Optimal Dexterity for a Snake-like Surgical Manipulator using Patient-specific Task-space Constraints in a Computational Design Algorithm

Andrew Razjigaev, Ajay K. Pandey, Jonathan Roberts Senior Member IEEE, and Liao Wu Member IEEE

Abstract—Tendon-driven snake-like arms have been used to create highly dexterous continuum robots so that they can bend around anatomical obstacles to access clinical targets. In this paper, we propose a design algorithm for developing patient-specific surgical continuum manipulators optimized for oriental dexterity constrained by task-space obstacles. The algorithm uses a sampling-based approach to finding the dexterity distribution in the workspace discretized by voxels. The oriental dexterity measured in the region of interest in the task-space formed a fitness function to be optimized through differential evolution. This was implemented in the design of a tendon-driven manipulator for knee arthroscopy. The results showed a feasible design that achieves significantly better dexterity than a rigid tool. This highlights the potential of the proposed method to be used in the process of designing dexterous surgical manipulators in the field.

I. INTRODUCTION

Tendons have been used to drive a variety of snake-like continuum and pseudo-continuum (hyper-redundant) robotic arms in the field. Minimally invasive surgery is one application where these robots are greatly beneficial because they are able to reach difficult-to-access surgical sites and complete tasks with great dexterity. Tendon-driven snake-like robots allow actuation to be done extrinsically from the arm allowing these robots to be miniaturized. These robots can decrease the footprint of teleoperated robotic surgery significantly as opposed to the current straight and rigid tools of state-of-the-art systems like the da Vinci [1].

In teleoperated surgery, dexterity is particularly important because the robot is not only required to reach the surgical sites but is also required to perform complicated manipulation. A recent survey on orthopaedic surgeries [2] revealed that surgeons find the procedure ergonomically challenging and have frustration with the current tools and technology. Dexterous snake-like robots controlled from a teleoperation console can be an ergonomic solution to the problems in knee arthroscopy. Therefore, this paper proposes a computational design algorithm for developing patient-specific snake-like tools for dexterous manipulation in arthroscopy, as illustrated in Fig. [1]

A. Related Work

Optimizing the design for snake-like robotic arms has often involved finding trade-offs in relationships with the mechanical design and its task space performance. The initial studies for optimal snake-like robots involved trade-offs with the workspace and certain mechanical properties like the stiffness in a flexible backbone joint [3]. When the patient-specific paradigm emerged, many literature in the field developed cost-functions to describe trade-offs for designing continuum robots for better工作spaces and path-planning in patient-specific anatomical environments. This was strongly evident in the design of concentric tube robots where patient 3D anatomies were collected by Magnetic Resonance Images (MRI) and optimizations were done to find the minimal tube lengths and their curvatures while penalizing collisions in a simulated navigation task to some desired targets [4] [5]. In other literature, a sampling-based motion planning approach to the problem was proposed to avoid complex inverse kinematics calculations [6].

Later in the field, new algorithms appeared that focused more towards on task space reachability in a volume rather than navigation to a point. Among those works were methods that discretised the workspace into voxels and generated an objective function to maximize the coverage of the concentric tube workspace in the region of interest [7]. Other algorithms turned this approach into an occupancy grid map where obstacles were dilated and sampled configurations of a centre-line representation of the robot were used to develop a cost function for motion planning [8]. One algorithm maximized reachability using a sampling-based motion planner to enable motions to a variety of target points in the lung [9]. All of these methods for optimizing continuum robots involve defining a cost function that would be optimized using generic optimization algorithms like generalized pattern search, the Nelder-Mead simplex algorithm and adaptive simulated annealing. These studies provided continuum robot algorithms for better navigation and reachability in patient-
specific anatomies but none of them focused on optimizing for dexterity.

Other patient-specific robot studies focused on the development of cost-effective disposable tools. Some of these works include developing patient-specific 3D printable concentric tube robots [10] from 3D Ultrasound scans [11]. One study designed disposable tendon-driven end-effectors for concentric tube robots to increase the dexterity at the tip [12]. In addition to this study, there was further work that conducted a dexterity analysis for the combination of disposable tendon-driven end-effectors with concentric tube robots [13]. They found that adding more degrees of freedom (DOF) to the distal end of the robot increased dexterity while adding DOF to the proximal end increased the workspace. They used the concept of orientability for analyzing dexterity which could be used as a guideline for future continuum robot design.

Other fields of research with conventional robotic manipulators had used the measure for manipulability for dexterity which is the singular value decomposition of the Jacobian matrix for a manipulator [14] [15]. One study with a conventional manipulator used a genetic algorithm to minimize a cost function to find the best trade-off between manipulability and the distance from obstacles in path-planning [16]. Although manipulability provides a quick and easy calculation for dexterity, the measure is only dependent on the joints and the forward kinematics which may not be suitable for dexterity in confined environments with many obstacles. Overall, the literature demonstrates that continuum robots are becoming patient-specific, disposable and optimized navigation and reachability performance in anatomical structures. However, there is a lack of work in designing them for the task performance of oriental dexterity in anatomical structures.

B. Contribution

Although there are computational design algorithms that exist for snake-like robots, no literature - as far as the authors are aware - has used oriental dexterity as a metric for their design given a confined task-space environment. Therefore, the contribution of this paper is to propose a novel computational algorithm for the optimal design of a robot arm for dexterous teleoperation. This algorithm was implemented for the surgical case study of designing a pseudo-continuum manipulator for knee arthroscopic teleoperation.

II. DESIGN PROBLEM FORMULATION

The problem is to create a process for designing dexterous robotic manipulators as a function of the task space. Specifically, it is to find the parameters for the design of a snake-like robot such that the dexterity is highest in the region of interest taken account of task space obstacles. The region of interest is the objects in the task space that are meant to be manipulated by the robot arm represented as voxels.

Given a set of parameters within bounds, all that can be provided from the robot arm model are:

- Samples of the configuration space bounded by joint limits;
- The forward kinematics;
- A measure of the robot arm skeletal shape from base frame to end-effector;
- And a path-plan navigation algorithm.

III. PROPOSED COMPUTATIONAL DESIGN PROCEDURE

The proposed procedure follows the patient-specific paradigm as visible in Fig. [2]. The processes of that procedure goes from a patient scan, to defining the task space as voxels, to optimizing the robot model for dexterity and finally to manufacturing a parametric design with its optimal parameters with 3D printing. The outcome is a dexterous patient-specific disposable tool for surgery.
A. Task Space Data Voxelisation

The first process is to turn 3D patient data into task space obstacles and regions of interest. This begins with collecting 3D data points of the task space or robot environment. For example in surgery, this could be an MRI, CT or ultrasound scan of the patient’s anatomy before the operation. It is assumed that a scan like that is a fairly good representative of the task space in the actual operation. The next phase is to discretize the 3D data points into voxels and classify regions as obstacles or as regions of interest. This can be done through a neural network that can do 3D semantic segmentation of the scanned anatomy or with inspection by a human in a computer program. The result of this discretization is evident in Fig. 2 where the knee scan scan of the patient’s anatomy before the operation. It is assumed that a scan like that is a fairly good representative of the task space. We discretized with respect to the port frame for robot insertion behind the region of interest as well as creating the voxel information needs to be summarized in a data file where all the voxel coordinates, bounds, class labels and future dexterity measures shall be kept.

B. Fitness Function: Sampling Configurations

The fitness function defines the performance metric of the robot that shall be optimised in an optimization algorithm. The fitness function in this study is defined in algorithm 1. The idea is to evaluate different parameterised designs consisting of vectors \( q \) in the design space \( D \) and measure its fitness \( F \) in the task space voxels \( V \).

\[
F = f(p,V); \quad p \in D \subset \mathbb{R}^m
\]  

To achieve optimal dexterity, we propose to use a Monte Carlo method to calculate the workspace to statistically analyze the dexterity distribution in the region of interest as the measure of the performance of each design. This method involves randomly sampling the joint space to find configurations that are representative of the configuration space \( Q \). For a sample size of \( N \), we get a matrix of \( N \times n \) consisting of vectors \( q_i \) to \( q_N \) which are kinematic inputs for \( N \) random configurations from the configuration space where \( n \) is the number of DOFs for the robot model. For each \( q_i \) from \( q_1 \) to \( q_N \), we calculate the forward kinematics for the design which outputs the homogeneous transform from robot base frame to end-effector frame.

\[
T_{\text{end}}^{\text{base}} = K(q_i,p); \quad q_i \in Q \subset \mathbb{R}^n
\]

In addition to this, we need to check if the configuration \( q_i \) and the motion path-plan \( P \) to it is collision-free. We can do this by sampling points along the robot configuration from the base to the end-effector to check that the robot 3D shape \( X \) is in a collision-free pose. These points represent the robot arm as a centre-line skeleton and can be integrated with the forward kinematics calculation. The sample size for shape should ensure that the sampled points are not further than the size of a voxel. For the path-plan, that would be a series of points for the robot shapes from configuration \( q_0 \) to \( q_i \) under an obstacle avoidance path-planning algorithm. For some tubular snake-like robots (like the one used in the implementation section), the path-plan varies little to the future dexterity measures shall be kept.

Algorithm 1 Fitness Function Dexterity in Region of Interest

```plaintext
1: function FITNESSFUNCTION(PARAMETERS,VOXELS)
2:    JointLimits, n ← RobotModel(parameters)
3:    for i = 1 : n do
4:        q ← random(qi_min, qi_max, SampleSize)
5:    end for
6:    Base Frame ← Voxels
7:    for i = 1 : SampleSize do
8:        q ← Q(i,n)
9:        T_{end}, X = ForwardKinematics(q,p,Baseframe)
10:       if Voxel(T_{end}) ∈ V_{roi} & Voxel(X) ∉ V_{obs} then
11:           if ReachablePathPlan(q0, qi) == True then
12:               P Z ← R_{end} ← T_{end}.
13:               θ, h = NearestPatch(P Z)
14:               UpdateServiceSphere(Voxel(T_{end}), (θ, h))
15:           end if
16:       end if
17:    end for
18:    N_{V_{roi}}, N_0, N_h ← Voxels
19:    for i = 1 : N_{V_{roi}} do
20:       surfacepatches ← ServiceSphere(Voxel(i))
21:       N_{o_{total}} ← N_{o}(i) = sum(surfacepatches)
22:    end for
23:    Dexterity = N_{o_{total}}/(N_{V_{roi}}N_0N_h)
24:    Return = -Dexterity
25: end function
```

1Patient specific data acquisition was approved by the Australian National Health and Medical Research Council (NHMRC) - Registered Committee Number EC00171 under Approval Number 1400000856. We thank Andres Marmol for providing the knee scan dataset used in this paper.
we can ignore that configuration and move on to the next one.

C. Fitness Function: Dexterity Index

The measure for orientable dexterity in this study is based on the index developed in [13]. To measure orientations for each specific spatial position (that is a voxel), each voxel contains a unit sphere called the Service Sphere. All of the possible areas on the sphere that can be oriented by the end-effector of the robot are referred to as Service Regions. This gives a dexterity measure for a voxel:

\[
Dex(v) = \frac{A_R(v)}{A_S} = \frac{N_O(v)}{N_\theta N_h} \in [0, 1]
\]  

(5)

Where \( A_R(v) \) is the area of the Service Region in voxel \( v \) and \( A_S \) is the total surface area of the unit sphere. To measure the area of Service Regions on the unit sphere, we use a discretization method. The surface of the sphere is discretized into \( N_\theta \) by \( N_h \) equally sized patches along longitude meridians and latitude lines with the angle between adjacent meridians being \( \delta \theta \) and the height interval of latitude lines being \( \delta h \). Letting \( N_O(v) \) represent the number of patches for the voxel that had been covered by the tooltip, we can calculate the dexterity \( Dex(v) \) as the ratio of the surface patches reached by the tip \( N_O(v)/N_\theta N_h \) over the total surface patches of the Service Sphere \( N_\theta N_h \). This ratio is bounded between 0 and 1 where 0 means that the tip could not reach any patch on the Service Sphere for that voxel and 1 means that the tip could reach all patches as the robot can touch the voxel from any orientation. So when the condition in Eq. (3) is true, we take the homogeneous transform \( r_{base}^{end} \), extract the rotation matrix \( R_{end} \), get the surface position from the z-axis unit vector \( P_z \) to find the corresponding latitude and longitude coordinate \((\theta, h)\) in the unit sphere. We then update the Service Region for voxel \( v \) and store the dexterity measure until all \( N \) configurations have gone through this procedure. Once that is complete, all that is left is to find the total dexterity for the voxels in the region of interest. This would be a sum of all dexterity measures for each voxel \( v \) in \( V_{roi} \):

\[
Dex(V_{roi}) = \int_{v\in V_{roi}} \frac{Dex(v)dv}{V_{roi}} \in [0, 1]
\]  

(6)

As a discrete measure this is a sum of all Service Region patches \( N_O \) in \( V_{roi} \) over the total area of the Service Sphere \( N_\theta N_h \) divided by the total number of voxels in the region of interest \( N_{V_{roi}} \):

\[
Dex(V_{roi}) = \frac{\sum_{i=1}^{N_{V_{roi}}} N_i}{N_{V_{roi}} N_\theta N_h} \in [0, 1]
\]  

(7)

This total dexterity measure is the main objective to maximize. Since most standard algorithms minimize functions, the measure is treated as a negative for the output of the fitness function:

\[
F = -Dex(V_{roi}) \in [-1, 0]
\]  

(8)

It is possible to add multiple objectives such as a penalty for the complexity of design parameters with a weighted sum of importance for the output of \( F \) in Eq. (1). For this study, only the dexterity objective shall be used for the implementation.

IV. IMPLEMENTATION OF THE COMPUTATIONAL DESIGN ALGORITHM FOR DESIGNING THE MANIPULATOR

This computational design algorithm was implemented for the application of designing a surgical manipulator. A snake-like manipulator is desired for the development of a teleoperated macro-micro robot for knee arthroscopy. This manipulator needs to pick and cut tissue in a certain region of cartilage. The Snake-like manipulator will be a micro robot attached to the end of the Raven II robot which is the macro robot. The Snake-like robot shall be manufactured cheaply through 3D printing so that it can be patient specific and disposable after the operation.

A. Snake-like Robot Model for the Experiment: A Variable Neutral-Line Mechanism

The variable neutral-line mechanism described in [17] proposed design improvements over traditional snake-like manipulators that use a flexible backbone in tendon driven robots. Their design proposed a rolling joint set-up such that the robot mechanism had an adjustable stiffness. Their design was also hollow which is significantly important for attaching tools to the end of the robot such as a gripper or a camera. Therefore, for its many advantages, this design was used as the basis for developing the snake-like robot for this study.

The forward kinematics for this robot is described in [18] as a series of the pan and tilt rolling joint transforms. For the parametric design for this mechanism, the fewest amount of variables to define the model were \( n, d, w \) and \( \alpha \) where \( n \) defines the number of disk joints, \( d \) is the joint height, \( w \) is the width of the robot while \( \alpha \) is half the rolling joint angle of curvature. These four parameters define the set of variables for a single segment or module. For multiple segments, the variable \( w \) should be constant as the model would not be practical having a variable thickness for each segment. In this study, we will reduce our design space by defining \( w = 4 \text{mm} \) based on a suitable size for the robot to house an instrument and fit the trocar. This gives us \( 3 \times s \) parameters to solve. There are also some other parameters defined for the transition between multiple segments. In [18], those parameters are \( \zeta \) and \( \sigma \) which define the transition rotation and translation. To simplify this, we assume that the transition is integrated with previous joint such that \( \sigma = \sigma_{proximal} \) and that \( \zeta = -\pi/s \) to ensure the tendons for each segment are evenly positioned. Each segment of the robot has two DOF which are pan and tilt joints. This gives a configuration space of five DOF for the single segment and seven DOF for a double segment.

B. Differential Evolution Optimisation Set-up

Since the fitness function is numerical, we need a good derivative free generic optimization algorithm to optimize for it. For this, differential evolution was chosen as the algorithm to optimize the function for its advantages over conventional genetic evolution algorithms [19]. It is a parallel direct search
method which has a scheme that helps the method converge faster to the optimal than adaptive simulated annealing and annealed Nelder-Mead approaches. For this study, the first strategy of differential evolution was used. This scheme generates trial parameter vectors by adding a weighted difference vector between two population members to a third member as shown in Eq. (9). For \( x_{r_1, G}, i = 0, 1, 2, ..., NP - 1 \), a trial vector \( v \) is generated according to:

\[ v = x_{r_1, G} + F \cdot (x_{r_2, G} - x_{r_3, G}). \]  

For this study, we used a differential weight of \( F = 0.8 \) and crossover value \( CR = 0.7 \) and the populations \( NP \) being \( 10 \times p \) which was 30 for the single segment and 60 for the double segment. The evolution stop time was set for 24 hours with 10 test repeats.

V. EXPERIMENTAL RESULTS

The optimization was done for a single- and a double-segment design, respectively. The corresponding mean and maximum voxel dexterity values were compared with the fitness function results for a rigid instrument. The average evaluation time of the fitness functions for 1 segment was about 1 minute and 23 seconds and for 2 segments it was about 2 minutes and 21 seconds. The sample size was chosen as 1,000,000, the voxels were 2\( \text{mm} \) cubes and the surface patches were \( N_\theta = 18, N_h = 9 \). The robot also had a 5\( \text{mm} \) long tool attached to the end-effector. The optimization was done as a job for a High Performance Computer with a timer of 24 hours. The convergence of the optimization algorithms are shown in Fig. 3. The designs and their dexterity measures are summarized in Tab. 1 and Fig. 4.

| Model             | Parameters     | Mean Dexterity | Maximum Dexterity |
|-------------------|----------------|----------------|-------------------|
| Rigid Tool        | NA             | 0.0055         | 0.0247            |
| Single Segment    | \( \alpha, d, n \) | 1.24, 1.62, 3  | 0.0835            | 0.1358            |
| Double Segment    | \( \alpha_1, d_1, n_1, \alpha_2, d_2, n_2 \) | 1.34, 6, 1, 1.18, 0.41, 3 | 0.1351            | 0.2272            |

VI. DISCUSSION

In general, the pattern from the results in Fig. 4 shows that the more DOF the robot has the better the dexterity. The service sphere and distribution for the rigid instrument in Fig. 4 shows that its orientable dexterity is much less than the optimal snake-like robots. It is intuitive to expect that a rigid tool would have a very limited range of orientability about a small voxel from a single port. By its distribution, it is also evident that it was designed to manipulate tissue from the furthest end of the region of interest from the port. The single segment, on the other hand, shows a much better service sphere and a more evenly distributed dexterity in the region of interest. The evolution algorithm chose to have \( n = 3 \) and a large \( \alpha \) angle. This choice leads to having a snake-robot with a small wrist-like joint for fine manipulation as the Raven II provides the workspace coverage. Lastly, the double segment has the greatest service sphere and a higher valued distribution in the region of interest. The evolution interestingly chose to have a single proximal joint and a few smaller distal joints. Note that this means that the design lacks a DOF as the proximal section is a single pan joint giving a total of 6 DOF in the design. This verifies that 6 DOF is sufficient for this dexterous task. Furthermore, this design resembles the structure of a human arm because the Raven II acts as a shoulder joint, the proximal joint behaves like an elbow for a wider workspace and the distal joints behave like a wrist for fine manipulation. Clearly, the algorithm chose these features as they desired the best task-space coverage and orientability for each voxel in the region of interest. To further verify that the two-segment design is significantly better than the one-segment design in terms of performance, a Mann-Whitney U test was used to compare

---

2 Computational resources and services used in this work were provided by the HPC and Research Support Group, Queensland University of Technology, Brisbane, Australia
the designs and their dexterity results for the 10 tests in Fig. 3. The Z-score from that was 4.3487 indicating a strong 99.99% level of confidence that the two segment design performs better than a single segment.

Implementation of the computational design algorithm has involved tuning many settings that may not be intuitive to justify. It is unclear how accurate the simulation of the task space should be to produce a sufficient design. Changing settings like increasing the sample size, decreasing the patch size and the voxel size all make the computation exponentially more expensive. Tuning these values can make this algorithm difficult to adapt to the practical use of developing patient-specific robots. Further work can be done to improve the algorithm computationally with parallel loops. The current challenge in doing this is with updating the same voxel service spheres simultaneously in Algorithm [1]. Other improvements can be to integrate a path plan checker using sampled configurations to define a reachable map. Another improvement can be to combine other objectives and global optimisation for the port location and the other tools in the operation. Finally, future work in using the design in a teleoperation task is needed to observe how effective the dexterity performance is for enhancing the surgeon’s abilities in the application.
VII. CONCLUSION

In this study, tendon-driven pseudo-continuum robots were designed under the patient-specific paradigm for knee arthroscopy. A novel algorithm was proposed to optimize the design for optimal oriental dexterity within a confined task space. It was implemented for a variable neutral-line mechanism for a tendon-driven robot. The results show that the optimal snake-like robot designs had significantly greater dexterity than a conventional rigid tool used for the application. Overall, the results show that this algorithm is feasible for the design of surgical manipulators under anatomical constraints but further work is required to improve the algorithm and to find ways of adjusting the settings for different surgical applications.

REFERENCES

[1] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, “Continuum Robots for Medical Applications: A Survey,” IEEE Transactions on Robotics, vol. 31, no. 6, pp. 1261–1280, Dec 2015.

[2] A. Jaipurakash, W. B O’Callaghan, S. Whitehouse, A. Pandey, L. Wu, J. Roberts, and R. Crawford, “Orthopaedic surgeon attitudes towards current limitations and the potential for robotic and technological innovation in arthroscopic surgery,” Journal of Orthopaedic Surgery, vol. 25, p. 230949901668499, 2017.

[3] Z. Zhang, G. Yang, S. H. Yeo, W. B. Lim, and S. K. Mustafa, “Design optimization of a cable-driven two-dof joint module with a flexible backbone,” in 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, July 2010, pp. 385–390.

[4] C. Bedell, J. Lock, A. Gosline, and P. E. Dupont, “Design optimization of concentric tube robots based on task and anatomical constraints,” Proceedings - IEEE International Conference on Robotics and Automation, pp. 398–403, 2011.

[5] T. Anor, J. R. Madsen, and P. Dupont, “Algorithms for design of concentric tube robots using mechanics-based models,” in 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2010, pp. 385–390.

[6] L. Wu, R. Crawford, and J. Roberts, “Orthopaedic surgeon attitudes towards current limitations and the potential for robotic and technological innovation in arthroscopic surgery,” IEEE Transactions on Robotics, vol. 30, no. 2, pp. 514–521, April 2014.

[7] C. Baykal, L. G. Torres, and R. Alterovitz, “Optimizing design parameters for sets of concentric tube robots using sampling-based motion planning,” in IEEE International Conference on Intelligent Robots and Systems, 2015.

[8] J. Kim and K. Khosla, “Dexterity measures for design and control of manipulators,” in Proceedings IROS ’91:IEEE/RSJ International Workshop on Intelligent Robots and Systems ’91, Nov 1991, pp. 758–763 vol.2.

[9] K. Abdel-Malek, W. Yu, and J. Yang, “Placement of Robot Manipulators to Maximize Dexterity,” International Journal of Robotics and Automation, vol. 19, no. 1, 2004.

[10] E. Amanov, T.-D. Nguyen, and J. Burgner-Kahrs, “Additive manufacturing of patient-specific tubular continuum manipulators,” Medical Imaging 2015: Image-Guided Procedures, Robotic Interventions, and Modeling, no. March 2015, 2015.

[11] T. K. Morimoto, J. J. Cerrolaza, M. H. Hsieh, K. Cleary, A. M. Okamura, and M. G. Linguraru, “Design of patient-specific concentric tube robots using path planning from 3-D ultrasound,” Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, pp. 165–168, 2017.

[12] A. B. Prasai, A. Jaipurakash, A. K. Pandey, R. Crawford, J. Roberts, and L. Wu, “Design and fabrication of a disposable micro end effector for concentric tube robots,” in 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), Nov 2016, pp. 1–6.

[13] L. Wu, R. Crawford, and J. Roberts, “Dexterity analysis of three 6-dof continuum robots combining concentric tube mechanisms and cable-driven mechanisms,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 514–521, April 2017.

[14] J. Price and R. Storn, “Differential Evolution - A simple evolution strategy for fast optimization,” Dr. Dobb’s journal, vol. 22, no. 4, pp. 18–24, 1997.