM*1 held the needle's middle arc, as per the grasping choice routine which forbids it from grasping in the same sector as *PSM*2). The subsequent handoff motions listed in Table III were then executed sequentially, and intermediate arm and grasp choices obeyed the *interpolate* function, eventually leaving *PSM*1 holding the needle with the optimal grasp transformation.



(a)



(b)

Fig. 3: Needle regrasping simulation environment setup. (a) The two Patient Side Manipulators and the endoscopic camera of the da Vinci® Surgical Robot simulation model. The camera holding arm is not shown to simplify the diagram. (b) Simulated needle model.

TABLE III: Solution Path Information

Motion	Motion Type	$(arm_{id}, grasp_{id})$
1	Initial Grasp	(2, 0)
2	Transfer	(2, 0)
3	Transfer	(2, 0)
4	Transit	(1, 24)
5	Transit	(2, 145)
6	Transfer	(2, 145)
7	Transit	(1, 147)

The RRT – Connect trees constructed by the planner were composed of a of total 16 nodes and 15 edges, with the solution path containing 7 nodes and 6 edges executed in motions 1 – 7 (excluding the initial grasp). The total planning time was 8.08 seconds. The solution is summarized in Table III; for conciseness, the needle pose is not listed.

B. Performance

Performance testing focused on the computational efficiency of the algorithm, namely, how quickly the hybrid-state-space based RRT – Connect motion planner could find a collision free solution path given start state hq_{start} and goal state hq_{goal}. For each test case, 100 trial (hq_{start}, hq_{goal}) pairs were fed into the planner, producing the results listed in Table IV.

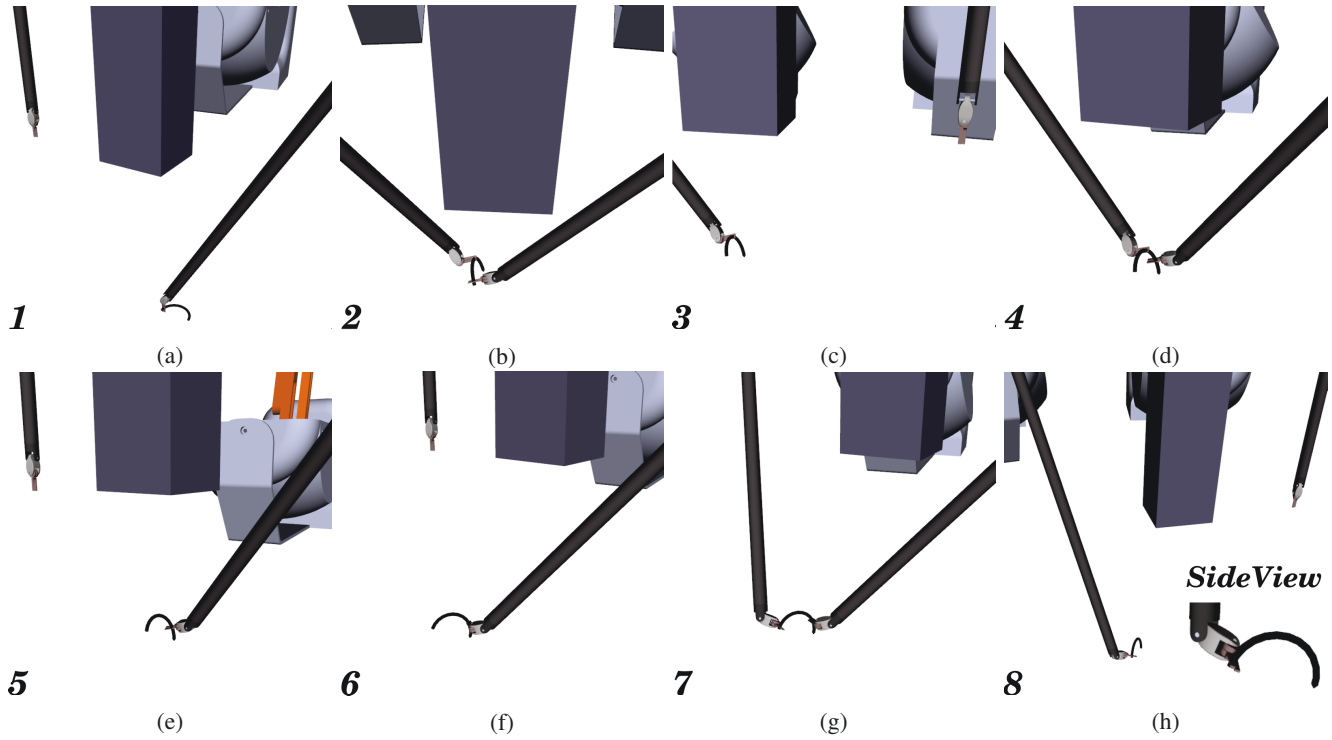


Fig. 4: Needle regrasping experiment with three handoffs and the needle successfully held by *PSM*1 with optimal grasp. This is a three-handoff situation since the needle is grasped by *PSM*2 at the start in the same location where *PSM*1 will grasp it in the end; thus, three handoffs are required since two *PSM*s cannot grasp the needle at adjacent locations.

TABLE IV: Performance Testing

Test Case	Num. Success	Ave. Time (s)	Std. Dev (s)	Min Time (s)	Max Time (s)
One Handoff	100	11.79	10.97	1.50	51.28
Two Handoffs	100	11.93	10.10	2.20	50.92
Three Handoffs	100	14.83	12.59	2.64	73.98

Note: For each test case, 100 tests were performed.

For each trial state, the needle pose was randomly sampled within the starting *PSM*'s estimated workspace; the values of the discrete variables $(arm_{id}, grasp_{id})$ were randomly sampled with the expectation that they would be solvable after at least a certain number of handoffs (one, two, or three).

The statistics listed in Table IV indicate that the proposed method was able to successfully and effectively generate collision free plans that allowed the needle to be regrasped at the desired needle grasp pose. No failures occurred out of a total of three hundred tests. The average running times of the first two tests were very close, around 0.2s difference. The three-handoffs case had the longest average time; this is expected as a longer planning time is required to accommodate more handoffs. The same consequence can be seen in the minimum running time across all cases. Since RRT – Connect is a probabilistic algorithm, large differences between the minimum and maximum running times are to be expected.

VI. Conclusions

Building upon the RRT – Connect motion planner, we address the needle regrasping manipulation task by allowing the two arms of the da Vinci® Surgical Unit to collaborate with each other. Planning in hybrid space is a non-trivial question, and to resolve it we propose a hybrid object state space accompanied by a corresponding heuristic distance measure capable of evaluating both continuous Cartesian space transitions and discrete grasp space transitions simultaneously. We also propose an interpolation algorithm which directs a local planner to compute motion choices between states- thus, two explorational trees in RRT – Connect are realized. Simulation experiments demonstrated a typical needle regrasping problem, in which the planner controlled the robot to finish the handoffs and move the needle to the goal pose. We also verified the runtime performance of our algorithm. With arbitrary initial and goal states a total of 300 tests were conducted with zero failures, suggesting that the planner can reliably find solutions.

In our future work, we will focus on physical-robot execution. Additionally, though collision-free regrasping paths are reliably generated, there exist redundant path segments (such as motion 2 listed in Table III) which could be eliminated by path optimization in hybrid state space. Another focus of future work will be the improvement of the computational efficiency of the proposed algorithm, including parallelization of the code.

Software is available online at [16].

REFERENCES

[1] T. Liu and M. C. Cavusoglu, "Needle grasp and entry port selection for automatic execution of suturing tasks in robotic minimally invasive surgery," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 552–563, 2016.

[2] T. Siméon, J. Cortés, A. Sahbani, and J.-P. Laumond, "A general manipulation task planner," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 311–327.

[3] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[4] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[5] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation—a survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[6] H. Marino, M. Ferrati, A. Settimi, C. Rosales, and M. Gabiccini, "On the problem of moving objects with autonomous robots: A unifying high-level planning approach," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 469–476, 2016.

[7] J.-P. Saut, M. Gharbi, J. Cortés, D. Sidobre, and T. Siméon, "Planning pick-and-place tasks with two-hand regrasping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4528–4533.

[8] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan, "Rrts for nonlinear, discrete, and hybrid planning and control," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 1. IEEE, 2003, pp. 657–663.

[9] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[10] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 928–935.

[11] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 945–952.

[12] O. Özgüner, R. Hao, R. C. Jackson, T. Shkurti, W. Newman, and M. C. Cavusoglu, "Three-dimensional surgical needle localization and tracking using stereo endoscopic image streams," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6617–6624.

[13] O. Özgüner, T. Shkurti, S. Lu, W. Newman, and M. C. Cavusoglu, "Visually guided needle driving and pull for autonomous suturing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, under review.

[14] T. Shkurti, "Simulation and control enhancements for the da Vinci surgical robot," Master's thesis, Case Western Reserve University, 5 2019.

[15] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[16] S. Lu, "Needle handoff," https://github.com/lusu8892/cwru_davinci_moveit, 2019.