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

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

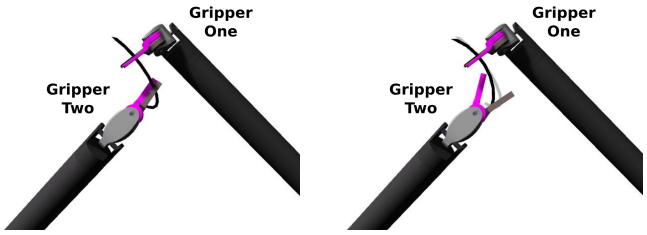
Abstract— **This paper proposes a path correction method for surgical robotic systems performing needle handoff manipulations as part of autonomous execution of surgical suturing. During handoff motions, the position and orientation of the needle is subject to perturbations from the idealized planned pose due to uncertainties in camera-robot calibration and needle localization. If, after a perturbation, the system needs to perform subsequent needle regrasp(s), but the robot still follows the originally planned trajectory out of the path planner [1], it has a lower chance of gripping the needle properly. In order to accommodate these unpredictable needle pose perturbations, the proposed path correction method works locally to direct the needle from the wrong pose to the original pose by partial replanning of the robot motion. The reliability of the proposed method is evaluated with three sets of experiments in a simulation environment.**

Index Terms— **Surgical Robotics: Laparoscopy; Medical Robots and Systems**

I. INTRODUCTION

In minimally-invasive surgical suturing, small, semicircular needles are introduced into the body cavity at arbitrary locations and must be grasped near the base before being driven through a specific section of tissue in a predetermined arc. Since the initial grasp of the needle may not be ideal [2] for driving it in the required suturing location, handoffs from one minimally-invasive surgical tool to another may need to be performed. Thus, dual-arm collaborative object manipulation [3] becomes of interest to surgical robotics applications, with needle handoffs needing to be performed between two patient-side manipulators (*PSM*s).

Physical experiments reveal that during the execution of the planned needle regrasping manipulations, the needle pose is subject to perturbations from its original planned pose during the handoff. This perturbation is due to uncertainties in camera-robot calibration and needle localization, as well as physical deformation and shifting of the needle by the robot. Small errors in how the grippers are holding the needle due to small calibration or needle localization errors lead to competing internal forces between the initial grasp and the new regrasp, which cannot be explicitly modeled or measured without force sensing. For instance, when the first gripper grips the needle while the second gripper is trying to release its grip off of the needle (Fig. 1a), the needle (which is slightly elastic and under torsion) may suddenly snap to a new unplanned pose different than the originally planned one (Fig. 1b) as soon as the second gripper opens its jaws. As a result, the second gripper ends up holding the needle with a grasp configuration different from what was in the original plan. The specific nature of this perturbation is highly unpredictable, but its occurrence is almost inevitable. Therefore proper compensation of needle pose perturbations is critical for successful completion of multi-step regrasp sequences.



a) b)

Fig. 1: A grasped needle is perturbed to a new pose after a needle-handoff transition from a) to b), which results in Gripper One holding the needle in a perturbed pose (shown as the bold curve in solid black) as opposed to the desired pose (shown as the bold curve in light grey).

In this paper, a path correction method based on a vision-based needle tracker [4] is proposed to overcome the uncertainties resulting from the execution of the needle regrasp manipulations. The rest of this paper is organized as follows. The dual-arm needle regrasping manipulation planner is briefly illustrated in Section III. The problem definition is given in Section IV. The path correction algorithm is described in Section V. Results of the simulation experiments are presented in Section VI, with concluding remarks in Section VII.

II. RELATED WORK

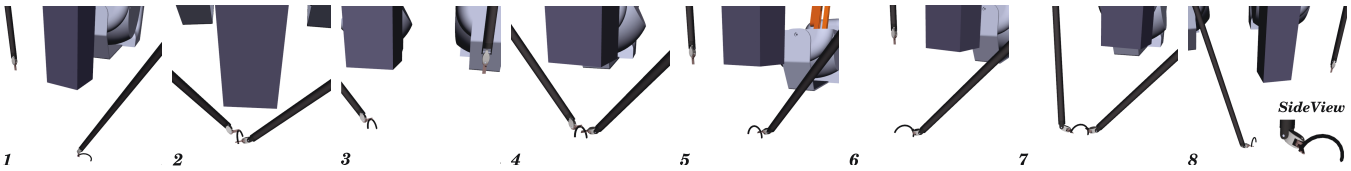
Studies to investigate dual-arm object manipulation have been carried out since the 1950's. By utilizing two arms, a robot can perform more complex manipulations which cannot be performed by a single manipulator [5], [6], such as a jar-opening tasks where one hand securely holds the jar body while the other hand unscrews the cap [7].

Recent developments in higher level planning, advanced control, perception, and learning by demonstration have led

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



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 2: Demonstration of needle regrasp between two *PSM*s. The solution path successfully led to the needle being regrasped by *PSM*1 (the arm on the left) with the optimal grasp [2], after three handoff motions.

to new advances in dual-arm manipulation [8]. A geometrically consistent 6-DOF control scheme is adapted to dual-arm cooperative manipulators by Caccavale et al. [9]. Surdilovic et al. [10] proposed a framework used to synthesize compliance control and dual-arm manipulation planning, which successfully realized a bimanual parts assembly task. Vahrenkamp et al. [11] investigated the problems of solving inverse kinematics and motion planning for dual-arm object regrasping, and two Rapidly-Exploring Random Tree (RRT) based planning algorithms were proposed to calculate regrasping manipulation paths. Unlike the traditional sampling-based or search-based planning methods applied to solve dual-arm manipulation problems [12], [13], a learning-based algorithm relies on the classification and segmentation of the coordinated bimanual actions for a task planner to plan the sequence of motion for decomposed tasks [7]. Jagersand et al. propose an experimental framework to quantitatively evaluate the precision improvements of visual servoing in a variety of environments featuring variable occlusion and kinematic obstacles [14]. Hynes et al. develop robust visual tool-tracking of multiple arms in a surgical environment [15], and subsequently employ uncalibrated visual servoing and a pair of collaborating robot arms to realize suture knot-tying within a physical tissue phantom [16]. Kruse et al. develop a visually-guided precision teleoperation system for two-arm industrial robots, autonomously translating human hand motions without force feedback into stable grasping of unknown objects [17]. Finally, Qu et al. examine the problem of uncertain grasp transforms when performing cyclical object handoffs in an industrial setting, and implement a vision-based cascade control structure based on a radial basis function neural network [18]. Their later work then employs marker fiducials and visual path planning to execute high-precision dual-arm motions in environments of very high initial position and rotation uncertainty [19].

III. Dual-Arm Needle Regrasping Manipulation Planner

Our previous work [1] presented a hybrid state space RRT algorithm to plan bimanual needle handoff motions between two *PSM*s of the da Vinci® robot, to address the needle regrasping planning problem abstracted in (discrete+continuous) hybrid state space.

The hybrid 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 [1]:

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

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-handoff*, which involves changing the holding arm and/or the grasp transform without moving the needle in space[1].

The solution path generated is a sequence of motions composed of *object-transfer* motion and *object-handoff* motion primitives. An example of a planned path for needle regrasping is given in Fig. 2. The execution of this demonstration assumes that the needle pose always remains accurate to the planned results, and the grasp transformation is kept static throughout each handoff. However these two assumptions do not hold in hardware because of needle perturbations that occur.

IV. Problem Definition

Given the planned needle regrasping trajectory generated from the handoff planner mentioned in Section III, the robot should accommodate perturbations by replanning, to direct the needle from the perturbed pose to the planned pose, such that the needle can be brought from initial state hq_{start} to the goal state hq_{goal} without any collisions.

Some notation related to the path correction method is as follows:

- $\boldsymbol{hq}$ 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.
- $\boldsymbol{hq}_{pert}$ is the perturbed hybrid object state.
- $\boldsymbol{hq}_{target}$ is the next hybrid state along the sequence of states generated by the RRT algorithm.
- $\boldsymbol{hq}_{cur\_ideal}$ is the state that the needle would be in if there was no perturbation.

V. Path Correction

The proposed path correction method employs *local-path-correction* (*LPC*). The needle regrasping paths generated by the algorithm presented in Section III are a combination

[1]This terminology differs from the standard "object transfer and object transit" terminology outlined in [20]. We have chosen to diverge from the terminology of general dual-arm manipulation to more clearly describe this specific problem.

Algorithm 1: CorrectTransfer()

Input: hq_{pert}, hq_{target}

```
while ‖ S_ndl ∈ hq_pert, S_ndl ∈ hq_target ‖ > ε do
    NewPath ← CalcCartesianPath(hq_pert, hq_target)
    if IsValid(NewPath) then
        ExecuteTrajectory(NewPath)
    else
        if hq_target ≠ hq_goal then
            return Succeeded
        else
            return Failure
        end
    end
    hq_pert ← UpdateNeedlePose()
end
```

Algorithm 2: CorrectHandoff()

Input: hq_{pert}, hq_{target}, hq_{cur_ideal}, $NewPath$

```
1  SameNdl ← ‖ S_ndl ∈ hq_pert, S_ndl ∈ hq_target ‖ < ε
2  SameRS ← RobotJointDist(hq_cur_ideal, hq_pert) < ε
3  if SameNdl ∧ SameRS then
4      NewPath = OriginalPath
5      return True
6  end
7  hq_temp ← hq_target
8  S_ndl ∈ hq_temp = S_ndl ∈ hq_pert
9  if ¬ ReplanHandoffPath(hq_pert, hq_temp, NewPath) then
10     if ¬ ValidPath(hq_pert, OriginalGraspPath) then
11         return Failure
12     else
13         if ¬ GenerateNewReleasePath(hq_pert) then
14             return Failure
15         end
16     end
17     NewPath = OriginalGraspPath + NewReleasePath
18     return True
19 end
20 return True
```

of several *object-transfer* and *object-handoff* path segments. The *LPC* method aims to correct each of these path segments in a local fashion.

A. Local Path Correction

As noted earlier, the needle is subject to perturbations after two grippers finish a handoff motion. As a result, the current needle pose and grasp may deviate from that of the original planned state provided by the planning algorithm. At this point, the next path segment should be replanned based on the perturbed needle pose and grasp (as reported by the vision-based needle tracking algorithm [4]), such that the new path steers the needle back on course. The replanning is determined by the next trajectory only, which means that no matter how many states are left to be visited, the replanning is only being done from the current perturbed state to the adjacent target state in order to avoid the additional computation needed for a full replanning of the needle regrasp. As the replanned new path only tries to connect two neighboring states, the rest of the path(s) is left untouched. This is why the method is called "local".

Given the current perturbed hybrid object state $hq_{pert} \in HySS$ that has a different needle pose and grasp than that of the current ideal state $hq_{cur\_ideal} \in HySS$, and the neighboring target state $hq_{target} \in HySS$, the *LPC* plans a local collision-free path according to the type of the next path segment.

Algorithm 1 summarizes the method used for local path correction when the next segment of the path is an *object-transfer* path. If the distance between the states hq_{pert} and hq_{target} is larger than a user specified threshold ε, the algorithm generates a new Cartesian straight-line path to connect hq_{pert} with hq_{target}, such that the needle is moved from the perturbed pose towards the next target pose with the same supporting arm but with a different motion to accommodate the incorrect grasp. Inside the *CalcCartesianPath* function, the full path connecting hq_{pert} and hq_{target} is divided into several steps, but $NewPath$ only stores the first segment (which transfers the needle from the current state hq_{pert} to the adjacent state hq_{adj}). This is because, under the visually guided system, the farther the distance traveled by the needle, the more error the needle pose would accumulate. Therefore, in order to minimize the error, the needle is instead moved towards the target pose for a certain distance at each iteration. As long as the target state hq_{target} is not the goal state hq_{goal}, the algorithm will still exit reporting *Succeeded* to the outer layer of the program regardless of value evaluated from $IsValid(NewPath)$, since the remainder of the error can potentially be compensated for by the rest of handoff manipulation path. The idea is that even if the algorithm has failed at progressing towards the temporary target, the program still has a chance to correct the rest of the paths to bring the needle to the goal state hq_{goal}. If hq_{target} equals hq_{goal} then the program will have no further chance to do more local corrections and could only report *Failure* as shown in Line 8. In summary the new *object-transfer* path will only bring the needle as close as possible to the target pose with the same *PSM* arm holding needle as in hq_{target}, however, cannot guarantee that the grasp transform will be the same as that of hq_{target}.

The method used for local path correction when the next segment is an *object-handoff* path is presented as Algorithm 2. The algorithm first examines if the observed perturbation of the needle is small enough to assume that the needle is gripped as planned. If so, the original plan is acceptable. If not, the algorithm attempts to plan a new handoff path $NewPath$ from the perturbed state hq_{pert} to the temporary state hq_{temp} such that the new supporting arm can hold the needle with the grasp specified in hq_{target}. For instance, consider the situation where the needle needs to be passed from *PSM*1 to *PSM*2, but, the current ideal state hq_{cur_ideal} has a different needle pose and grasp from that of

the current perturbed state hq_{pert}. In order to direct *PSM*2 to grasp the needle as defined by the configuration hq_{target}, a new *object-handoff* path has to be generated. This new path needs to compensate for the change of the needle pose so that *PSM*2 correctly grasps the needle. *PSM*1 then releases the grip and retreats back according to the new configuration of *PSM*2. If any of the motions described above are not both kinematically feasible and collision-free, then the algorithm will attempt to keep using the original grasping path to validate if *PSM*2 could correctly grasp the needle without any collision. If not, then the program reports *Failure* and exits. If the original grasping path is acceptable, then a new grip release path for *PSM*1 is generated according to the perturbed needle pose. It is important to note that the *CorrectHandoff* algorithm only tries to correct the needle grasp to the value given in the target state hq_{target} but does not transfer the needle to the target pose.

The local path correction strategy results in either a new *object-transfer* path or a new *object-handoff* path that attempts to correct the needle pose or correct the needle grasp, respectively, from the perturbed state hq_{pert} to the adjacent target state hq_{target}. Consequently, the entire needle-handoff path is locally corrected, one path segment at a time.

B. Path Correction Algorithm

The *LPC* scheme described in Section V is incorporated into a high-level path correction algorithm as presented in Algorithm 3 to accommodate uncertainties during the execution of the needle-handoff manipulation trajectory generated by the Dual-Arm Needle Regrasping Manipulation Planner (Section III).

The input of the algorithm is the originally planned handoff manipulation trajectory $\boldsymbol{PT}$ that is a list of trajectories sequentially composed of *object-transfer* and *object-handoff* motions. If the k^{th} trajectory is an *object-transfer* path, then the algorithm tries to find a new path by Algorithm 1 to try to direct the needle from the perturbed pose to the next target pose. If the k^{th} trajectory is a *object-handoff* path, then algorithm tries to plan a new handoff path by Algorithm 2. If successful, the chosen *PSM* in $hq_{target_{k+1}}$ will correctly grasp the needle at the pose given in hq_{pert_k}. After the execution of the new trajectory, the k^{th} state hq_{pert_k} should have same arm_{id} and $grasp_{id}$ as $hq_{target_{k+1}}$ but a different needle pose, which means the needle is already held by the right *PSM* and the desired grasp configuration. The algorithm will then compensate for the perturbation by moving the needle to the target pose in $hq_{target_{k+1}}$ as shown from Line 13 by Algorithm 3. If this failed and no more trajectory segments are left in $\boldsymbol{PT}$, then the program reports *False*, which means that a global replanning of the regrasping manipulation needs to be performed.

Algorithm 3: DualArmNeedleManipulation(PT)

```
1  N ← length(PT)
2  for k ← 1 to N do
3      hq_pert ← hq_target_k
4      hq_pert ← UpdateNeedlePose()
5      if PT_k = object-transfer-path then
6          if ¬ CorrectTransfer(hq_pert_k, hq_target_k+1) then
7              return False
8          end
9      end
10     if CorrectHandoff(hq_pert_k, hq_target_k+1, NewPath_k)
         then
11         ExecuteTrajectory(NewPath_k)
12         hq_pert_k ← UpdateState()
13         if ¬ CorrectTransfer(hq_pert_k, hq_target_k+1) then
14             if k = N then
15                 return False
16             end
17         end
18     else
19         return False
20     end
21 end
22 return True
```

VI. Validation

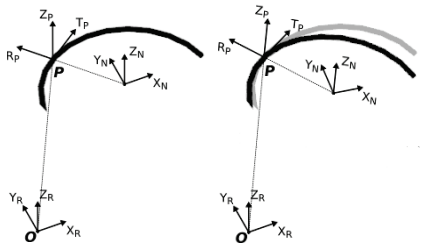
In this section, the simulation experiments conducted to validate the performance of the path correction algorithm are presented. In the simulation experiment the needle is placed at a random pose and with a random choice of grasp configuration from $\boldsymbol{D}_{grasp}$, the algorithm is asked to find solution to bring the needle to a different random goal state. Two kinds of system noises are introduced in a Gazebo-based dVRK simulator [21] to emulate perturbations to needle pose and errors in tracking of the needle pose. The first experiment does not perform any path correction to establish the baseline performance for comparison, while the second and the third experiments use Algorithm 3 to validate the reliability of *LPC* method. The first two experiments use the true needle pose data received directly from the simulator, evaluating performance under accurate visual needle tracking conditions, while the third experiment uses noisy needle pose estimates to evaluate the robustness of the path correction algorithm to needle tracking errors.

A. Error Injection in Simulation

When validating the needle path correction algorithm, artificial noise is injected to the simulated needle pose in the Gazebo simulator to mimic the physical behavior of the real needle. It is important to know that this simulated perturbation is not based on mathematical modeling of the underlying physics causing the random pose alteration of the needle; only the end result of the visible sudden pose change of the needle.[2]

The artificial perturbation is simulated by a random rotation, which is obtained by rotating the needle about an

[2]In the simulations, the needle pose perturbation is assumed, without loss of generality, not to happen in the last handoff motion of the whole manipulation trajectory. Otherwise, the gripper would never correctly grasp the needle, which would lead to an infinite loop of path corrections.



a) Original Pose b) After Perturbation

Fig. 3: The diagram describes the transform performed to add perturbation to the true needle pose.

axis through grasp point at $\boldsymbol{P} \in \mathbb{R}^3$ as shown in Fig. 3. In the physical experiments, most of the perturbations were observed to be similar to rotating the needle around the vertical axis $\boldsymbol{Z}_p$. Therefore, across the experiments, the perturbation is generated with 50% probability as a rotation around the vertical axis $\boldsymbol{Z}_p$, 25% probability around the radial axis $\boldsymbol{R}_p$ and 25% probability around the tangential axis $\boldsymbol{T}_p$. The angle of rotation dictates the magnitude of the perturbation, which is applied at various grades.

The second type of noise is the object tracking error. This error stems from matching errors in the tracking program, as well as any errors in camera calibration and errors resulting from finite camera resolution [4]. Object tracking error does not modify the actual pose of the simulated needle, it only perturbs the pose information reported by the needle tracking algorithm to the needle-handoff algorithm. As a result, the program guides the *PSM* to an inaccurate grasp pose, which could lead to either an incorrect grasp or missing the needle. Object tracking error is emulated by fetching the accurate needle pose from the simulator, then perturbing it before reporting the result to the path correction algorithm. Both translational and rotational errors were included.

B. Experiment I

The first experiment demonstrates the baseline performance of the execution of dual-arm needle manipulation under perturbation to needle pose but without the help of the correction algorithm. The perturbation error magnitude was set to $0.2rad$. Out of 10 trials, only 2 cases succeeded. All failed cases exhibited the same issue: once the needle pose was perturbed, insisting on the original manipulation trajectory led to the manipulator missing the needle during subsequent regrasps.

C. Experiment II

In order to validate the ability of the *LPC* algorithm to handle the perturbations experienced during handoff motions, the second experiment was conducted with various levels of perturbation injected. 20 trials were performed for each error magnitude, and both positive and negative perturbations were considered. The results are presented in Table I. In order to demonstrate the performance improvement realized by the path correction algorithm, the percentage points (%-points) of improvement over the control group is given in the last column of the Table I.

TABLE I: Path Correction Performance

Pert. (*rad*)	Successes	Success Rate (%)	Performance Change (%-points)
0.2 ± 0.01	18	90	70
0.3 ± 0.01	18	90	70
0.4 ± 0.01	16	80	60
0.5 ± 0.01	14	70	50

Note: For each level of error, 20 trials were performed.

As shown in Table I, the *LPC* algorithm accommodates the perturbations well. With low-level perturbation at $0.2rad$ and $0.3rad$, the success rates are brought up to 90%, which gives a substantial improvement over the control group. Even at the larger magnitudes, $0.4rad$ and $0.5rad$, the success rate was maintained at 80% and 70%, respectively. The cause of all the failures is that the *LPC* algorithm failed at replanning either the local *object-transfer* path or the *object-handoff* path.

An example of dual-arm needle manipulation plan is shown in Fig. 4. The manipulation plan is composed of a total of 8 states, sequentially connected by either *object-transfer* or *object-handoff* trajectory. In order to ensure that the needle can be grasped by *PSM*1 with the selected grasp, the *LPC* algorithm is used during at the execution of the trajectories. The Fig. 4 demonstrates the trajectory execution with the involvement of *LPC*. During the execution, each connecting trajectory is corrected by the *LPC* algorithm. After three hand-offs, the needle is positioned at the goal pose held by *PSM*1 with the optimal grasp.

D. Experiment III

The last experiment demonstrates the tolerance of the needle regrasp plans to the noisy needle pose information received from the tracking system. Specifically, the goal is to determine the level of noise where *LPC* fails at assisting the handoff manipulation. For each case, 20 trials were performed with the magnitude of error represented by a Gaussian distribution with standard deviation as $0.001rad$ for orientation noise and $0.1mm$ for position error. The magnitude of the needle pose perturbation was set as $\pm 0.2 \pm 0.01rad$ for all tests. The results are separately summarized in Table II and Table III. The last column of each shows the performance difference compared with the row of Table I that corresponds to the test case with the same level of needle pose perturbation but without noisy tracking information.

According to the outcomes in Table II, the *LPC* algorithm is not adversely affected by the noise around $0.01rad$, which reaches a 95% success rate out of 20 trials, a performance level similar to the first test case in Experiment II. However with the magnitude of noise increased to $0.03rad$ and $0.07rad$, the success rate decreases to only $60\% - 65\%$. When compared to the result of the test case with the accurate needle pose tracking, the performance dropped

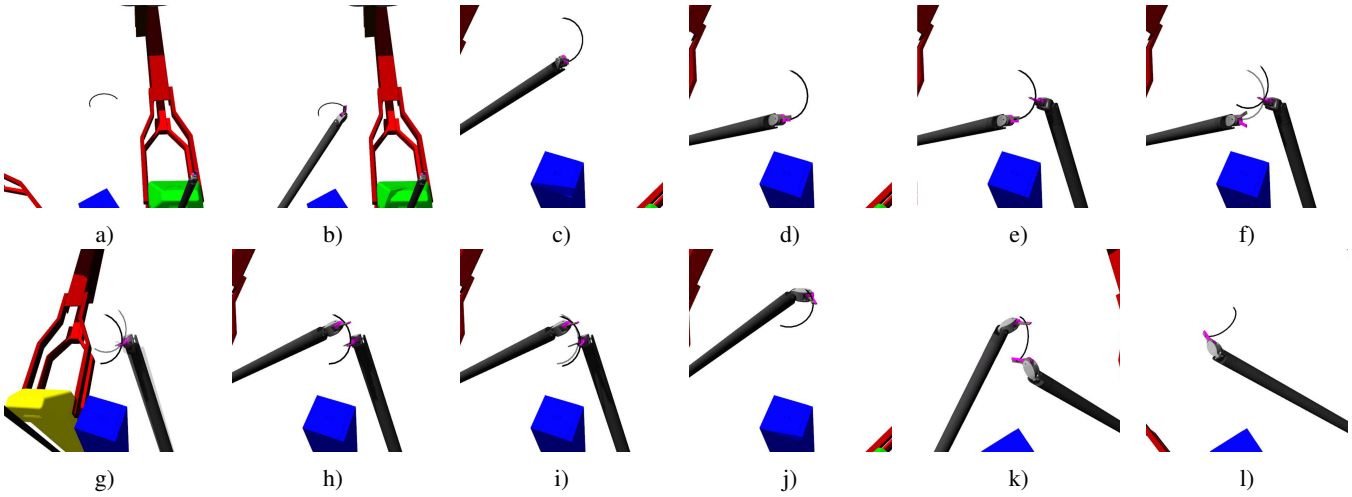


Fig. 4: The diagram describes an example execution of dual-arm needle manipulation with *LPC*. The needle is handed off between *PSM*s three times with several *object-transfer* motions in between. The perturbation happens at the first handoff motion, from e) to f), and the second handoff motion, from h) to i).

by 25 to 30 percentage points. The reason of the failed trials is that the manipulator was deceived into adopting a disadvantageous grasp pose where the needle was insecurely grasped. For example, the needle was grasped too close to the gripper's tip, or the needle base or tip was barely attached to the gripper.

As can be observed in the results listed in Table III, the *LPC* algorithm is more strongly influenced by the noise along the depth perceived by the camera. Even a $1mm$ error could make the performance drop by 20% as compared to the ground truth case. This 20% performance decrease consists of 3 failures where the gripper missed the grasp of the needle at handoff. At $2mm$ noise level, the success rate is slightly decreased to 60%, but more failures where the needle grasp is missed contributed to the number of failed trials. With noise increased to $3mm$, the performance of the *LPC* is further reduced where only 25% of the tests could succeed.

Based on the last three experiments, we conclude the *LPC* algorithm successfully handles perturbation at the magnitude of $0.2rad - 0.3rad$ with good performance. In terms of tolerance to errors in needle tracking, we can expect the *LPC* algorithm to be effective at dealing with the perturbations when the orientation noise is smaller than $0.03rad$ and the position noise is at most $1mm$.

TABLE II: Tolerance to Orientation Noise

Level of Noise (rad)	Successes	Success Rate (%)	Performance Change (%-points)
0.01	19(1**F**)	95	+5
0.03	13(2**F**, 5**I**)	65	-25
0.07	12(1**F**, 7**I**)	60	-30

Note: The results in the second column also indicate the reason for failure with the additional information in the parenthesis where **F** and **I** represent the failure of *LPC* and insecure grasp, respectively.

VII. Conclusion

The algorithm presented in this paper focuses on how to handle the perturbations in needle pose that result during needle-handoff using a *local-path-correction* (*LPC*) strategy. The *LPC* strategy adjusts the individual *object-transfer* and *object-handoff* segments of the original needle regrasping plan to compensate for perturbations to the needle position. The validation of the *LPC* algorithm was performed by experiments conducted in a simulated environment, with artificial random perturbations. The *LPC* algorithm resulted in performance improvement of $50-70$ percentage points as compared to the baseline performance. Investigation of tolerance of the *LPC* algorithm to errors in needle pose estimation showed the *LPC* algorithm to be effective at dealing with perturbations when orientation measurement noise is smaller than $0.03rad$ and position measurement noise is at most $1mm$.

Future work will focus on the validation of the path correction method on the physical dVRK setup. This demands sufficient needle pose estimation accuracy from the needle tracking system as well as sufficiently high camera-robot calibration, (or robot tool localization and tracking capability).

TABLE III: Tolerance to Position Error

Noise (mm)	Successes	Success Rate (%)	Performance Change (%-points)
1	14(3**F**, 3**M**)	70	-20
2	12(3**F**, 5**M**)	60	-30
3	16(1**F**, 14**M**)	25	-65

Note: The results in the second column also indicate the reason for failure with the additional information in the parenthesis where **F** and **M** represent the failure of *LPC* and missing-of-needle-grasp, respectively.

REFERENCES

[1] S. Lu, T. Shkurti, and M. C. Cavusoglu, "Dual Arm Needle Manipulation on daVinci Surgical Robot," in *Proceedings of the 2020 International Symposium on Medical Robotics (ISMR)*, 2020, (In Press).

[2] 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.

[3] 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.

[4] 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.

[5] A. Edsinger and C. C. Kemp, "Two arms are better than one: A behavior based control system for assistive bimanual manipulation," in *Recent progress in robotics: Viable robotic service to human*. Springer, 2007, pp. 345–355.

[6] ——, "Human-robot interaction for cooperative manipulation: Handing objects to one another," in *RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2007, pp. 1167–1172.

[7] R. Zollner, T. Asfour, and R. Dillmann, "Programming by demonstration: dual-arm manipulation tasks for humanoid robots," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1. IEEE, 2004, pp. 479–484.

[8] P. Ögren, C. Smith, Y. Karayiannidis, and D. Kragic, "A multi objective control approach to online dual arm manipulation1," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 747–752, 2012.

[9] F. Caccavale, P. Chiacchio, A. Marino, and L. Villani, "Six-dof impedance control of dual-arm cooperative manipulators," *IEEE/ASME Transactions On Mechatronics*, vol. 13, no. 5, pp. 576–586, 2008.

[10] D. Surdilovic, Y. Yakut, T. M. Nguyen, X. B. Pham, A. Vick, and R. Martin-Martin, "Compliance control with dual-arm humanoid robots: Design, planning and programming," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2010, pp. 275–281.

[11] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 2464–2470.

[12] 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.

[13] B. Cohen, S. Chitta, and M. Likhachev, "Single-and dual-arm motion planning with heuristic search," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 305–320, 2014.

[14] M. Jagersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. IEEE, 1997, pp. 2874–2880.

[15] P. Hynes, G. I. Dodds, and A. Wilkinson, "Uncalibrated visual-servoing of a dual-arm robot for surgical tasks," in *2005 International Symposium on Computational Intelligence in Robotics and Automation*. IEEE, 2005, pp. 151–156.

[16] P. Hynes, G. Dodds, and A. Wilkinson, "Uncalibrated visual-servoing of a dual-arm robot for mis suturing," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.* IEEE, 2006, pp. 420–425.

[17] D. Kruse, J. T. Wen, and R. J. Radke, "A sensor-based dual-arm tele-robotic system," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 4–18, 2014.

[18] J. Qu, F. Zhang, Y. Fu, G. Li, and S. Guo, "Adaptive neural network visual servoing of dual-arm robot for cyclic motion," *Industrial Robot: An International Journal*, 2017.

[19] J. Qu, F. Zhang, Y. Fu, and S. Guo, "Multi-cameras visual servoing for dual-arm coordinated manipulation," *Robotica*, vol. 35, no. 11, p. 2218, 2017.

[20] 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.

[21] T. Shkurti, "Simulation and control enhancements for the da vinci surgical robot™," Master's thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, 5 2019.