Robotics and Autonomous Systems 62 (2014) 1478–1485

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Real-time method for tip following navigation of continuum snake arm robots

David Palmer, Salvador Cobos-Guzman, Dragos Axinte *

Rolls-Royce UTC in Manufacturing Technology, University of Nottingham, Department of Mechanical, Materials and Manufacturing Engineering, Nottingham, NG7 2RD, United Kingdom

HIGHLIGHTS

- Optimization approach for Tip Following snake arm robots.
- Designed for a generic continuum Snake Arm construction.
- Proposes an efficient method of creating the objective function.
- Minimizes the deviation from the navigated path, rather than environmental objects.

ARTICLE INFO

Article history:
Received 12 August 2013
Received in revised form
16 May 2014
Accepted 26 May 2014
Available online 4 June 2014

Keywords:
Hyper-redundant manipulator
Optimization
Snake arm
Tip following

ABSTRACT

This paper presents a novel technique for the navigation of a snake arm robot, for real-time inspections in complex and constrained environments. These kinds of manipulators rely on redundancy, making the inverse kinematics very difficult. Therefore, a tip following method is proposed using the sequential quadratic programming optimization approach to navigate the robot. This optimization is used to minimize a set of changes to the arrangement of the snake arm that lets the algorithm follow the desired trajectory with minimal error. The information of the Snake Arm pose is used to limit deviations from the path taken. Therefore, the main objective is to find an efficient objective function that allows uninterrupted movements in real-time. The method proposed is validated through an extensive set of simulations of common arrangements and poses for the snake arm robot. For a 24 DoF robot, the average computation time is 0.4 s, achieving a speed of 4.5 mm/s, with deviation of no more than 25 mm from the ideal path.

1. Introduction

Snake Arm robots are a specific subset of hyper-redundant manipulators which are primarily defined by their ability to navigate through a complex environment, such as a nuclear power plant [1] or the aftermath of an earthquake [2]. These robots can be used for inspection tasks or minimally invasive repairs (if their stiffness allows).

The term hyper-redundant, coined by Chirikjian and Burdick [3], refers to a redundant manipulator with a large number of degrees of freedom (DoF) and a similar morphology to a tentacle or snake. Normally, the common designs use repeating structures of two revolute joints. These joints can either be continuum, formed of a flexible rod or a spring [4] or discretized as a series of universal joints [5]. There are various methods that can be used to actuate this kind of manipulator including cables [4], pneumatics [6] or shape memory alloys [7].

Snake arms often consist of four main parts [8], as depicted in Fig. 1. The arm is split up into several sections; each section has 2 DoF, and an end-effector that depends on the type of scenario. The end-effectors can consist of grippers, high-speed spindles, a vision system, or a combination. Most designs keep the main actuation mechanism extrinsically in order to reduce the weight within the snake arm. This actuation is often mounted in a pack at the base of the robot. Unlike most hyper-redundant manipulators, some snake arms and industrial robots use a feed-in mechanism, to allow advancing of the base position of the arm, which makes longitudinal/rotational navigation easier. In the following equations it is assumed that the feed-in system used can accurately provide a linear advancement along the initial trajectory and it is not intended to be the subject of the current communication.

* Corresponding author. Tel.: +44 0 1159514117; fax: +44 0 1159513800.
E-mail address: Dragos.Axinte@nottingham.ac.uk (D. Axinte).

http://dx.doi.org/10.1016/j.robot.2014.05.013
0921-8890/

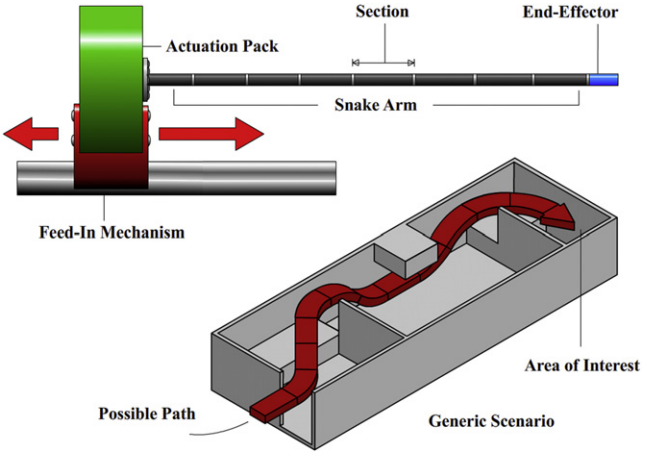

Fig. 1. Generic snake arm design and a generic scenario.

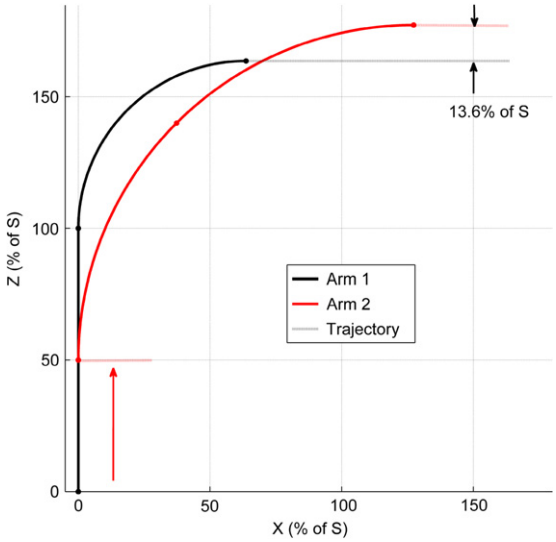

Fig. 2. Non-linear relationship between the change in the bend and advancement of the base.

Fig. 1 also shows a simplified concept of a typical problem that can be solved by the snake arm. The user wishes to inspect or repair a feature in the area of interest; access is not possible with a conventional tool and this scenario is not immediately solvable with the sweeping motions commonly used by hyper-redundant manipulators [9]. However, the snake arm can move along the path suggested and weave through the obstacles and then perform the required task. This environment is used to demonstrate the algorithm.

The approach often taken, for this longitudinal navigation, is to apply already existing algorithms such as obstacle avoidance or path planning. Chirikjian et al. [10] demonstrates a planar obstacle avoidance algorithm with a "tunneling" technique (to constrain the geometry) to navigate through a known environment. An extension of this performed by Choset and Henning [11] uses sensor data to create Generalized Voronoi Graphs to achieve a follow-the-leader approach. Another optimization approach is presented by [12] which uses pre-defined obstacles to navigate active cannulas. Moreover, an obstacle avoidance algorithm for rigid link robots [13] was updated by [14] to show how it can be adapted to continuum manipulators with the use of discrete sensors along the length of the arm.

Most obstacle avoidance algorithms or path planners require a well-defined environment and they would struggle in tasks with free-moving elements that, although are known by their CAD information, their position is unknown within the navigation space, such as the rotatives within a jet engine that can move freely on a shaft; or for disaster sites where no environmental data exists. The obstacles in these types of environment can be mapped by sensors, as suggested in [14]; however sensors add extra weight which many designs could not carry due to limited payload capacity.

An alternate technique for navigating the Snake Arm is Tip Following; this process works on-the-fly by treating the tip of the arm as a free-moving point. As this point is moved in the desired direction, an algorithm is used to determine a new arrangement of the snake arm that advances the tip to the new position and minimizes the deviation from the path already travelled. Additionally, it is possible to map the movements of this point to a two-axis gamepad to create an intuitive interface for the end-user. The user can navigate the tip around obstacles with only a camera feed embedded in the end-effector, if the algorithm can guarantee a solution within a deviation tolerance.

For uninterrupted movement, the algorithm needs to determine a new arrangement of the robot in the time that is taken for the previous change to be performed. A simple concept for calculating a solution in real-time is commonly used in computer graphics, especially in the videogame Snake. Each section of the snake avatar, as it moves, takes the position of the previous section. In a similar manner, by recording the configuration of the tip section every step interval, it is possible to pass this data to subsequent sections when they reach the same position.

Experimentation with this concept [15] showed that, for snake arms with long sections and high bending angles, there is an oscillating error in the tip position. To explain the origins of this error, Fig. 2 shows a pair of 2-section snake arms. Arm 1 is formed as a straight trajectory, followed by a $\pi/2$ bend angle. This arm is to be advanced a full section length, S, such that the first section is a $\pi/2$ bend angle and a straight trajectory depicted by the dotted line. To preserve the orientation of the tip, it is obvious that the two sections must pass through $\pi/4$ bending angle at the same time. Arm 2 shows this intermediate stage if it is assumed to occur midway through the advancement, i.e. assuming a linear relationship. However the tip position overshoots the trajectory by 13.66% of S. As such, it is clear that the relationship is non-linear and therefore not capable of maintaining the path and this might lead to collisions within the environment.

Recently, a new tip following method suitable for snaking has been reported [16]; this takes sections in pairs along the length and finds an arrangement that starts tangential to the path and ends tangential to the path. By following this process through the whole arm it is possible to find a completely new pose for small changes. However this method limits the freedom of the snake, as every other section end must be on the path, which may not be possible in complex poses with high bending angles.

This paper presents a novel methodology for calculating the performance of a new arrangement, to be used as an objective function for the sequential quadratic programming (SQP) method. This optimization process is then implemented within a tip following algorithm suitable for navigating a snake arm in real-time. This algorithm is free of any requirements of environmental data and represented such that it can be used by a wide range of continuum designs with various different specifications. Performance of simulations of the complete system is determined based on the path deviation, tip accuracy, and computation time.

2. Geometric representation of a snake arm

Any continuum design of sufficient flexibility, or a discrete design with enough passive joints to be considered serpentine, can be modelled as a series of arcs for each actuated section. The formulation of this model uses the kinematics offered by [17], since

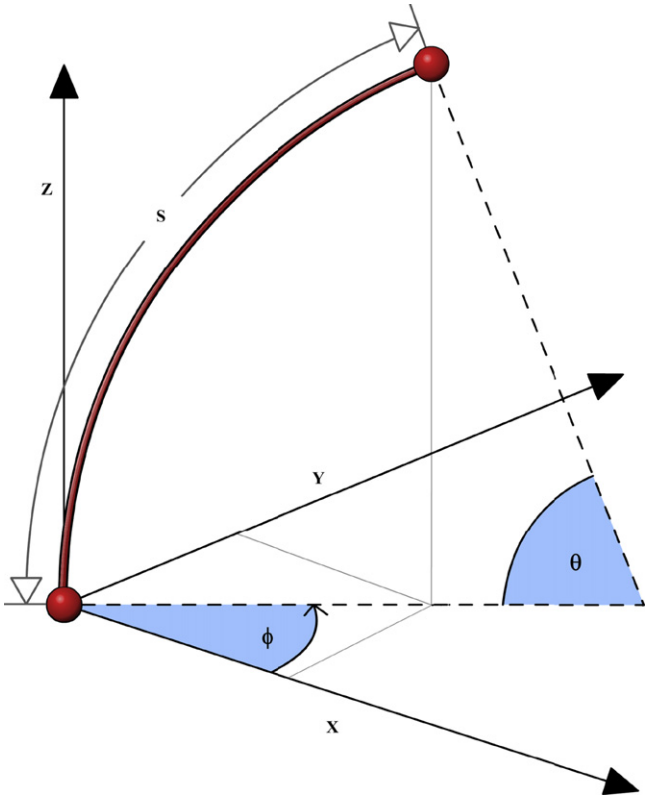

Fig. 3. Representation of one actuated section with configuration parameters annotated.

these equations present a purely geometric case and ignore the effect of gravity on the arm. It is assumed that any external influence on the accurate positioning of the arm is taken into account by the low-level control.

Taking the equation in [18], the end point of one section can be calculated from the configuration parameters, shown in Fig. 3, as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{S}{\theta} \cos(\phi)\,(1-\cos(\theta)) \\ \frac{S}{\theta} \sin(\phi)\,(1-\cos(\theta)) \\ \frac{S}{\theta} \sin\theta \end{bmatrix} \tag{1}$$

where S is the section length, θ is the bending angle, and ϕ is the direction angle (counter-clockwise from the x-axis). Note that ϕ denotes the heading of the bend and not a rotation about the z-axis. This is because the rotation matrix, R, between the origin and the plane at the end point of this section is a ZYZ rotation matrix of ϕ, θ and $-\phi$ respectively [19]. The arc that represents this section's path is assumed to be a smooth curve between the origin and the point calculated by (1) with a radius of S/θ.

The next section can be calculated by the same method, if the end of the previous section is taken as the origin of a new coordinate frame. By using the rotation matrix and a translation, it is possible to represent this second section in the base coordinate frame starting from the tip of the previous section. This technique can be repeated, assembling the representation of the snake arm one section at a time. The end of the nth section can be calculated as

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = R_n \begin{bmatrix} \frac{S_n}{\theta_n} \cos(\phi_n)\,(1-\cos(\theta_n)) \\ \frac{S_n}{\theta_n} \sin(\phi_n)\,(1-\cos(\theta_n)) \\ \frac{S_n}{\theta_n} \sin(\theta_n) \end{bmatrix} + \begin{bmatrix} x_{n-1} \\ y_{n-1} \\ z_{n-1} \end{bmatrix} \tag{2}$$

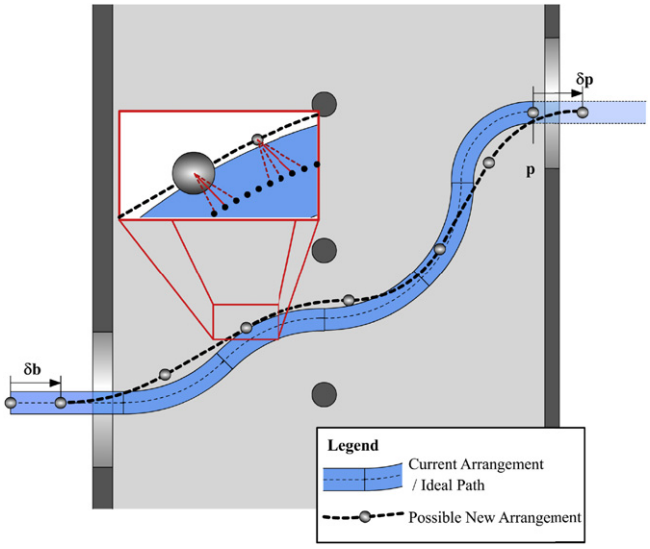

Fig. 4. Comparison of a new arrangement to the current path.

where

$$R_n = R_{n-1}\, Rot_z(\phi_{n-1})\, Rot_y(\phi_{n-1})\, Rot_z(-\phi_{n-1})$$

Rot_z and Rot_y are the standard 3×3 rotation matrices about the z and y axis respectively. Eq. (2) is written such that it can be implemented as a recursive function, taking R_0 as an identity matrix and $[x_0, y_0, z_0]^{\mathbf{T}}$ as the origin. This can be used to create a simple program to find the end positions, rotations and path of an n-section continuum snake arm; this is similar to the work Chirikjian [20] accomplished on the dynamics of such structures.

3. Formulation of the objective function

Any optimization process is only as good as the objective function. Nevertheless, there are some parameters that are difficult to calculate very quickly such as the performance value of a hyper-redundant system. With respect to the tip following, this performance value has to indicate on the one hand, the accuracy of the tip position and on the other hand the deviation from the path.

3.1. Creating the basic function

Fig. 4 shows the Snake Arm mid-way through navigating a typical scenario. Since the configuration parameters of this arrangement are known, the tip position, p, can be found through the forward kinematics described in Section 2. The tip is advanced a small step, δp. Smaller values for δp result in smoother motion whereas larger values will reduce the number of steps. With each step, a new arrangement is needed that minimizes the deviation from the ideal path. Consequently, in order to keep the changes to the configuration parameters small, it is essential that the base of the snake is advanced as well; this is represented by a linear movement of δb at the base.

The optimization algorithm inputs a set of small changes to the current configuration to the objective function to create a new arrangement, characterized by the dotted line. The path produced can be represented as an array of three-dimensional points. The maximum distance between these points and the current path can be used to quantify the deviation of the new solution. This can be written as an objective function

$$f([\delta\theta, \delta\phi, \delta b]) = Max\,(Path\,(\theta+\delta\theta, \phi+\delta\phi, b+\delta b) - [Path\,(\theta, \phi, o), p+\delta p]) \tag{3}$$

where *Path* is the function, based on (2), to create an array of points denoting the arrangement; $p + \delta p$ is appended to the current path to indicate the advance in tip position.

However, this does not guarantee that the new arrangement will reach δp; hence the Euclidean distance between the new tip position and the desired is weighted by a factor of ten to strengthen the effect this error has on the performance value. With this adjustment the problem is now written as

$$\begin{aligned} f([\delta\theta, \delta\phi, \delta b]) = {} & 10\,(Tip\,(\theta + \delta\theta, \phi + \delta\phi, b + \delta b) \\ & - (Tip\,(\theta, \phi, b) + \delta p)) \\ & + Max\,(Path\,(\theta + \delta\theta, \phi + \delta\phi, b + \delta b) \\ & - [Path\,(\theta, \phi, b), \delta p])\,. \end{aligned} \tag{4}$$

By finding values that minimize (4), it is possible to calculate a new arrangement of the snake arm which reaches $p + \delta p$ and follows a similar path to the old arrangement.

3.2. Improving the efficiency of the objective function

As previously mentioned, it is vital that this optimization process is efficient such that computation times can be kept to a minimum: this will enable the snake arm to navigate in real-time. As such, it should be ensured that the optimization does not perform unnecessary calculations.

3.2.1. Reducing the number of calculations

The deviation between the two paths need not be a direct point–point comparison; in fact this can lead to more errors. Alternately the solution to be tested only needs a few points per section, which can be compared to the higher resolution current path. An assumption of where these points should line up can be made to minimize the number of calculations required. The pseudo-code in Fig. 5 demonstrates how this is achieved for a solution path with 5 points per section and a target path of 100 points per section for a 10-section snake arm.

Each point takes an approximation of where to start based on the known resolutions of each path. These indices are taken from the tip rather than the base, since in real scenarios the target path can be longer than the snake arrangement. The code uses a gradient search to adjust the target path index of each point to find the best representation of the distance. By using this approach, the average calculations per function evaluation can be reduced to 250; compared to 1500 needed for direct comparison of paths with 50 points per section.

3.2.2. Spreading the computational load

Since the iterations of the FOR loop in the code are independent from each other it is possible to use parallel computation to speed up the process. Most optimization techniques can also benefit from being calculated in parallel. On a similar thread, by setting common functions as re-entrant allows multiple instances to be run concurrently rather than waiting for the shared resource to be free.

3.2.3. Reducing the search space

Another way to increase the efficiency of the system is to constrain the input and range; because the optimization algorithm treats the objective function as a black box, it does not have an idea of what order of magnitude the inputs should be given. As such it may waste a number of function evaluations on non-sensible values. Furthermore, because a solution is likely to be similar to the current arrangement it is possible to reduce the search space within a tight area.

The constraints used are ± 0.064 radians, $\pm\pi$ radians, and ± 2 mm for $\delta\theta$, $\delta\phi$ and δb respectively. These values are based of

```
For  i = 1 : length(solutionPath)

  %find absolute length
  aL = length(targetPath)
  %find expected length
  eL = (100*10)+1
  %estimate target path index
  j = aL − eL − ((i−1)*(100/5))

  %initialize current value
  b = Euclidean²(solutionPath(:,i),targetPath(:,j))

  %perform gradient search
  isMin = false
  While(isMin == false)
    a = Euclidean²(solutionPath(:,i),targetPath(:,j−1))
    c = Euclidean²(solutionPath(:,i),targetPath(:,j+1))

    If(a < b)
      b = a
      j = j−1
    Elseif(c < b)
      b = c
      j = j+1
    Else
      isMin = true
    End
  End
  error(i) = b
End
max(error)
```

Fig. 5. Pseudo-code for efficient path comparison.

the extreme case of a chicane in the path with bending angles of $\pi/2$, in this scenario a section has to change from $\pi/2$ to $-\pi/2$ within the number of steps to advance a section length. An additional constraint is added to state that a section cannot bend further than $\pi/2$.

3.3. Implementation

The results in Section 5 are being performed using a non-linear constrained optimization by the means of the SQP algorithm from the LabView© software package. With all the techniques discussed above it is possible to find a solution in less than 0.4 s for most scenarios with a max path deviation of approximately 25 mm; this simulation time is found to be satisfactory where continuum are to be used for interventions where navigating time is only a fraction compared with the benefits of accessing in-situ. More details on how these values change depending on scenario can be found in the results.

4. Implementing the complete system

So far this paper presents only the movement of the tip in a straight line, and not how the curves in the path are created or how to integrate the base advancement with a feed-in mechanism. These have been overlooked intentionally in the previous section as neither of these behaviours affects the optimization process; but they are critical for the final system.

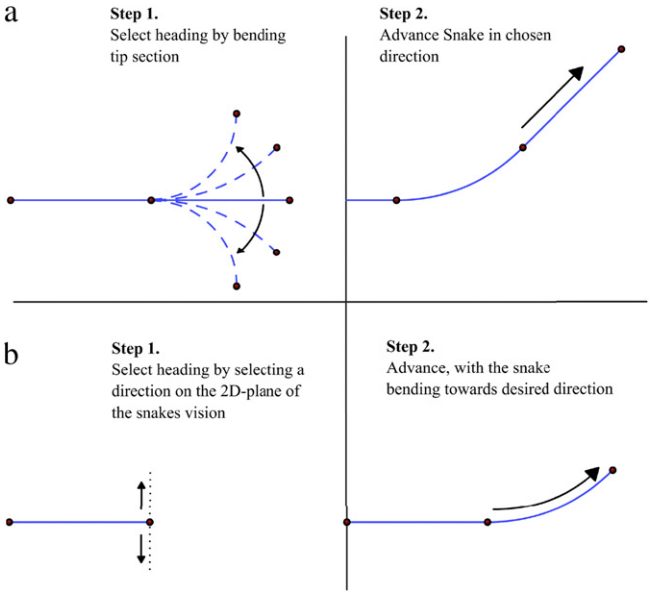

Fig. 6. Two different methods for creating bends (a) sweep and then move (b) bend as advanced.

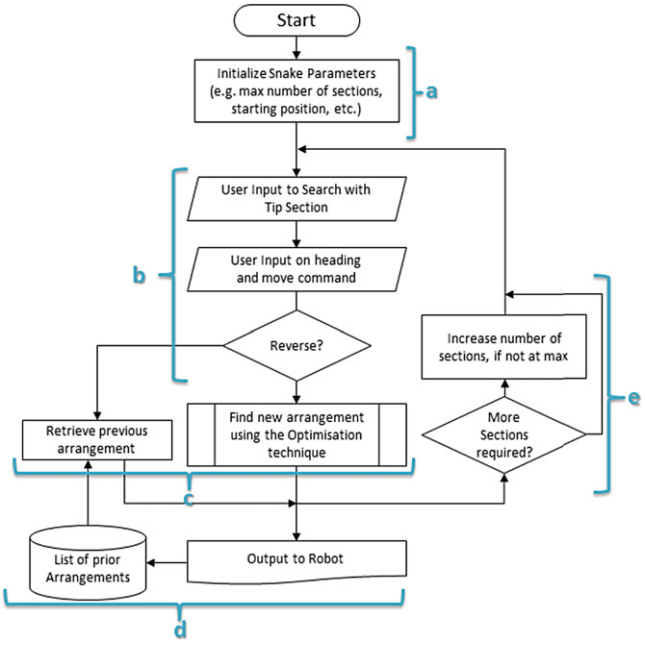

Fig. 7. Flowchart of complete Tip Following algorithm.

4.1. Navigating a curve

There are two different methods for controlling the direction travelled by the Snake Arm. The first, Fig. 6a is designed to be familiar to users with prior experience of videoscopes. The user controls the bending of the tip section to decide on which direction to head, with the aid of a camera mounted at the tip. In very restrictive environments this allows the snake to achieve sharp curves quicker, but creates a risk of collision as the tip is flexed. The alternative is to modify δp such that the tip will bend towards the desired heading as it is advanced; this is qualitatively similar to the steering methods of needles [21]. While this is a slower method to create the bend, it creates a smoother path. The choice between the two methods comes down to user preference and environment operating within; it is also feasible to provide a path of coordinates points to follow.

4.2. Combining with the feed-in system

When utilizing a feed-in mechanism, it is possible to ignore the parameters of some sections of the Snake Arm that have not entered the environment. In turn, this greatly benefits the optimization, as the number of sections exponentially increases the computational time. Therefore this means that the time per step can be greatly reduced for the first few sections to enter the environment.

To achieve this combination, the last section to enter the scenario is treated as the base until the system has advanced enough for a new section to be considered. At this time, this new section must be straight to avoid complications when introduced.

4.3. The complete tip following system

A flowchart of how the full-algorithm works is shown in, Fig. 7. In this algorithm, all the parameters are initialized, such as the number of sections, in the block labelled (a); afterwards the user decides which direction to head (by either technique) and gives the command to advance or reverse (b). This then gets the algorithm to determine δp required and input the variables into the optimization; alternately, if the user wishes to reverse, the previous pose is recovered from the record (c). Once a new arrangement is found, it is given to the actuators and recorded (d). The system then verifies whether another step is feasible, or next section should be passed by the feed-in system (e). The cycle then repeats from the bending decision.

5. Simulation trials and results

The simulations for the new tip following strategy were performed on a standard commercial desktop running an i7 960 (3.2 GHz) processor. Simulations were performed in LabView and analyzed in MATLAB©. All tests have a section length of 100 mm and are advanced with steps of 2 mm. For each trial, the computation time per step is measured and the deviation from the desired path and tip position are recorded. The results are summarized in Table 1, along with some extra trials that are not covered in the text.

5.1. Performance of optimization

The performance of the optimization is very subjective to the problem proposed. Hence it is important to get an idea of how different arrangements of the target path affect the computational time and path deviation. The profiles simulated are s-bends, right-angled bends and non-planar paths. S-bends consist of two bends which cancel each other out to return the tip to the initial orientation. Right-angle bends create a $\pi/2$ bend out of varying number of sections. Non-planar paths are to demonstrate that the system works in three dimensions and is not limited to the x-plane.

Each graph shows every pose used by the Snake Arm as the tip is advanced from the origin to the end point. By overlaying each of these arrangements it creates an envelope which provides a visualization of the deviations and where they occur.

Fig. 8 shows 3 different S-bend profiles with the same number of sections. The simulations vary the magnitude of the bending angle from $\pi/2$ to $\pi/6$. $\pi/2$ gave the most deviation at 19.5 mm, $\pi/4$ deviates by 5.58 mm, whereas $\pi/6$ maintained the path within 3.67 mm. The computational time was very similar for each graph, averaging 0.05 s per step.

The deviation in $\pi/2$ is significantly larger mainly because this scenario is the limit used to set the constraints, however larger deviation is to be expected at the higher bending angles since it is harder to maintain the path as it is apparent in all the graphs.

Table 1
Summary of simulation results.

	Profile	Total time	Max time	Avg time	Max dev	Avg dev	Max tip	Avg tip
S-Bend profiles	$[0, \pi/2, -\pi/2, 0]$	10.29	0.260	0.051	19.50	7.170	1.29	0.475
	$[0, \pi/4, -\pi/4, 0]$	11.09	0.324	0.055	5.577	1.935	0.66	0.162
	$[0, \pi/6, -\pi/6, 0]$	10.75	0.352	0.053	3.674	1.225	0.54	0.129
$\pi/2$ profiles	$[0, 0, 0, \pi/2, 0]$	23.52	0.438	0.093	16.59	2.298	1.10	0.151
	$[0, 0, \pi/4, \pi/4, 0]$	22.27	0.429	0.088	1.952	0.651	0.26	0.044
	$[0, \pi/6, \pi/6, \pi/6, 0]$	18.92	0.554	0.075	1.517	0.699	0.17	0.037
7-Section profiles	$\theta = [0, \pi/6, \pi/4, \pi/5, \pi/5, \pi/5, \pi/4]$ $\phi = [0, \pi, 0, 0, 0, \pi/4, 0]$	55.81	1.016	0.159	4.031	1.821	0.84	0.094
	$\theta = [0, \pi/2, \pi/4, \pi/4, \pi/3, \pi/3, \pi/3]$ $\phi = [0, 0, \pi/2, \pi/2, \pi, \pi, \pi]$	61.33	0.910	0.174	8.609	3.843	0.87	0.143
12-Section profiles	$\theta = [0, \pi/4, 0, 0, \pi/4, \pi/4, \pi/4, 0, \pi/4, \pi/4, \pi/4, \pi/4]$ $\phi = [0, \pi, 0, 0, 0, \pi/2, 3\pi/2, 0, 3\pi/2, \pi/2, 0, 0]$	235.3	2.301	0.391	20.15	6.340	0.90	0.177
	$\theta = [0, \pi/9, \pi/9, \pi/9, \pi/9, \pi/9, \pi/5, \pi/5, \pi/5, 0, \pi/2, 0]$ $\phi = [0, 0, \pi/18, \pi/9, \pi/6, 2\pi/9, \pi, \pi, \pi, 0, \pi/2, 0]$	250.7	2.234	0.417	24.73	2.742	1.32	0.108
	$\theta = [0, \pi/4, 0, \pi/8, \pi/8, \pi/6, \pi/6, \pi/6, \pi/6, \pi/6, 0, \pi/6, 0]$ $\phi = [0, 0, 0, \pi, \pi, 3\pi/2, 3\pi/2, 3\pi/2, 3\pi/2, 0, \pi/2, 0]$	262.6	2.323	0.436	14.71	6.314	0.55	0.144

Max Dev is the maximum deviation in the path and Max Tip is the max deviation in the tip position. Times measured in seconds, deviation and tip deviation are measured in millimetres.

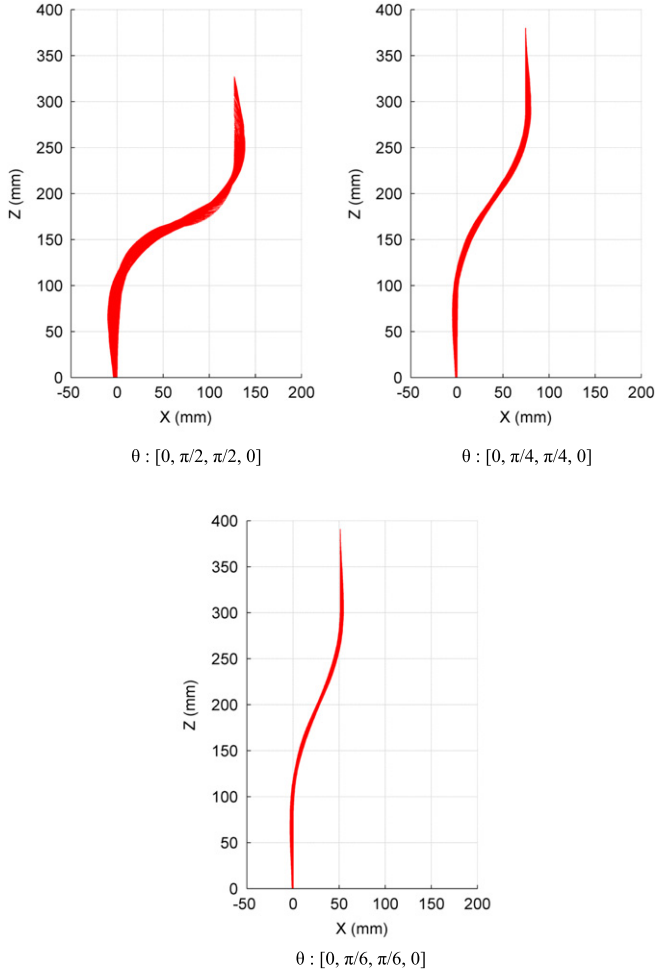

Fig. 8. Deviation in the path of 4-section S-bends.

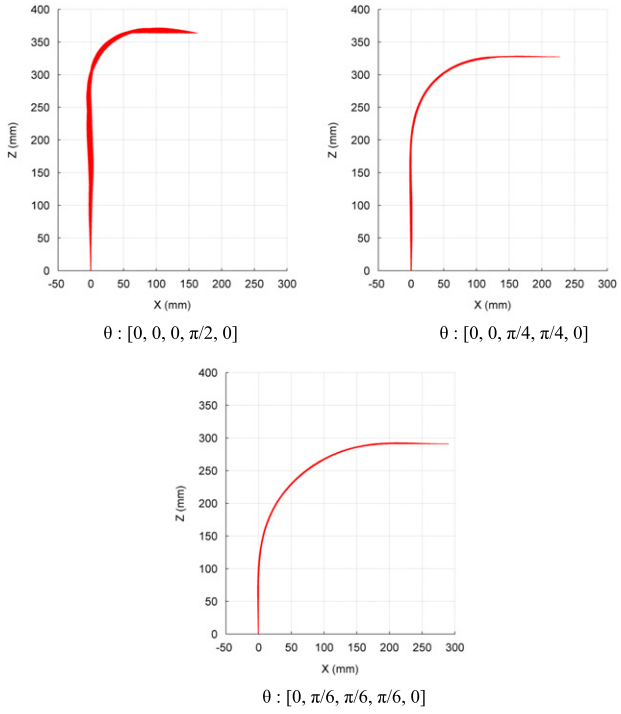

Fig. 9. Deviation in the path of different right angled bends.

Fig. 9 shows different right angled bends created by a $\pi/2$, $2\pi/4$ or $3\pi/6$ bends. Similar to the previous results, the tightest curve has 16.59 mm deviation, while the next has 1.95 mm and the smoothest curve has only 1.52 mm. The average computational times have increased around 0.85 s but are more varied. This is because there are now 5 sections in use and the paths have different levels of complexity. $\pi/2$ has the fastest average time, mostly because the first 250 mm can be straight, $\pi/6$ has the second fastest for the opposite reason, that the sections do not differ much during the bend.

Both the previous sets of graphs have been on a single plan, but this is not representative of real scenarios. As such Fig. 10 shows the deviation in two arrangements that operate in 3D. Fig. 10a has a max deviation of 4.03 mm whereas 10b has 8.61 mm. These profiles have seven sections, which have increased the average computation time per step to 0.16 s.

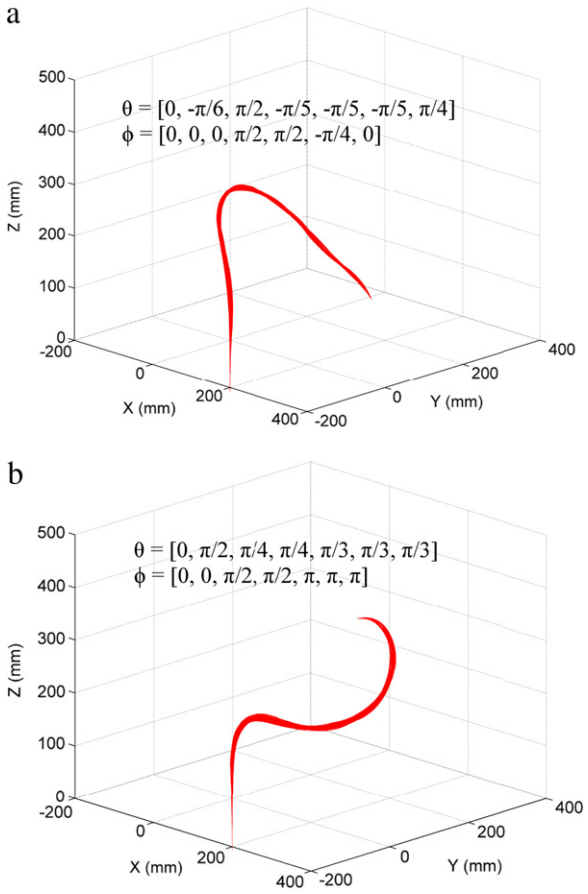

Fig. 10. Deviations in 7-section non-planar paths.

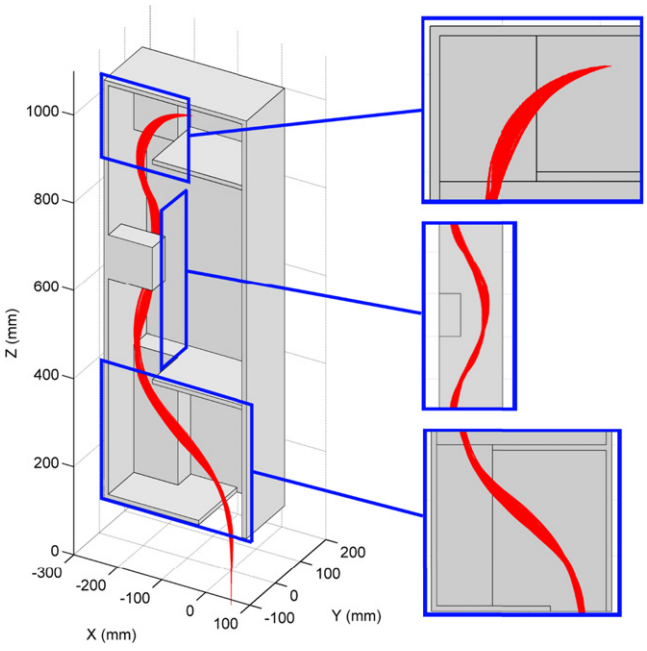

Fig. 11. Tip Following deviations within the generic scenario.

5.2. Simulation of the full system

The final simulation is to demonstrate the complete algorithm. In this test, a Snake Arm with 12-sections is navigated through the CAD model of the generic scenario. Fig. 11 shows the resulting envelope of all the poses taken to navigate through the environment.

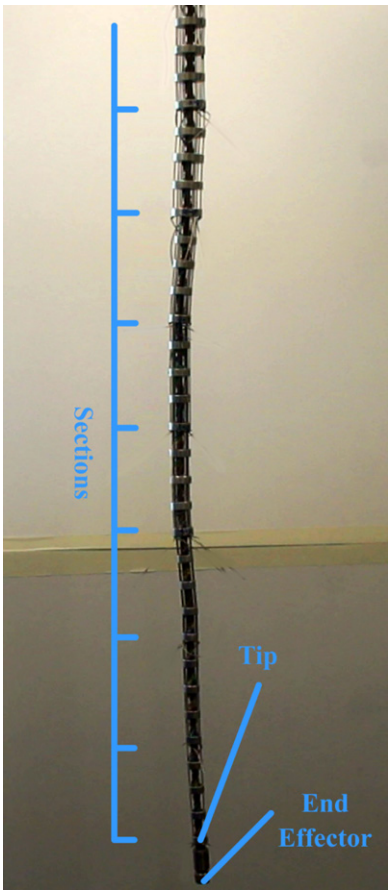

Fig. 12. 24 DoF Snake Arm used for trials.

The maximum deviation in this simulation is 20.15 mm and the maximum computational time for a step is 2.30 s, with an average time of 0.391 s per step. This means the Snake Arm could navigate this scenario with a tip speed of 4.5 mm/s while calculating in real-time without waiting for the computation. A supplementary video of this simulation can be found online at http://dx.doi.org/10.1016/j.robot.2014.05.013.

5.3. Demonstration on physical hardware

To verify that the system performs as expect with physical hardware, the algorithm was performed on a 24 DoF Snake Arm, shown in Fig. 12. The robot is controlled using LabView with a National Instruments single-board RIO, while the Tip Following is calculated at the same time on the desktop. This manipulator does not have a feed-in mechanism. It is not the intention of this paper to discuss the design of this snake arm in detail.

The program buffers the computed poses from the Tip Following and sends each one to the robot with a step rate greater than the expected average time. With this method the snake completes the motion without delays waiting for the calculations. The sequence depicted in Fig. 13 was completed in 118.3 s and performed at a rate of 0.2 s per step with a deviation of 14.71 mm.

6. Conclusions

This paper presented a novel technique for navigating a Snake Arm without the need for path planning or well-defined environments. With an average computation time of 0.410 s, for snake arms with 24 DoF, it is possible to advance the tip at a speed of 4.5 mm/s without interrupting the motion. For most arrangements

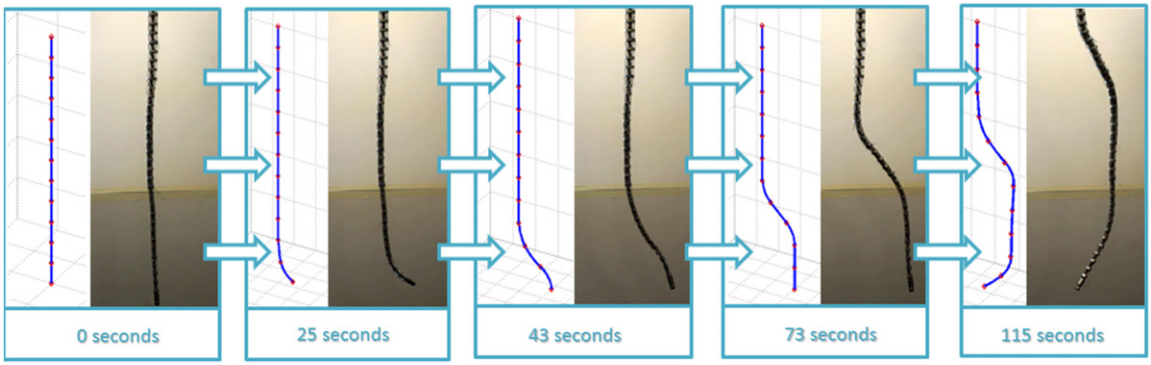

Fig. 13. Photoseries of a Snake Arm performing Tip Following.

the deviations in the path can be kept below 15 mm, and a tip accuracy of ±0.75 mm. Considering the typical in-situ scenario the snake arm is intended for, 4 min to navigate is a vast improvement compared to the days of labour needed to make the part accessible.

It is potentially possible to decrease the computational time further by exploring optimization processes on-board a real-time operating system (OS) rather than general purpose OS like Windows©. With a real-time OS, there is greater control of the task priority and less tasks competing for CPU time.

Currently the algorithm is limited to continuum designs; an extension to this work could include functionality for Snake Arms that can vary the section lengths or with more discrete and rigid designs.

Further work will look at how the same techniques can be applied to the feed-in mechanism to create the motion using the redundant sections to take in the slack.

Acknowledgements

With thanks to the work by Dr. Mark Raffles and Xin Dong for the design and construction of the snake arm shown. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP72011) under grant agreement no. 284959 (MiRoR: http://www.miror.eu/) and Rolls-Royce Plc.

References

[1] R. Buckingham, A. Graham, Snaking around in a nuclear jungle, Indust. Robotan Int. J. 32 (2005) 120–127.
[2] H. Tsukagoshi, A. Kitagawa, M. Segawa, Active Hose: an artificial elephant's nose with maneuverability for rescue operation, in: IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA., vol. 3, 2001, pp. 2454–2459.
[3] G.S. Chirikjian, J.W. Burdick, Hyper-redundant robot mechanisms and their applications, in: IEEE/RSJ International Workshop on Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. vol. 1, 1991, pp. 185–190.
[4] C.Q. Li, C.D. Rahn, Design of continuous backbone, cable-driven robots, J. Mech. Des. 124 (2002) 265–271.
[5] G.Z. Lum, S.K. Mustafa, H.R. Lim, W.B. Lim, G. Yang, S.H. Yeo, Design and motion control of a cable-driven dexterous robotic arm, in: 2010 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2010, pp. 106–111.
[6] W. McMahan, V. Chitrakaran, M. Csencsits, D. Dawson, I.D. Walker, B.A. Jones, M. Pritts, D. Dienno, M. Grissom, C.D. Rahn, Field trials and testing of the OctArm continuum manipulator, in: Proceedings 2006 IEEE International Conference on Robotics and Automation. ICRA 2006, 2006, pp. 2336–2341.
[7] M. Ivanescu, N. Bizdoaca, D. Pana, Dynamic control for a tentacle manipulator with SMA actuators, in: IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03. vol. 2, 2003, pp. 2079–2084.
[8] R. Buckingham, Snake arm robots, Ind. Robot 29 (2002) 242–245.
[9] M. Schlemmer, On-line trajectory optimization for kinematically redundant robot-manipulators and avoidance of moving obstacles, in: IEEE International Conference on Robotics and Automation, 1996. Proceedings., vol. 1, 1996, pp. 474–479.
[10] G.S. Chirikjian, J.W. Burdick, An obstacle avoidance algorithm for hyper-redundant manipulators, in: IEEE International Conference on Robotics and Automation, 1990. Proceedings., vol. 1, 1990. pp. 625–631.
[11] H. Choset, W. Henning, A follow-the-leader approach to serpentine robot motion planning, J. Aerosp. Eng. 12 (2) (1999) 65–73.
[12] L.A. Lyons, R.J. Webster, R. Alterovitz, Motion planning for active cannulas, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009., 2009, pp. 801–806.
[13] A.A. Maciejewski, C.A. Klein, Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments, Int. J. Robot. Res. 4 (1985) 109–117.
[14] I.S. Godage, D.T. Branson, E. Guglielmino, D.G. Caldwell, Path planning for multisection continuum arms, in: International Conference on Mechatronics and Automation, ICMA, 2012, pp. 1208–1213.
[15] D. Palmer, First Year PhD Report, unpublished.
[16] R. Buckingham, System and Method for Controlling a Robotic Arm, US Patent, 2009.
[17] S. Neppalli, M.A. Csencsits, B.A. Jones, I. Walker, A geometrical approach to inverse kinematics for continuum manipulators, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008. 2008, pp. 3565–3570.
[18] S. Neppalli, B.A. Jones, Design, construction, and analysis of a continuum robot, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, 2007, pp. 1503–1507.
[19] S. Neppalli, M.A. Csencsits, B.A. Jones, I.D. Walker, Closed-form inverse kinematics for continuum manipulators, Adv. Robot. 23 (2009) 2077–2091.
[20] G.S. Chirikjian, Hyper-redundant manipulator dynamics: a continuum approximation, Adv. Robot. 9 (3) (1995) 217–243.
[21] W. Park, Y. Wang, G.S. Chirikjian, The path-of-probability algorithm for steering and feedback control of flexible needles, Int. J. Robot. Res. 29 (7) (2010) 813–830.

David Palmer graduated with an M.Eng. in Cybernetics from the University of Reading, United Kingdom in 2011. He is currently studying as a Ph.D. candidate at University of Nottingham.

Salvador Cobos Guzman received his B.Sc. in Industrial Robotics from National Polytechnic Institute (I.P.N.), Mexico, in July 2003, another Engineering degree in Automation and Industrial Electronics from Polytechnic University of Madrid (U.P.M), Spain, in 2009, the Advances Studies Diploma in Robotics and Automation from Polytechnic University of Madrid (U.P.M), Spain, in 2007, and his Ph.D. with honours (''Sobresaliente Cum Laude'') in Robotics and Automation from Polytechnic University of Madrid (U.P.M), Spain, in 2010. He is currently a research fellow at Machining and Condition Monitoring research Group at University of Nottingham, United Kingdom.

Dragos Axinte is Professor of Manufacturing Engineering in the Department of M3. He held two NATO Research Fellowships in Italy and Denmark and then moved to UK to carry out research with University of Birmingham and later with University of Nottingham. He was appointed as Lecturer in Manufacturing Engineering (2005) and successively promoted to Associate Professor (2007), Reader (2010) and Professor (2011). Since 2009 he is Director of Rolls-Royce University Technology Centre in Manufacturing Technology and from 2006 he acts as Guest Professor at Royal Institute of Technology (KTH), Sweden where he teaches at M.Sc. level. He is Fellow of Institution of Mechanical Engineer (IMechE) and Fellow of International Academy of Production Engineering (FCIRP).